Efficient Nearest Neighbor Classification with Data Reduction and Fast Search Algorithms*

J. S. Snchez, J. M. Sotoca, F. Pla
Dept. Llenguatges i Sistemes Informtics, Universitat Jaume I
Av. Sos Baynat s/n, E-12071 Castell de la Plana (Spain)
Email: {sanchez,sotoca,pla}@uji.es

Abstract – *The Nearest Neighbor classifier is one of the most popular non-parametric classification methods. It is very simple, intuitive and accurate in a great variety of real-world applications. Despite its simplicity and effectiveness, practical use of this decision rule has been historically limited due to its high storage requirements and the computational costs involved. In order to overcome these drawbacks, it is possible either to employ fast search algorithms or to use a training set size reduction scheme. The present paper provides a comparative analysis of fast search algorithms and data reduction techniques to assess their pros and cons from both theoretical and practical viewpoints.*

Keywords: Classification, nearest neighbor, data reduction, fast search algorithm.

1 Introduction

One of the most widely studied non-parametric classification approaches corresponds to the k-Nearest Neighbor (NN) decision rule [6, 7]. Given a set of n previously labeled instances (training set, TS) in a d-dimensional feature space, the k-NN classifier consists of assigning an input sample to the class most frequently represented among the k closest instances in the TS, according to a certain similarity measure. A particular case of this rule is when $k = 1$, in which an input sample is assigned to the class indicated by its closest neighbor.

The asymptotic classification error of the k-NN rule (i.e., when n grows to infinity) tends to the optimal Bayes error rate as $k \to \infty$ and $k/n \to 0$. Moreover, if $k = 1$, the error is bounded by approximately twice the Bayes error [6]. The optimal behavior of this rule in asymptotic classification performance combines with a conceptual and implementational simplicity, which makes it a powerful classification technique capable of dealing with arbitrarily complex problems, provided that there is a large enough TS available.

Unfortunately, in many practical situations this theoretical behavior can hardly be achieved because of certain inherent weaknesses that significantly reduce the effective applicability of k-NN classifiers in real-world domains. For example, the performance of these rules, as with any non-parametric classification approach, is extremely sensitive to incorrectness or imperfections in the TS. Consequently, various works have been devoted to improve the NN classification performance by eliminating outliers (i.e., noisy, atypical and mislabeled instances) from the original TS [9, 16, 17, 22, 24].

Nevertheless, the main problem using the NN rule with a large set of training instances of high dimensionality is the apparent necessity of a lot of memory and computational resources. This is why numerous investigations have been concerned with finding new approaches that are efficient with computations. Within this context, many fast algorithms to search for the NN have been proposed. Alternatively, data reduction techniques have been directed to reduce the TS size by selecting only the most relevant instances among all the available ones, or by generating new prototypes in locations accurately defined.

Considering these two approaches for reducing the computational requirements, the goal of this paper is to compare fast search algorithms and data reduction techniques for efficient NN classification. To this end, several algorithms from both categories are reviewed in Section 2 and Section 3, respectively. Section 4 empirically analyzes the main characteristics relative to these approaches. Finally, Section 5 summarizes the main results presented in this paper and provides a discussion on future research directions.

2 Data reduction techniques

One basic idea for attenuating the storage requirements and time complexity consists of reducing the TS size, but obviously without affecting considerably the classification performance. In fact, there may be situations in which there are too much data and this information in most cases is not equally useful for the training phase of a learning algorithm. Taking into account this situation, data reduction techniques have been proposed to choose the most suitable instances in the TS, which correspondingly provides some savings in classification time and storage needs.

Size reduction techniques can be sorted into two groups by distinguishing between *selection* and *abstraction* methods. While the former family consists of picking up a subset of the

*0-7803-8566-7/04/$20.00 © 2004 IEEE.

original training instances [8,9,13,16,24], the abstraction (or *generation*) group builds new artificial prototypes summarizing a number of similar instances [4, 12, 14, 15].

One problem with using the original instances is that they assume that optimal examples can be found in the original set but, in fact there may not be any vector located at the precise points that would make the most accurate learning algorithm. Thus, prototypes can be artificially generated to exist exactly where they are needed, if such locations can be accurately determined.

In Pattern Recognition and Machine Learning, it is widely accepted an alternative categorization of reduction methods based on the ultimate objective of each algorithm. Accordingly, one can distinguish between *editing* and *condensing* algorithms. These methods will be briefly reviewed in the next sections.

2.1 Editing

The editing (or *filtering*) approaches [9, 16, 17, 22, 24, 25] eliminate erroneous instances from the original TS and "clean" possible overlapping among regions from different classes. Thus the focus of editing is not on reducing the set size, but on defining a high quality TS by removing outliers. Nevertheless, as a by-product these algorithms also obtain some decrease in size and consequently, a reduction of the computational burden of the NN classifier.

Wilson [24] introduced the first editing proposal. Briefly, this consists of using the k-NN rule to estimate the class of each instance in the TS, and removing those whose class label does not agree with that of the majority of its k neighbors. Note that this algorithm tries to eliminate mislabeled instances from the TS as well as close border instances, smoothing the decision boundaries.

2.2 Condensing

Condensing techniques [8,13,14,20,21] aim at selecting a sufficiently small subset of training instances without a significant degradation of classification accuracy. Other terms, such as *pruning* or *thinning*, have also been employed to refer them. It is to be noted that condensing makes sense only when the classes are clustered and well-separated, which constitutes the focus of the editing algorithms.

Hart's algorithm [13] is the earliest attempt at minimizing the number of stored instances by retaining only a *consistent* subset of the original TS. A consistent subset, say S, of a set of instances, T, is some subset that correctly classifies every instance in T using the NN rule. Although there are usually many consistent subsets, one generally is interested in the *minimal* consistent subset (i.e., the subset with the minimum number of instances) to minimize the cost of storage and computing time. Unfortunately, Hart's algorithm cannot guarantee that the resulting subset is minimal in size.

3 Fast search algorithms

In order to lessen the computational burden of the NN classifier, it is also possible to employ fast search algorithms, that is, some procedure capable of finding the NN of an input sample with as few computations as possible [5,10,11,19,23,26]. In contrast to data reduction techniques, these algorithms do not lead to a decrease in memory requirements, but only in time complexity. In fact, savings in classification time are here due to the particular strategy used to search for the NN of the input sample.

Fast search algorithm can become especially interesting when the number of training instances available is not large enough and therefore, it is not possible to throw out potentially useful data. On the other hand, another characteristic of these techniques is that in general they do not require a learning phase: they involve mainly storing all the training examples, which will be further used to classify new samples.

3.1 Fukunaga's algorithm

Fukunaga and Narendra [11] proposed one of the first hierarchical methods for fast NN search. Their algorithm exploits the triangle inequality to reduce distance computations searching for a hierarchical decomposition of space.

This method recursively employs standard clustering techniques (in particular, the K-means algorithm) to effect the decomposition, and then combines the branch-and-bound technique with depth-first search over the resulting data structure to locate the optimal solution. During search, the triangle inequality implies that a cluster does not need to be explored if the input sample is far enough outside it.

3.2 k-Dimensional tree

Although many different uses of k-dimensional (k-d) trees [3, 10] have been devised, their purpose is always to hierarchically decompose space into a relatively small number of cells such that no cell contains too many instances This provides a fast way to access any input sample by position. We traverse down the hierarchy until we find the cell containing the object and then scan through the few instances in the cell to identify the right one.

Typical algorithms construct k-d trees by recursively partitioning the TS. Each node in the tree is defined by a plane through one of the dimensions that partitions the set of points into left/right (or up/down) sets, each with half the points of the parent node. These children are again partitioned into equal halves, using planes through a different dimension. Partitioning stops after $\log n$ levels, with each point in its own leaf cell. Alternate k-d tree construction algorithms insert points incrementally and divide the appropriate cell, although such trees can become seriously unbalanced.

3.3 Vantage point tree

Although most trees partition the data based on their positions along a dimension, the vantage point tree (vp-tree) [26] employs a different technique. At each node in the tree, it selects one of the data points to function as a vantage point, and the division of the remaining data points is based on their distances from it. This is accomplished by sorting the data

points based on their distances from the vantage point, then dividing them into a number of groups. The division is performed so that the number of data points in each partition is as even as possible. Because it is based on the distance from a point, the partitions of a vp-tree are spherical instead of rectangular.

Actually, a vantage point is a point selected from a vector space, or a set of data points. However, the choice of vantage points plays an important role in the performance of the classification algorithm. It is worth noting that, unlike Fukunaga's algorithm and$k$-d trees, this method is suited for non-Minkowski metrics. A disadvantage, however, is that in high dimensional spaces, the partitions become very thin. This results in an increase in the number of branches of the tree that must be searched.

4 Experiments and discussion

In this section, classical data reduction schemes are empirically compared with three well-known fast search algorithms. In order to evaluate the conditions under which each algorithm performs well, we have here employed four artificial databases whose characteristics can be fully controlled. The last experiment is over a real-problem database to assess the behavior of the techniques used.

For the data reduction methods, we have tested Wilson's editing, Hart's condensing, and the *combining* edited and condensed set. In this latter case, we have firstly applied Wilson's editing to the original TS in order to remove mislabeled instances and smooth the decision boundaries, and then Hart's algorithm has been used over the Wilson's edited set to further reduce the number of training examples. After preprocessing the TS by means of some data reduction method, the NN classifier has been applied to the test set.

Among the many fast NN search proposals that can be found in the literature, we have taken the Fukunaga's branch-and-bound scheme, the k-d tree approach, and the vp-tree method. Note that in all these cases, the training instances are directly used to classify the test samples without any preprocessing, as already remarked in the previous section.

4.1 Gaussian database

The first experiment has been carried out over a two-dimensional artificial data set containing two classes. One class is represented by a multivariate normal distribution with zero mean and standard deviation equal to 1, while the other class corresponds to a normal distribution with zero mean and standard deviation equal to 2. Each class consists of 1,250 training instances and 1,250 samples in an independent test set. As can be seen in Figure 1, this corresponds to a hard classification problem with a very high overlapping between regions from both classes. The theoretical Bayes error rate is 26.37%.

From results in Table 1, it is possible to draw some preliminary conclusions. In terms of error rate, it is clear that both Wilson's editing and the combining scheme provide the best results (about to 7% higher), despite using considerably less training instances than the other procedures. This is due to the fact that editing has eliminated the high overlapping between different class regions. Analogously, the application of Hart's algorithm without editing suffers from this high overlapping, giving a low reduction rate and a considerably high error percentage.

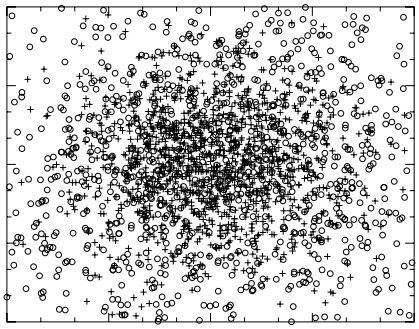

Figure 1: The Gaussian database.

Comparing the classification times required by each algorithm, one can see that the combining approach and the k-d tree excel to all other methods. Nevertheless, the storage requirements of the k-d tree (2,500 training instances) are much more important than those of the combining algorithm (only 59 examples).

Table 1: Error rate (% Err), number of training instances (Size), and classification time (Sec) for the Gaussian database

	% Err	Size	Sec
Wilson's	27.48	1,689	1.57
Hart's	36.88	1,362	1.22
Combining	27.28	59	0.05
vp-tree	35.68	2,500	0.84
Fukunaga's	35.68	2,500	0.30
k-d tree	35.68	2,500	0.02

4.2 Waveform21 database

This corresponds to a well-known artificial database taken from the UCI Database Repository [18]. The total number of instances is 5,000 (4,000 for training and 1,000 for test), belonging to three different classes (all classes contain approximately the same number of instances). Each instance is defined by 21 attributes.

Table 2 provides the experimental results for the Waveform21 database. In terms of computing time, it is to be remarked that Wilson's editing and fast search algorithms are

clearly worse than Hart's condensing and the combining approach. Moreover, both these methods also produce a very important reduction in storage needs, specially in the case of the combining scheme (83.4% of reduction percentage).

Table 2: Results for the Waveform21 database

	% Err	Size	Sec
Wilson's	17.32	3,267	4.33
Hart's	25.33	1,631	1.15
Combining	18.24	664	0.36
vp-tree	20.72	4,000	4.09
Fukunaga's	20.72	4,000	5.16
k-d tree	20.72	4,000	4.80

Finally, comparing the resulting error rates, one can observe that Wilson's algorithm and the combined editing-condensing technique produce the best classification performance. It is to be mentioned the fact that Hart's condensing obtains a higher error rate than the fast NN search algorithms.

4.3 Clouds database

The third experiment is also over a two-dimensional artificial database [1] in which the instances are from two classes (see Figure 2): one class is the sum of three different Gaussian distributions, while the other class is a single Gaussian distribution. Each class contains 2,500 instances (half for training and half for testing purposes). Both classes are relatively highly overlapped, and the decision boundaries are highly nonlinear. The optimal Bayes error is 9.66%.

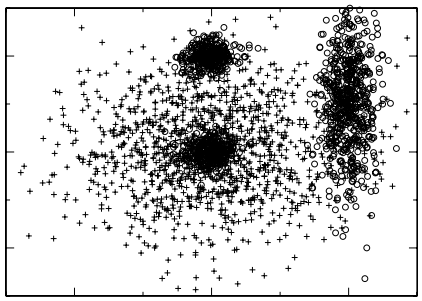

Figure 2: The Clouds database.

The results given in Table 3 are very similar to those obtained in the first experiment. Thus Wilson's procedure and the combining approach provide the highest classification performance, while the k-d tree algorithm and the combining technique show the lowest classification time. With respect to the storage requirements, it is clear enough that the employment of editing and condensing obtains the highest decrease in size.

Table 3: Results for the Clouds database

	% Err	Size	Sec
Wilson's	11.12	2,202	2.08
Hart's	17.04	710	0.63
Combining	11.36	94	0.07
vp-tree	15.24	2,500	0.58
Fukunaga's	15.24	2,500	0.25
k-d tree	15.24	2,500	0.02

4.4 Concentric database

This experiment has been carried out over a two-dimensional database [1] with instances belonging to two different classes and uniform concentric circular distributions. The points of class ω_1 are uniformly distributed into a circle of radius 0.3 centered on (o.5, 0.5). On the other hand, the points of class ω_2 are uniformly distributed into a ring centered on (o.5, 0.5) with internal and external radii equal to 0.3 and 0.5, respectively. The theoretical error rate is 0%.

The graphical representation of the database is plotted in Figure 3. There are 2,500 instances, with 1,579 in class ω_1 and the remainder belonging to class ω_2. The whole data set has been divided into a TS and a test set, each one with a half of the original instances.

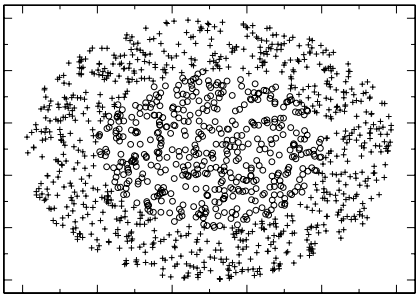

Figure 3: The Concentric database.

From the results shown in Table 4, one can see that in this case differences are not significant. Both fast NN search algorithms and data reduction techniques provide very similar error percentage and classification time. The differences with respect to the results obtained in the previous experiments can be explained taking into account the characteristics of the Concentric database: this constitutes a simple classification problem, in which there is no overlapping between the classes. Therefore, it seems clear enough that for this particular data set, the application of editing as a "cleaning" process is not necessary. In such a situation, fast NN search algorithms perform as well as the data reduction schemes, that is, all of them yield low error rates.

Table 4: Results for the Concentric database

	% Err	Size	Sec
Wilson's	2.72	1,232	0.56
Hart's	2.64	88	0.04
Combining	1.76	58	0.03
vp-tree	2.16	1,250	0.23
Fukunaga's	2.16	1,250	0.05
k-d tree	2.16	1,250	0.02

4.5 Satimage database

The last experiment has been performed over a real database [18]. The goal of this experiment is to classify the multispectral values of a real image (2,340 x 3,380 pixels) of the Landsat satellite. This data set is a subarea of a scene, consisting of 82 x 100 pixels. Thus each instance corresponds to a 3 x 3 square neighborhood of pixels completely contained within the subarea. There is a total of 6,435 patterns of 36 attributes (4 spectral bands x 9 pixels in neighborhood), belonging to six distinct classes. The original data set has been divided into a TS and a test set with 3,216 and 3,219 instances, respectively.

Table 5 shows the results for the experiments carried out over the Satimage database. Taking into account the three parameters here evaluated (error rate, size, and computing time), it seems that the combining approach excels any other technique. Although the error rate obtained with the combining scheme is some higher than that of the Wilson's editing, it is clearly superior in terms of storage reduction and computing time.

Table 5: Results for the Satimage database

	% Err	Size	Sec
Wilson's	14.01	2,953	17.63
Hart's	14.17	578	1.58
Combining	15.35	295	0.69
vp-tree	13.70	3,216	16.53
Fukunaga's	13.70	3,216	16.97
k-d tree	13.70	3,216	7.81

One can see that reduction schemes outperform the three fast NN search algorithms, even in terms of computing time. This can be due to the high dimensionality of the instances and the moderate size of the TS. For example, despite the poor size reduction of Wilson's algorithm (about 5%), computing time is almost equal to that of the fast search methods; only the k-d tree obtains a lower classification time than Wilson's editing.

5 Concluding remarks

The primary goal of this paper has been to compare fast search algorithms and data reduction techniques for efficient NN classification. Correspondingly, a number of theoretical characteristics of the two categories have been reviewed and then, an empirical analysis has been carried out over four artificial databases and one real data set in order to investigate both efficiency and accuracy of these methods.

While data reduction provides both time and storage savings, fast search algorithms can reduce the number of computations during classification but they still maintain the memory requirements. On the other hand, in many practical situations, data reduction methods also provide a better classification accuracy than that of the fast search algorithms, mainly when some editing algorithm is firstly employed to "clean" the original TS and then condensing is applied to the resulting edited set in order to obtain a higher decrease in size.

Focusing on the set of experiments carried out, one can see that the combining operation results specially useful in problems where there exists a high overlapping between regions from different classes. On the other hand, when the classes are not overlapped, it has to be admitted that both fast NN search algorithms and data reduction methods perform well, providing very similar results (in terms of classification accuracy and time).

Note that one could employ a fast search algorithm jointly with a data reduction scheme previously applied to the original TS. By this, it is expected to significantly improve efficiency (storage and computational requirements) while preserving or even increasing the NN classification performance. This study and a more extensive experimentation with other data reduction techniques and fast NN search algorithms constitute the most important directions for future research.

On the other hand, it could be also interesting to analyze the behavior of both these methods when applied to problems where one class is much less represented than the others in the corresponding TS. It has been observed that this situation, which arises in several practical domains, may produce a very significant deterioration in the classification performance, specially with instances belonging to the less represented class in the TS [2].

Acknowledgments

This work has been supported in part by grants TIC2003-08496 from the Spanish CICYT and P1-1B2002-07 from Fundaci Caixa Castell-Bancaixa. We would also like to thank Dr. Francisco Moreno-Seco for providing us the source code of the fast NN search algorithms employed in this paper.

References

[1] P. Alinat, *Periodic Progress Report 4: ROARS Project ESPRIT II 5516*, Thomson Report, 1993.

[2] R. Barandela, J. S. Snchez, V. Garca and E. Rangel, "Strategies for learning in class imbalance problems", Pattern Recognition, Vol. 36, pp. 849-851, 2003.

[3] J. L. Bentley, "Multidimensional binary search trees used for associative searching", Commun. ACM, Vol. 18, pp. 509-517, 1975.

[4] C. L. Chang, "Finding prototypes for nearest neighbor classifiers", IEEE Trans. on Computers, Vol. 23, pp. 1179-1184, 1974.

[5] E. Chavez, G. Navarro, R. A. Baeza-Yates and J. L. Marroquin, "Searching in metric spaces", ACM Computing Surveys, Vol. 33, pp. 273-321, 2001.

[6] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification", IEEE Trans. on Information Theory, Vol. 13, pp. 21-27, 1967.

[7] B. V. Dasarathy, *Nearest Neighbor Norms: NN Pattern Classification Techniques*, IEEE Computer Society Press, Los Alamos, CA, 1991.

[8] B. V. Dasarathy, "Minimal consistent subset (MCS) identification for optimal nearest neighbordecision systems design", IEEE Trans. on Systems, Man, and Cybernetics, Vol. 24, pp. 511-517, 1994.

[9] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, Englewood Cliffs, NJ: Prentice Hall, 1982.

[10] J. H. Friedman, J. L. Bentley and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time", ACM Trans. on Mathematical Software, Vol. 3, pp. 209-226, 1977.

[11] K. Fukunaga and M. Narendra, "A branch and bound algorithm for computing k nearest-neighbors", IEEE Trans. on Computing, Vol. 24, pp. 750-753, 1975.

[12] S. Geva and J. Sitte, "Adaptive nearest neighbor pattern classification", IEEE Trans. on Neural Networks, Vol. 2, pp. 318-322, 1991.

[13] P. E. Hart, "The condensed nearest neighbor rule", IEEE Trans. on Information Theory, Vol. 14, pp. 515-516, 1968.

[14] S.-W. Kim and B. J. Oommen, "Enhancing prototype reduction schemes with LVQ3-type algorithms", Pattern Recognition, Vol. 36, pp. 1083-1093, 2003.

[15] T. Kohonen, "The self-organizing map", Proc. of IEEE, Vol. 9, pp. 1464-1480, 1990.

[16] J. Koplowitz and T. A. Brown, "On the relation of performance to editing in nearest neighbor rules", Pattern Recognition, Vol. 13, pp. 251-255, 1981.

[17] L. I. Kuncheva, "Editing for the k-nearest neighbors rule by a genetic algorithm", Pattern Recognition Letters, Vol. 16, pp. 809-814, 1995.

[18] C. J. Merz and P. M. Murphy, UCI Repository of Machine Learning Databases. University of California, Irvine. `http://www.ics.uci.edu/~mlearn` (1998).

[19] L. Mic, J. Oncina and R. C. Carrasco, "A fast branch and bound nearest neighbour classifier in metric spaces", Pattern Recognition Letters, Vol. 17, pp. 731-739, 1996.

[20] R. A. Mollineda, F. J. Ferri and E. Vidal, "An efficient prototype merging strategy for the condensed 1-NN rule through class-conditional hierarchical clustering", Pattern Recognition, Vol. 35, pp. 2771-2782, 2002.

[21] G. L. Ritter, H. B. Woodruff, S. R. Lowry and T. L. Isenhour, "An algorithm for a selective nearest neighbour decision rule", IEEE Trans. on Information Theory, Vol. 21, pp. 665-669, 1975.

[22] I. Tomek, "An experiment with the edited nearest neighbor rule", IEEE Trans. on Systems, Man and Cybernetics, Vol. 6, pp. 448-452, 1976.

[23] E. Vidal, "An algorithm for finding the nearest neighbour in (approximately) constant time", Pattern Recognition Letters, Vol. 4, pp. 145-157, 1986.

[24] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data sets", IEEE Trans. on Systems, Man and Cybernetics, Vol. 2, pp. 408-421, 1972.

[25] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms", Machine Learning, Vol. 38, pp. 257-286, 2000.

[26] P. N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces", Proc. 4th. ACM-SIAM Symposium on Discrete Algorithms, Austin, pp. 311-321, 1993.