Pattern Analysis & Applications (2000)3:19–30

Nearest Neighbour Editing and Condensing Tools–Synergy Exploitation

B. V. Dasarathy[1], J. S. Sánchez[2] and S. Townsend[1]

[1]*Dynetics, Inc., Huntsville, AL, USA;* [2]*Department d'Informàtica, Universitat Jaume I, Castelló, Spain*

Abstract: The objective of this study has been to explore and exploit the synergy among the Nearest Neighbour (NN) editing and condensing tools previously reported in the literature in order to facilitate the use of NN techniques in near real-time applications. The extraordinary progress in the computer field has made NN techniques, once considered impractical from a computational viewpoint, feasible for consideration in time-constrained, real-world applications. This study accordingly addresses the issue of minimising the computational resource requirements of NN techniques, memory as well as time, through the use of prototype reduction techniques such as Minimal Consistent Set (MCS) selection while preserving the performance quality through suitable editing techniques, such as Proximity Graphs (PG). The tools employed in this investigation are first described briefly. Results of experiments conducted on well known data sets in the literature with various combinations of editing and condensing tools are then presented and discussed to assess the benefits of synergy among these tools. These results demonstrate the potential benefits of such synergy, and highlight the desirability of a more thorough exploration of combinations of other alternative editing and condensing tools that have been reported in the literature over the past few decades.

Keywords: Editing and condensing tools; Nearest neighbour; Synergy exploitation

1. INTRODUCTION

The advent of the computer revolution of the 1990s, in terms of inexpensive memory and high processing speeds, has created a resurgence of interest in Nearest Neighbour (NN) techniques, and has brought the NN rule to the forefront as a popular non-parametric classification technique. Given a set of N previously labelled prototypes (a training set), this rule assigns a sample to the same class as the closest prototype in the set, according to a measure of dissimilarity in the feature space. Apart from other advantages common to most non-parametric classification approaches, the NN rule and its extension to k neighbours (or the k-NN rule, in which the k closest neighbours 'vote' for the label of the sample) combine their conceptual simplicity with the fact that their asymptotic or infinite sample size error is less than twice the Bayes classification error [1].

Unfortunately, the NN rules are also saddled with some significant drawbacks. First, for a large set of prototypes of high dimensionality, the use of these approaches in real-time applications becomes computationally burdensome (although less so than before with the advances made in the computer industry), due to the large number of distances to be computed for each test sample. Secondly, the training set may contain noisy or erroneously labelled prototypes which usually lead to a decrease in performance.

Two distinct methodologies have been proposed to minimise these problems. The first, which is prototype generation, creates a new prototype set by using suitably weighted averages of the original training set. The second category, Prototype Selection (PS), represents the techniques that are studied in this paper. PS consists of selecting a particular subset of prototypes and applying the NN rule using only the samples selected. Two different families of PS methods exist in the literature [2,3]. First, the reducing or condensing algorithms aim at selecting the minimal subset of prototypes that lead to (approximately) the same performance as the NN rule using the whole training set [4–8]. Secondly, editing algorithms eliminate erroneously labelled prototypes from the original set and 'clean' the overlapping among regions from different classes. These techniques tend to offer improvements in performance [9–12].

Although the two PS perspectives are not always distinct, and are indeed largely merged in many studies, it is worth

Received: 18 January 1999
Received in revised form: 17 May 1999
Accepted: 8 June 1999

mentioning that the heuristic nature of most condensing algorithms contrasts with the strong statistical foundation of the most popular edited NN rules. Nevertheless, it has been observed that asymptotically optimal edited NN rules, such as the well known Multiedit algorithm [3], may lead to arbitrarily bad classification results if the number of prototypes is not large enough compared to the intrinsic dimensionality of the feature space. This makes editing a more critical problem than condensing. This fact has motivated a number of improvements and alternatives to classical editing algorithms for finite sample size problems [13,14].

Another distinction between editing and condensing arises from the subtle differences in purpose. Since editing is used to clean erroneously labelled samples from the training set, the main goal is to improve recognition accuracy by producing a sterilised set. The fact that a computational advantage may be gained is a secondary benefit. Condensing, however, is used primarily for the purpose of reducing the number of samples to gain a computational advantage. Even though it is done in a manner that attempts to minimise the change in recognition accuracy, an unfortunate characteristic of many condensing procedures is that they can often result in marginally poorer recognition performance. As a result, the amount of data set reduction due to editing is often small when compared to condensing methods, but the recognition accuracy for edited training sets is better. It is because of these distinctions that, while editing is often preferable to the analyst, condensing is of the most practical importance to the engineer developing a pattern recognition system for deployment in applications with real-time constraints.

This distinction in purpose and performance between editing and condensing suggests a synergistic relationship between them that begs to be exploited. The idea is that, by using them in conjunction, the improved recognition induced by editing can be combined with the larger reduction provided by condensing tools to produce a training set that is significantly smaller than the original with similar or better recognition capabilities. This composite approach improves performance both computationally and in terms of recognition. In short, it offers the best of both worlds.

The focus of this paper is therefore primarily on exploring the joint use of editing and condensing methods for the NN rule, and not on increasing the theoretical knowledge of NN techniques. An investigation of the practical implications of jointly using editing and condensing methods is needed to consider applying the techniques to real-world problems. The goal is thus to help practitioners in evaluating the practical use of prototype selection to increase the use of the NN rule.

More specifically, this work concentrates on the joint application of the MCS algorithm [8] with the methods based on Proximity Graphs (PG) [14]. The two methods appear to be a natural fit, because the MCS algorithm has been shown to be a robust means of condensing data, with minimal reductions in accuracy, while the PG-based editing schemes have been able to produce training sets with improved accuracy and decent reduction rates for editing.

Another point of investigation in this paper is the ordering of methods, especially when multiple condensing methods are used. Normally, editing is applied followed by condensing, which is the natural order based upon the goals of cleaning and reduction. The questions arise as to whether this order matters, and can multiple condensing methods be used to better improve performance. We investigate these questions by using one editing and one condensing method, as well as two condensing methods (a PG-based condensing method and MCS) in varied sequences. The details of the PG-based methods and the MCS algorithm are provided in Sections 2 and 3, respectively. Section 4 provides a description of our experiments, with their results in Section 5. The final section presents some conclusions and directions for further research. An appendix detailing the PG methodology is also included.

2. PROTOTYPE SELECTION USING PROXIMITY GRAPHS

Some special cases of PG, such as the Gabriel Graph (GG) or the Relative Neighbourhood Graph (RNG), have recently been used to introduce a set of editing and editing-condensing methods [14] for the NN rule. These approaches try to set a geometrical relation between a sample and some of its

Table 1. Editing/condensing experimental combinations

Experiment	Description
TRN	No Preprocessing
TRN MCS	MCS Only
GG1	GG1 Editing
GG1 MCS	GG1 Editing and MCS Condensing
MCS GG1	MCS Condensing and GG1 Editing
GG1C	GG1 Editing and PG-condensing
GG1C MCS	GG1 Editing, PG-condensing, and MCS Condensing
MCS GG1C	MCS Condensing, GG1 Editing, and PG-condensing
GG2	GG2 Editing
GG2 MCS	GG2 Editing and MCS Condensing
MCS GG2	MCS Condensing and GG2 Editing
GG2C	GG2 Editing and PG-condensing
GG2C MCS	GG2 Editing, PG-condensing, and MCS Condensing
MCS GG2C	MCS Condensing, GG2 Editing, and PG-condensing
RNG1	RNG1 Editing
RNG1 MCS	RNG1 Editing and MCS Condensing
MCS RNG1	MCS Condensing and RNG1 Editing
RNG1C	RNG1 Editing and PG-condensing
RNG1C MCS	RNG1 Editing, PG-condensing, and MCS Condensing
MCS RNG1C	MCS Condensing, RNG1 Editing, and PG-condensing
RNG2	RNG2 Editing
RNG2 MCS	RNG2 Editing and MCS Condensing
MCS RNG2	MCS Condensing and RNG2 Editing
RNG2C	RNG2 Editing and PG-condensing
RNG2C MCS	RNG2 Editing, PG-condensing, and MCS Condensing
MCS RNG2C	MCS Condensing, RNG2 Editing, and PG-condensing

neighbours in order to improve the performance of the NN rule by using a suitably reduced set of prototypes.

PG editing consists of applying the general idea of Wilson's algorithm [9], but using the graph neighbours of each sample instead of the Euclidean or other norm-based distance neighbourhood. Two samples x and y are graph neighbours in a PG, $G = (V,E)$ if there exists an edge $(x, y) \in E$ between them. Taking into account the definitions of GG and RNG [15,16], the graph neighbourhood of a given point requires that no other point lies inside the union of the zones of influence (i.e. hypersphere or lune of influence) corresponding to all its graph neighbours. (See Appendix I for a careful definition of GG and RNG, along with an algorithm to generate these graph structures.) From this neighbourhood relation, it seems possible to completely surround a prototype by means of a variable number of neighbours (that is, all its graph neighbours).

The application of PG to editing has some additional properties as compared to the conventional methods: first, they consider the number of neighbours as a variable feature which depends upon every prototype. Secondly, since the graph neighbourhood of a sample always tends to be widely distributed around it, the information extracted from samples close to decision boundaries (where uncertainty is much higher) may be richer in the sense of the distribution of prototypes.

The simplest PG editing approach [14] can be summarised as follows: after computing the graph neighbourhood of every sample in the input training set, all the graph neighbours of a sample (instead of just its k nearest neighbours) 'vote' for its class. In other words, all prototypes around a sample take part in the process of estimating whether it is an outlier or not, regardless of their actual distance to the sample.

1st order graph neighbourhood editing

Step 1 Construct the corresponding PG by means of the heuristic approach described in Appendix I.

Step 2 Discard those prototypes that are misclassified by their graph neighbours (by the usual voting criterion).

A further refinement of this general idea consists of taking not only the graph neighbours of a point, but also the neighbours of the graph neighbours from its same class (i.e.

Table 2. Recognition accuracies for the Iris database

	Recognition accuracy (%)					
	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
TRN	94.67	92.00	96.00	94.67	93.33	94.13
TRN MCS	92.00	90.67	92.00	94.67	94.67	92.80
GG1	94.67	92.00	98.67	94.67	90.67	94.14
GG1 MCS	92.00	97.33	97.33	93.33	92.00	94.40
MCS GG1	61.33	58.67	65.33	33.33	57.33	55.20
GG1C	94.67	92.00	98.67	94.67	90.67	94.14
GG1C MCS	92.00	92.00	97.33	92.00	90.67	92.80
MCS GG1C	61.33	58.67	65.33	0.00	57.33	48.53
GG2	94.67	92.00	98.67	94.67	93.33	94.67
GG2 MCS	92.00	90.67	97.33	94.67	94.67	93.87
MCS GG2	58.67	57.33	65.33	33.33	61.33	55.20
GG2C	94.67	92.00	98.67	94.67	92.00	94.40
GG2C MCS	90.67	89.33	97.33	93.33	90.67	92.27
MCS GG2C	56.00	57.33	65.33	0.00	58.67	47.47
RNG1	94.67	92.00	98.67	94.67	90.67	94.14
RNG1 MCS	97.33	97.33	94.67	94.67	90.67	94.93
MCS RNG1	58.67	57.33	65.33	60.00	58.67	60.00
RNG1C	93.33	90.67	96.00	93.33	92.00	93.07
RNG1C MCS	90.67	89.33	96.00	92.00	90.67	91.73
MCS RNG1C	58.67	57.33	65.33	62.67	58.67	60.53
RNG2	94.67	92.00	98.67	94.67	93.33	94.67
RNG2 MCS	97.33	97.33	96.00	94.67	94.67	96.00
MCS RNG2	52.00	57.33	65.33	61.33	60.00	59.20
RNG2C	92.00	90.67	93.33	93.33	90.67	92.00
RNG2C MCS	93.33	89.33	93.33	92.00	89.33	91.47
MCS RNG2C	52.00	57.33	65.33	61.33	58.67	58.93

some of the second level graph neighbours). Therefore, the previous algorithm can be modified in the following way:

2nd order graph neighbourhood editing

Step 1 Construct the corresponding PG by means of the heuristic approach described in Appendix I.

Step 2 For each sample, p, misclassified by its graph neighbours:

Step 2.1 Consider the subgraph, S, given by p and all its graph neighbours from its same class.

Step 2.2 Discard p if the graph neighbourhood of S has a majority of neighbours from a class other than p.

These editing algorithms have been combined with PG-based condensing [16] to define an editing-condensing approach using the same graph [14]. In fact, editing and condensing are two closely related and complementary techniques [3], and there exist some practical advantages if we are to apply both editing and condensing using PG approaches. In particular, computation can be saved if part of the proximity information used for editing can be reused for condensing.

Note that, to apply PG-based condensing [16] after the editing algorithms using the same graph structure, the edges must be recomputed when the discarded prototypes, along with all their edges, are removed from the graph. After this obvious step, it may be necessary to add new edges between pairs of non-neighbouring nodes. Nevertheless, from the definition of graph neighbours, it seems clear that only pairs of nodes whose zone of influence held an eliminated node can have a new edge. Otherwise, the reason which made a pair of samples non-neighbours still holds after editing.

Bearing this in mind, it is possible to store for each pair of non-neighbouring nodes the first detected node (hereafter called the *marked* node) inside its zone of influence, when the graph was computed for the first time (that is, before editing). This makes the selection of new edges much faster, because only the (few) pairs of nodes whose marked prototype has been discarded during editing need to make a search through the whole set. Thus, the combined editing-condensing algorithm can be written as follows:

Table 3. Training set reduction for the Iris database

	Training set reduction (%)					
	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
TRN	0.00	0.00	0.00	0.00	0.00	0.00
TRN MCS	89.33	88.00	84.00	89.33	90.67	88.27
GG1	5.33	5.33	6.67	6.67	4.00	5.60
GG1 MCS	96.00	96.00	93.33	93.33	94.67	94.67
MCS GG1	93.33	92.00	94.67	94.67	94.67	93.87
GG1C	73.33	72.00	82.67	74.67	74.67	75.47
GG1C MCS	94.67	94.67	93.33	93.33	94.67	94.13
MCS GG1C	93.33	92.00	97.33	100.00	94.67	95.47
GG2	0.00	1.33	5.33	2.67	0.00	1.87
GG2 MCS	89.33	88.00	93.33	92.00	90.67	90.67
MCS GG2	92.00	89.33	90.67	94.67	92.00	91.73
GG2C	65.33	65.33	81.33	66.67	68.00	69.33
GG2C MCS	89.33	89.33	93.33	92.00	90.67	90.93
MCS GG2C	93.33	89.33	94.67	100.00	93.33	94.13
RNG1	4.00	4.00	8.00	1.33	1.33	3.73
RNG1 MCS	96.00	96.00	94.67	92.00	96.00	94.93
MCS RNG1	97.33	97.33	93.33	94.67	94.67	95.47
RNG1C	88.00	89.33	94.67	89.33	89.33	90.13
RNG1C MCS	92.00	92.00	94.67	92.00	92.00	92.53
MCS RNG1C	97.33	97.33	97.33	96.00	94.67	96.53
RNG2	2.67	4.00	6.67	1.33	0.00	2.93
RNG2 MCS	96.00	96.00	94.67	92.00	90.67	93.87
MCS RNG2	96.00	97.33	93.33	92.00	93.33	94.40
RNG2C	89.33	89.33	94.67	89.33	89.33	90.40
RNG2C MCS	93.33	92.00	94.67	92.00	92.00	92.80
MCS RNG2C	97.33	97.33	97.33	93.33	94.67	96.00

Graph neighbourhood editing-condensing

Step 1 Construct the corresponding PG, G = (*V*,*E*) and, for each pair of points (*pi*, *pj*) ∈ *E*, mark the first node that lies inside its zone of influence.

Step 2 PG-based editing.

Step 3 Construct the subgraph, G′ = (*V*′, *E*′), corresponding to non-discarded nodes.

Step 4 For each pair from *V*′, (*pi*, *pj*) ∈ *E*′ and whose stored node is not in *V*′ put an edge if no other node from *V*′ lies inside its zone of influence.

Step 5 PG-based condensing [16] with the recomputed graph.

3. MINIMAL CONSISTENT SET SELECTION

MCS selection [8] is based on the concepts of NUNS, the Nearest Unlike Neighbour Subset [17], which can be looked upon as an optimal descriptor of the inter-class boundaries. The NUN subset is defined as the unique set of all samples which are the nearest unlike neighbours of one or more of the given samples. The nearest unlike neighbour of a sample $x \in$ class A is the sample $y \notin$ class A of the shortest distance from x. The properties of NUN sets and other related topics are covered elsewhere [2,8].

Based on this concept, we can see that for every given sample, the sufficient condition for its correct classification (i.e. for consistency) is the presence within MCS of a sample from its own class that is closer than its NUN (nearest unlike neighbour). Obviously, many samples independently satisfy this sufficiency condition for each given sample under consideration. This can be looked upon as a vote of confidence cast by the given sample and received by such closer-than-NUN samples. The sample with the most such votes (i.e. the sample that satisfies the consistency conditions for the most number of samples) therefore represents the prime candidate for inclusion in a MCS. Once this is picked, all the samples which were the voters contributing to the selection of the candidate for MCS can be disregarded from further consideration, and the vote counts of other candidates are reduced to reflect this. The candidate with the maximum votes after this update becomes the next most effective MCS sample. This process is repeated until all the voters have been taken into account, i.e. until full

Table 4. Distance measure for the Iris database

	Distance measure					
	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Avg
TRN	0.4733	0.4600	0.4800	0.4733	0.4667	0.4707
TRN MCS	0.6412	0.6318	0.6229	0.6508	0.6554	0.6404
GG1	0.4741	0.4608	0.4945	0.4745	0.4538	0.4715
GG1 MCS	0.6648	0.6836	0.6743	0.6600	0.6600	0.6685
MCS GG1	0.5584	0.5456	0.5751	0.5018	0.5534	0.5445
GG1C	0.5988	0.5841	0.6436	0.6029	0.5873	0.6033
GG1C MCS	0.6600	0.6600	0.6743	0.6553	0.6554	0.6609
MCS GG1C	0.5584	0.5456	0.5861	0.5000	0.5534	0.5355
GG2	0.4734	0.4600	0.4941	0.4735	0.4667	0.4734
GG2 MCS	0.6412	0.6318	0.6743	0.6600	0.6554	0.6525
MCS GG2	0.5456	0.5307	0.5588	0.5018	0.5528	0.5353
GG2C	0.5751	0.5642	0.6394	0.5789	0.5720	0.5856
GG2C MCS	0.6364	0.6317	0.6743	0.6553	0.6411	0.6477
MCS GG2C	0.5442	0.5307	0.5751	0.5000	0.5512	0.5271
RNG1	0.4738	0.4604	0.4950	0.4734	0.4534	0.4711
RNG1 MCS	0.6836	0.6836	0.6694	0.6600	0.6602	0.6713
MCS RNG1	0.5682	0.5648	0.5696	0.5604	0.5569	0.5638
RNG1C	0.6414	0.6364	0.6741	0.6460	0.6412	0.6478
RNG1C MCS	0.6458	0.6412	0.6741	0.6505	0.6458	0.6515
MCS RNG1C	0.5682	0.5648	0.5861	0.5732	0.5569	0.5697
RNG2	0.4735	0.4604	0.4945	0.4734	0.4667	0.4736
RNG2 MCS	0.6836	0.6836	0.6741	0.6600	0.6554	0.6713
MCS RNG2	0.5459	0.5648	0.5696	0.5528	0.5548	0.5571
RNG2C	0.6412	0.6364	0.6647	0.6460	0.6364	0.6449
RNG2C MCS	0.6600	0.6412	0.6647	0.6505	0.6412	0.6515
MCS RNG2C	0.5518	0.5648	0.5861	0.5584	0.5569	0.5632

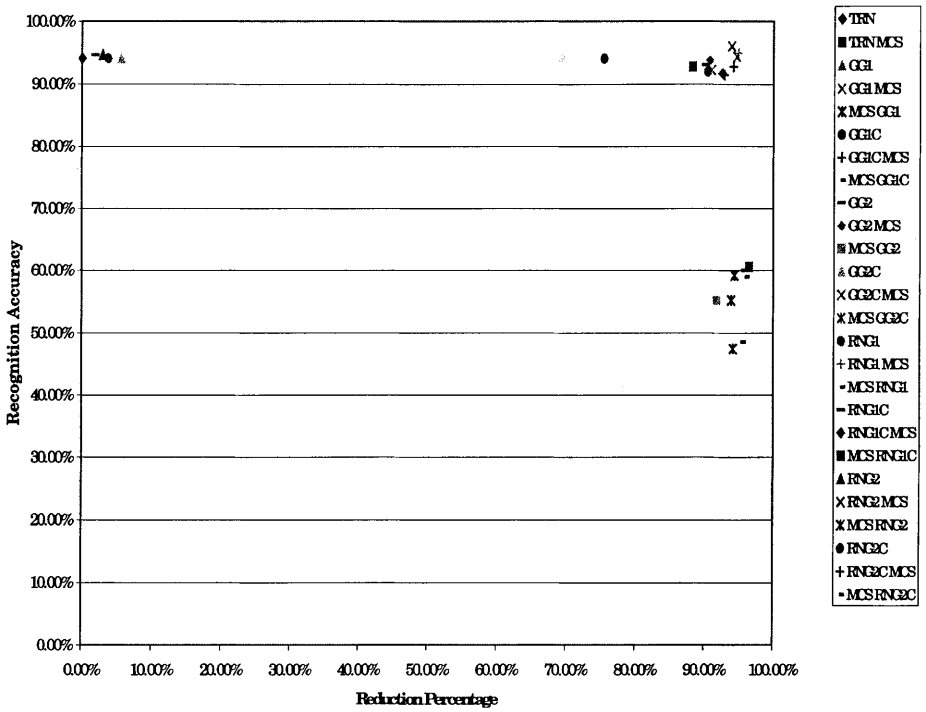

Fig. 1. Plot of average (recognition, reduction) – pairs for the Iris database.

consistency is achieved. It is of course possible that in some cases, the samples may have only one vote, i.e. of itself. In such cases, these automatically become MCS candidates. It is also possible that the voters to another sample may themselves become candidates for MCS.

Once a candidate MCS set has been identified, it is necessary to reexamine the problem as the effective NUN distances are now likely to be larger than before, as some NUNs are no longer in the subset under consideration. Thus, there is now scope for reducing the candidate MCS further. However, for the process to be monotonically reducing, we have to ensure that the candidate list will only include samples (other than the last MCS candidates) that will not create any new inconsistencies (see step 5 under the algorithmic procedure). This process is thus repeated until the set size can no longer be reduced.

MCS algorithmic procedure

Step 1 Define an initial consistent set to be the given training data set.

Step 2 For a specific sample, identify the associated nearest unlike neighbour and store the distance between the two samples.

Step 3 For this sample, identify all the neighbouring samples from its own class in the given data set which are closer than this NUN distance, and cast an approval vote to each of these samples in the given set by incrementing the corresponding vote registers, while noting this voter's (sample) identity by updating the corresponding voter lists.

Step 4 Repeat Steps 2 and 3 for all samples in the given training set, which results in a list of the number of votes received by each sample in the given set along with the records of identity of its voters.

Step 5 Create a potential candidate consistent set consisting of all samples in the given set which are either (a) already present in the current consistent set, or (b) whose inclusion will not create an inconsistency, i.e. the sample should not be nearer to any member of any other class than that member's current NUN distance.

Step 6 Identify the most voted sample in this candidate consistent list, designate it as a member of a newly selected consistent set, and identify all of its contributing voters.

Step 7 Delete these voters from all the voter lists wherein they currently appear, and correspondingly decrement the appropriate vote counts.

Step 8 Repeat Steps 6 and 7 until all the voters have been accounted for by the selected consistent set.

Step 9 Now with this selected consistent set, the NUN distances of the input samples are likely to be greater than before, as some of the original NUN samples may no longer be in the selected consistent set. Accordingly, repeat Step 2 using this selected consistent set to determine the NUN distance thresholds for each sample in the given set.

Step 10 Repeat Steps 3 through 8 using all the samples in the given set to identify a new consistent set.

This process of recursive application of Steps 2 through 8 is continued until the selected set is no longer getting smaller.

4. EXPERIMENTAL INVESTIGATION

To test the hypothesis, trials were conducted based on two data sets available in the literature. The first was the famous Iris data set, which consisted of three classes, each with 50 four-dimensional samples. Five unique experimental sets were created from the Iris data; each was created by partitioning the 150 samples into 75 training samples and 75 test samples. The second data set was the phoneme database used in the ROARS project [18]. The phoneme samples represent vowels that are divided into nasal (3818 samples) and oral (1586 samples) classes. Similar to the Iris data, five unique experimental sets were created. The experiment consisted of applying the NN rule to each of the ten experimental sets, where the training portion had been preprocessed using the editing and condensing combinations shown in Table 1.

From each trial, the reduction in the number of training samples and the recognition accuracy were recorded. The percentage reduction of training samples gives a direct measure of the amount of computational saving, which is the main goal. However, this reduction must not be made by compromising the ability of the NN technique to correctly classify the test set. The recognition accuracy provides a check on this trade-off. To balance these two competing goals, a normalised Euclidean distance between each (Reduction, Recognition) pair and the origin (0% reduction, 0% recognition) was calculated. (Note that this is really a weighted distance with the weights set to 1. As a consequence, if one of the measures of merit were to be of more importance in a particular application, then the corresponding weights could be adjusted appropriately to reflect this bias.) Using this measure, the 'best' combination was the one that produced the largest distance. Another way of visualising this is to plot the recognition accuracy versus the reduction percentage, and look for the point that is closest to the (100%, 100%) corner. For each data set, tables of the percentage reduction, reduction accuracy and normalised distance for each trial and an average for each

Table 5. Recognition accuracies for the Phoneme database

	Recognition accuracy (%)					
	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
TRN	88.56	88.82	88.75	88.27	88.19	88.52
TRN MCS	84.75	85.38	85.86	85.31	85.12	85.28
GG1	87.56	87.12	87.79	87.49	86.27	87.25
GG1 MCS	86.01	86.23	86.71	85.79	85.83	86.11
MCS GG1	83.53	83.90	82.72	83.53	82.72	83.28
GG1C	87.53	86.97	87.79	87.53	86.31	87.23
GG1C MCS	82.83	85.71	86.71	86.64	85.46	85.47
MCS GG1C	83.35	83.79	82.12	83.31	82.38	82.99
GG2	87.97	87.68	87.75	87.56	86.94	87.58
GG2 MCS	85.94	86.82	86.31	86.60	86.38	86.41
MCS GG2	83.90	84.64	83.53	83.97	84.05	84.02
GG2C	87.90	87.64	87.75	87.60	87.05	87.59
GG2C MCS	81.01	86.60	85.79	87.05	86.23	85.34
MCS GG2C	83.60	84.53	82.94	83.79	83.97	83.77
RNG1	88.64	87.82	88.97	87.49	86.05	87.79
RNG1 MCS	85.86	85.90	86.38	84.79	84.01	85.39
MCS RNG1	82.68	82.20	82.27	81.68	81.16	82.00
RNG1C	86.60	85.53	86.64	85.46	83.42	85.53
RNG1C MCS	84.90	73.02	76.39	83.05	82.28	79.93
MCS RNG1C	80.46	80.83	80.72	77.46	78.72	79.64
RNG2	88.71	88.42	88.93	87.71	86.60	88.07
RNG2 MCS	86.49	86.45	86.38	84.97	84.16	85.69
MCS RNG2	82.72	82.57	82.16	82.46	82.31	82.44
RNG2C	86.94	85.71	87.01	86.08	84.23	85.99
RNG2C MCS	80.31	73.61	78.05	83.97	82.53	79.69
MCS RNG2C	80.94	79.68	78.76	75.35	80.38	79.02

of these values over all five experimental sets have been provided. A plot of the percentage reduction verses reduction accuracy for the average values are also given.

5. EXPERIMENTAL RESULTS

For the Iris data (see Tables 2–4 and Fig. 1), the editing-condensing combinations provide 50–80% more reduction than editing alone. Of the individual combinations, the PG editing and MCS condensing provide a greater reduction than PG-based editing and condensing. These techniques also provide comparable recognition accuracies, so that overall they provide the best performance (using the normalised distance as the measure). By applying the condensing first, in the form of MCS, or by using both forms of condensing, slightly higher reduction rates can be obtained, but the associated loss in recognition accuracy more than offsets the advantage. This can be seen with a direct comparison of normalised distances, or in the plot, where the PG editing-condensing-MCS and MCS first symbols are offset from the associated PG-editing-MCS symbols. The results also indicate that condensing alone provides 'better' results than editing alone, underscoring why condensing is of such importance to engineers building real world pattern recognition systems.

The phoneme data (see Tables 5–7 and Fig. 2) indicates similar results with a couple of interesting differences. Any of the editing condensing combinations provide more reduction than editing alone (65–90% addition reduction) with comparable recognition accuracies. In fact, condensing often provides better recognition than editing alone. Again, the PG editing and MCS condensing achieve a higher reduction rate with similar recognition accuracies to the PG editing and condensing, and thus achieve the best results in terms of normalised distance. Also like the Iris data, the use of multiple condensing methods produced degraded results as compared to using only PG editing and MCS. In the phoneme data, however, this degradation is in terms of not only lower recognition accuracies, but also lower reduction rates. At first glance, this does not make sense because it would be expected that additional condensing would reduce the number of samples even more. The reason, however, is that when one of the condensing methods is applied, it changes the structure of the set on which the other method will work. For example, when PG condensing

Table 6. Training set reduction for the Phoneme database

	Training set reduction (%)					
	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Average
TRN	0.00	0.00	0.00	0.00	0.00	0.00
TRN MCS	78.05	78.16	78.05	77.68	77.65	77.92
GG1	12.73	13.29	12.92	13.06	13.62	13.12
GG1 MCS	90.93	91.08	90.93	90.86	91.67	91.10
MCS GG1	88.38	88.16	88.93	87.97	87.19	88.13
GG1C	67.25	68.17	67.73	67.58	70.32	68.21
GG1C MCS	91.04	90.82	90.64	90.86	91.78	91.03
MCS GG1C	91.97	90.82	91.27	90.60	90.16	90.96
GG2	8.88	8.96	9.03	9.07	9.66	9.12
GG2 MCS	88.42	89.05	89.23	88.79	89.67	89.03
MCS GG2	86.27	85.83	86.68	85.83	85.34	85.99
GG2C	62.69	63.47	63.21	63.40	65.80	63.72
GG2C MCS	88.79	88.79	89.16	89.12	89.93	89.16
MCS GG2C	90.04	89.08	89.53	88.79	88.19	89.13
RNG1	10.36	10.47	9.81	9.47	8.85	9.79
RNG1 MCS	89.79	89.90	89.16	89.45	88.90	89.44
MCS RNG1	90.38	91.15	90.86	89.75	88.79	90.19
RNG1C	82.20	82.16	82.27	82.31	81.64	82.12
RNG1C MCS	89.86	90.04	89.60	89.53	89.30	89.67
MCS RNG1C	95.08	95.78	95.26	94.45	94.19	94.95
RNG2	6.55	6.62	5.88	6.14	5.85	6.21
RNG2 MCS	87.64	87.71	86.82	87.45	86.68	87.26
MCS RNG2	88.53	89.45	88.56	87.16	86.23	87.99
RNG2C	79.39	79.31	78.13	79.09	79.02	78.99
RNG2C MCS	87.93	88.19	87.27	87.53	87.64	87.71
MCS RNG2C	93.93	94.93	94.15	93.30	92.30	93.72

Table 7. Distance measure for Phoneme database

	Distance measure					
	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Avg
TRN	0.4428	0.4441	0.4438	0.4414	0.4410	0.4426
TRN MCS	0.5761	0.5788	0.5802	0.5769	0.5761	0.5776
GG1	0.4424	0.4406	0.4437	0.4423	0.4367	0.4411
GG1 MCS	0.6258	0.6271	0.6282	0.6248	0.6279	0.6268
MCS GG1	0.6080	0.6085	0.6073	0.6066	0.6009	0.6063
GG1C	0.5519	0.5525	0.5544	0.5529	0.5566	0.5537
GG1C MCS	0.6154	0.6244	0.6272	0.6277	0.6271	0.6244
MCS GG1C	0.6206	0.6178	0.6139	0.6154	0.6106	0.6157
GG2	0.4421	0.4407	0.4411	0.4401	0.4374	0.4403
GG2 MCS	0.6165	0.6218	0.6207	0.6201	0.6226	0.6203
MCS GG2	0.6017	0.6027	0.6019	0.6004	0.5989	0.6011
GG2C	0.5398	0.5410	0.5407	0.5407	0.5456	0.5416
GG2C MCS	0.6009	0.6201	0.6186	0.6229	0.6230	0.6171
MCS GG2C	0.6143	0.6140	0.6102	0.6104	0.6089	0.6116
RNG1	0.4462	0.4422	0.4475	0.4400	0.4325	0.4417
RNG1 MCS	0.6212	0.6217	0.6207	0.6163	0.6116	0.6183
MCS RNG1	0.6125	0.6137	0.6129	0.6068	0.6015	0.6094
RNG1C	0.5970	0.5930	0.5974	0.5933	0.5836	0.5929
RNG1C MCS	0.6181	0.5797	0.5887	0.6106	0.6071	0.6008
MCS RNG1C	0.6228	0.6266	0.6243	0.6107	0.6138	0.6196
RNG2	0.4448	0.4433	0.4456	0.4396	0.4340	0.4415
RNG2C MCS	0.6157	0.6158	0.6124	0.6097	0.6041	0.6115
MCS RNG2	0.6058	0.6087	0.6040	0.5999	0.5960	0.6029
RNG2C	0.5887	0.5839	0.5847	0.5845	0.5775	0.5838
RNG2C MCS	0.5954	0.5744	0.5854	0.6065	0.6019	0.5927
MCS RNG2C	0.6200	0.6197	0.6138	0.5996	0.6120	0.6130

is applied to a set it 'cleans' up the set by removing erroneous samples. Since the 'clean set' and the original set have different internal structures (especially the NUN set), MCS may produce different consistent sets for them, with the 'clean' version being larger due to the altered NUN set inducing less removals. Another difference that should be pointed out is the significant degradation in recognition accuracy that occurs when MCS is applied first.

6. CONCLUDING REMARKS

The enormous progress in the computer industry over the past decade has made the deployment of nearest neighbour techniques in real-time applications a feasible proposition. However, making what is feasible truly practical requires additional creative thinking in the use of NN methodology. It is therefore advisable, from a practical viewpoint of minimising storage space and classification time, to identify the smallest possible sized prototype set. Nevertheless, theory dictates large training sets in order to approach optimal recognition accuracy. On the other hand, the presence of mislabelled prototypes can strongly degrade the classification accuracy. Thus, a trade-off between classification accuracy and training set size would offer the best compromise. Taking into account both issues, some preliminary conclusions can be drawn from the experiments. In general, PG-based editing achieves a high enough classification accuracy, but retains a very large number of prototypes. It seems that the use of PG without any condensing is of no practical value, since it would not meet the underlying real-world requirements for NN deployment in terms of a significant reduction in computational demands.

For both databases, the experiments indicated that the combination of PG editing with MCS condensing produce the best results in terms of balancing data set reduction for implementation purposes with recognition accuracy. While the choice of the particular PG editing technique is important when using combined PG editing-condensing, its effect is minimal when used in conjunction with MCS condensing. In fact, different 'best' performers were observed for each data set. All four were comparable, however, for both databases.

The results for the Iris and phoneme data sets both supported the conjecture that editing should be done before condensing. While the phoneme data results indicated the difference is only marginal, the Iris data showed the performance loss could be significant. This provides not only a

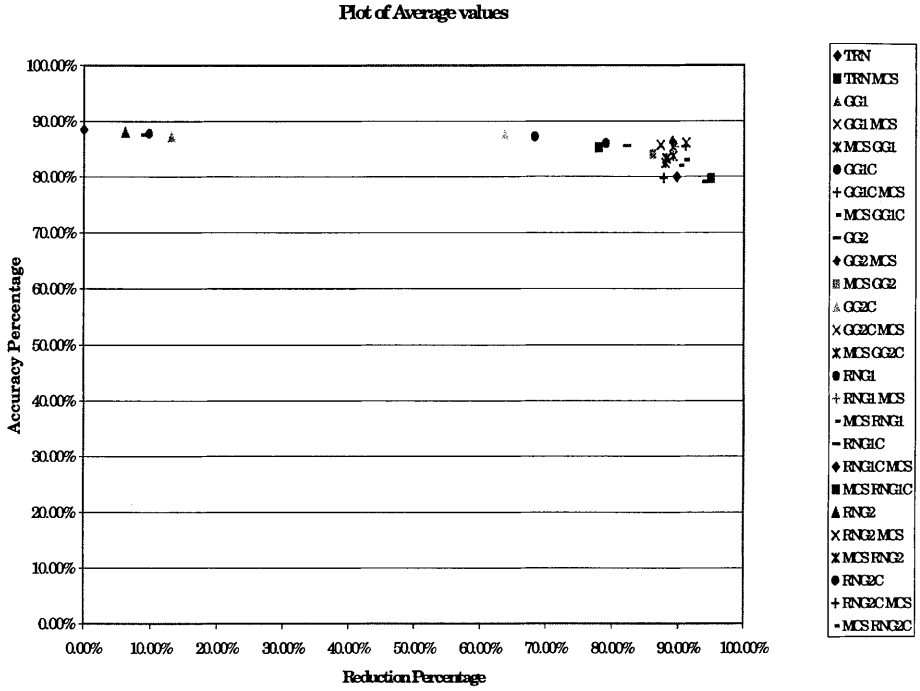

Fig. 2. Plot of the average (recognition, reduction) – pairs for the Phoneme database.

conceptual, but an experimental validation for using the editing-condensing order of application.

A final indication from the experiments, and perhaps the most surprising, is that using PG condensing and MCS condensing together not only fails to improve results, but can make them worse. A drop in recognition accuracy would have been foreseen, although probably not at the levels observed in the data, but the drop in the reduction rate that occurred with the Iris data was a surprise. This underscores the delicate balance that exists in using editing and condensing methods. The choice of methods must be studied to determine which are the best for a given application.

As a final comment, it should be pointed out that the computational run-time performance of the training phase algorithms has not been discussed. This was ignored, since PS is carried out just once prior to the implementation of a recognition system using the NN rule. Once the system has been trained, these computations need not be repeated. Hence, as long as the algorithms are computationally feasible, these relative orders of magnitude are unimportant. The computational requirements may be of relevance if an adaptive training of the pattern recognition system is contemplated. Hence, it should be pointed out that the PG method is much more computationally intense (by orders of magnitude) than MCS condensing. While it may be feasible to implement MCS condensing in a real or near real-time system, PG editing could not be used due to the $O(dn^2)$ expected complexity in calculating the initial PG (where n is the number of samples and d is the dimensionality).

While this work had focused on examining the benefits of using specific editing and condensing techniques in combination, there are other combinations that could be investigated. The results presented here should be viewed as a first step toward a more complete understanding of how to exploit the synergy between editing and condensing, through the identification of the optimal techniques for such exploitation.

References

1. Cover TM, Hart PE. Nearest neighbor pattern classification. IEEE Transactions on Information Theory 1967; 13:21–27
2. Dasarathy BV. Nearest neighbor (NN) norms: NN pattern classification techniques. IEEE Press, Los Alamitos, CA, 1990
3. Devijver PA, Kittler J. Pattern Recognition: A statistical approach. Prentice Hall, Englewood Cliffs, NJ, 1982
4. Hart PE. The condensed nearest neighbor rule. IEEE Transactions on Information Theory 1968; 14(3):515–516
5. Swonger CW. Sample set condensation for a condensed nearest neighbor decision rule for pattern recognition. Frontiers of Pattern Recognition Ed. S. Watanabe, Academic Press. 1972: 511–519
6. Gowda KC, Krishna G. The condensed nearest neighbor rule using the concept of mutual nearest neighborhood. IEEE Transactions on Information Theory 1979; 24(4):488–490
7. Fukunaga K, Mantock JM. Nonparametric data reduction. IEEE Transactions on Pattern Analysis and Machine Intelligence 1984; 6(1):115–118
8. Dasarathy BV. Minimal consistent subset (MCS) identification for optimal nearest neighbor decision systems design. IEEE Transactions on Systems, Man and Cybernetics 1994; 24:511–517
9. Wilson DL. Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems, Man and Cybernetics 1972; 2:408–421

10. Tomek I. An experiment with the edited nearest-neighbor rule. IEEE Transactions on Systems, Man and Cybernetics 1976; 6(6):448–452
11. Penrod CS, Wagner TJ. Another look at the edited nearest neighbor rule. IEEE Transactions on Systems, Man and Cybernetics 1977; 7(2):92–94
12. Broder AZ, Bruckstein AM, Koplowitz J. On the performance of edited nearest neighbor rules in high dimensions. IEEE Transactions on Systems, Man and Cybernetics 1985; 15(1):136–139
13. Kuncheva L. Editing for the k-nearest neighbors rule by a genetic algorithm. Pattern Recognition Letters 1995; 16:809–814
14. Sánchez JS, Pla F, Ferri FJ. Prototype selection for the nearest neighbor rule through proximity graphs. Pattern Recognition Letters 1997; 18:507–513
15. Jaromczyk JW, Tourssaint GT. Relative neighborhood graphs and their relatives. Proceedings of the IEEE 1992; 80:1502–1517
16. Toussaint GT, Bhattacharya BK, Poulsen RS. The application of Voronoi diagrams to nonparametric decision rules. Computer Science and Statistics 1985:97–108
17. Dasarathy BV. Nearest unlike neighbor (NUN) – an aid to decision confidence estimation. Optical Engineering 1995; 34(9): 2785–2792
18. Alinat P. Periodic progress report 4: ROARS project ESPRIT II 5516. Thomson Report 1993

Belur V. Dasarathy, Senior Principal Engineer at Dynetics, Inc., Huntsville, Alabama, USA, is engaged in R&D in the areas of pattern recognition, information fusion and related topics for the design and development of automated intelligent decision systems in defence and civilian applications. He is the editor-in-chief of *Information Fusion*, being launched in 1999 by Elsevier Science. He was the editor of several special issues on sensor fusion in the *Optical Engineering Journal*. He earned his PhD at the Indian Institute of Science, Bangalore, India, where he later served as one of the founding faculty at the School of Automation under the Electrical Sciences Division. He was honored in 1997 as the IEEE Region 3 Outstanding Engineer. Dr Dasarathy has over 170 publications and is the author of three IEEE Computer Society Press books: *Decision Fusion, Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, and *Image Data Compression: Block Truncation Coding*.

J.S. Sánchez received his MSc in computer science from the Technical University of Valencia in 1990, and his PhD in computer science engineering from the University Jaume I in 1998. Dr Sánchez is currently an Assistant Professor at the Department of Computer Science of the University Jaume I. His main research interests include pattern recognition, machine learning and computational geometry, particularly as applied to neighbourhood-based classification.

Sean Townsend received his masters degree in mathematics from the University of Kentucky in 1997. Mr Townsend is presently employed at Dynetics, Inc., Huntsville, Alabama, USA, as a Systems Analyst. His interests include pattern recognition, graph theory, coding theory, and sparse linear systems.

Correspondence and offprint requests to: Dr B.V. Dasarathy, Dynetics Inc., PO Box 5500, Huntsville, AL 35814–5500, USA. Email: belur.d@dynetics.com

APPENDIX I

Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a set of points in R^d, where n is the number of prototypes and d is the dimensionality of the feature space. A proximity graph, $G(V,E)$, is an undirected graph with the set of vertices $V = \mathcal{X}$, and a set of edges, E, such that $(x_i, x_j) \in E$ if and only if x_i and x_j satisfy some neighbourhood relation. In this case, we say that x_i and x_j are *graph neighbours*. The graph neighbours of a given point constitute its *graph neighbourhood*. The graph neighbourhood of a subset, $S \in V$, consists of the union of the graph neighbours of every node in S.

The proximity graphs used in this work are the GG and the RNG. Only essential concepts needed in this paper are reproduced here. Definitions for and properties of these structures are widely available in the literature [15,16].

Let $d(\cdot,\cdot)$ be the Euclidean distance in R^d. The GG is a proximity graph with the set of edges defined as follows:

$$(x_i, x_j) \in E \Leftrightarrow d^2(x_i, x_j) \leq d^2(x_i, x_k) + d^2(x_j, x_k) \quad \forall x_k \in \mathcal{X},\ k \neq i, j$$

In this case, x_i, x_j are said to be *Gabriel neighbours*. Its geometric interpretation is based on the concept of the hypersphere of influence between x_i and x_j: that is, two points are Gabriel neighbours if and only if there is no other point from $\mathcal{X}$ laying in the hypersphere centred at their middle point, and whose diameter is the distance between them.

On the other hand, the set of edges in the RNG is defined as follows:

$$(x_i, x_j) \in E \Leftrightarrow d(x_i, x_j) \leq \max\,[d(x_i, x_k), d(x_j, x_k)] \quad \forall x_k \in \mathcal{X},\ k \neq i$$

Now, x_i, x_j are said to be *relative neighbours*. An equivalent definition is based on the concept of *lune* [7], defined as the disjoint intersection between two hyperspheres centred at x_i and x_j, and whose radii are equal to the distance between them. Two points are relative neighbours if and only if their lune does not contain other points from $\mathcal{X}$.

It is always possible to construct any of these graphs once all the graph neighbours of the input data set are known. These graph neighbours can be computed exhaustively by using the brute-force method (i.e. by testing all pairs of prototypes), with a complexity of $O(dn^3)$.

Nevertheless, in the case of the GG and RNG, the number of pairs of graph neighbours in a set of n points is, in general, very much less than the total number of pairs $n(n-1)/2$ considered in the brute-force method. Thus, if by some means we can reduce the number of pairs to be tested for graph neighbours, then the brute-force method will be even more efficient. This goal can be achieved with a heuristic approach [16], which considerably reduces the number of pairs to be tested, and whose computational cost is believed to be closer to $O(dn^2)$.

Let us consider a set of points $\mathcal{X}$. Let p be a point of the set whose graph neighbours we are interested in. Consider a point $q \in \mathcal{X}$. Draw a line $B(p,q)$ through q, which is perpendicular to the line joining p to q. Let $LH(B, p)$ be the half-space determined by $B(p,q)$, which contains the point p. Let $RH(B,p)$ be the half-space which does not contain the point p. Then, no point of the set in $RH(B,p)$ can be a graph neighbour of p.

Using the above heuristic, the algorithm to generate the GG or the RNG can be written as follows:

Heuristic algorithm

For each point p_i of the given input set X do:.

Step 1 Start with $N_i = \{p_1, p_2, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n\}$ as the set of potential graph neighbours of the point p_i.

Step 2 For each potential graph neighbour $p_r \in N_i$ do Step 3.

Step 3 For each point $p_k \in X$, $p_k \neq p_i \neq p_r$ do:

(i) Test whether p_k lies inside the hypersphere (or lune) of influence determined by p_i and p_r. If so, remove p_r from the set N_i and go to Step 2 with a new potential graph neighbour.

(ii) If p_k is contained in N_i, then test whether p_r lies inside the hypersphere (or lune) of influence between p_i and p_k. If so, remove p_k from the set N_i.

Step 4 Accept the remaining points of the set N_i as the graph neighbours of p_i.