Extending Fast Nearest Neighbour Search Algorithms for Approximate k-NN Classification

Francisco Moreno-Seco, Luisa Micó, and Jose Oncina*

Dept. Lenguajes y Sistemas Informáticos
Universdad de Alicante, E-03071 Alicante, Spain
{paco,mico,oncina}@dlsi.ua.es

Abstract. The nearest neighbour (NN) and k-nearest neighbour (k-NN) classification rules have been widely used in pattern recognition due to its simplicity and good behaviour. Exhaustive nearest neighbour search can become unpractical when facing large training sets, high dimensional data or expensive similarity measures. In the last years a lot of NN search algorithms have been developed to overcome those problems, and many of them are based on traversing a data structure (usually a tree) and selecting several candidates until the nearest neighbour is found.
In this paper we propose a new classification rule that makes use of those selected (and usually discarded) prototypes. Several fast and widely known NN search algorithms have been extended with this rule obtaining classification results similar to those of a k-NN classifier without extra computational overhead.

Keywords: Nearest Neighbour, Classification Rule, Pattern Recognition.

1 Introduction

The nearest neighbour (NN) rule classifies an unknown sample into the class of its nearest neighbour according to some similarity measure (a *distance*). Despite its simplicity, classification accuracy is usually enough for many tasks. However, some tasks may require finding the k nearest neighbours in order to improve classification, thus the NN rule has been generalized to the k-NN rule [3]. Many classification tasks represent data as vectors and use one of the Minkowsky metrics as the distance, usually the L_2 (Euclidean distance). However, there are other tasks where a vector representation is not natural, and thus other distance measures are used: string distance, tree distance, etc.

Although initially used in pattern recognition, the NN rules have been also of interest for other fields such as data mining and information retrieval, which

* The authors wish to thank the Spanish CICyT for partial support of this work through project TIC2000–1703-CO3-02.

F.J. Perales et al. (Eds.): IbPRIA 2003, LNCS 2652, pp. 589–597, 2003.

usually involves searching in very large databases and facing with high dimensionality data. Whenever the classification task requires large training sets, expensive distance measures or high dimensionality, the simple exhaustive search for the NN becomes unpractical. To overcome some of these problems, a large number of fast NN search algorithms [5, 4, 18, 16, 2, 14] have been developed, and most of them can be easily extended to find the k-NN. However, the requirement of finding exactly the k-NN involves higher computing effort (dependent on the value of k).

Recently developed algorithms are suitable for any kind of representation which allows to define a distance that holds the properties of a metric, that is, they do not make use of the coordinates of the prototype. Several fast NN search algorithms are based on or can be viewed in an approximation and elimination framework [15]: a training prototype is selected as the current nearest neighbour, and then it is used to prune the training set and find the next candidate to nearest neighbour, until the training set is completely traversed or pruned. Then, the current candidate to nearest neighbour is actually the nearest neighbor.

The structure of the paper is as follows: in the next section we shall briefly describe the new classification rule. Then, we will show the results of this rule when applied to various NN search algorithms in experiments with synthetic and real data. Finally, we will conclude and outline some future work.

2 The k-NSN Classification Rule

In this paper we propose a simple but powerful extension for any approximation-elimination based NN search algorithm: when looking for the nearest neighbour, a number of candidates to nearest neighbour are selected until the actual nearest neighbour is found. We store the k nearest selected neighbours (k-NSN); at the end, the sample is classified by majority voting using these neighbours (which include the nearest neighbour). This technique can be considered a new classification rule which requires very little computational effort over a NN search (storing the k nearest selected neighbours). In addition, as we shall see in the next section, this rule achieves classification results very similar to those of the k-NN rule. Obviously, if this rule is applied to an exhaustive NN search it yields to the k-NN rule. This rule raises up as an extension of our previous work on the LAESA algorithm [13, 11].

Approximation and elimination search algorithms are usually based on the following idea: during preprocessing, a data structure is built to allow pruning of the training set. During classification, a candidate to nearest neighbour is selected and stored, and its distance to the sample is computed. This distance is used to prune the training set (using the data structure) and maybe to select a new candidate. This process ends when all the training set has been pruned or selected. Extending such an algorithm to find the k-NN is usually simple: the distance used to prune the training set is the distance to the kth nearest neighbour found so far. This involves less pruning and more distances to compute,

Table 1. Fast NN search algorithms which have been extended with the k-NSN rule

Algorithm	Author(s)
kd-tree	Friedman *et al.*[4]
FN75	Fukunaga and Narendra [5]
vp-tree	Yianilos [18]
AESA	Vidal [16]
LAESA	Micó *et al.* [13]
TLAESA	Micó *et al.* [12]
GNAT	Brin [2]

which derives in an additional computational overhead, always dependent on the value of k.

The k-NSN rule does not involve more significant overhead[1] than a typical NN search, and usually achieves similar results as a k-NN search. Of course, there is a drawback in our approach: the value of k may not be augmented indefinitely to improve classification rates; beyond a certain (big) value of k the rates start to worsen.

3 Experiments

We have performed several series of experiments in order to test the application of the k-NSN rule to some fast NN search algorithms (see table 1). All these algorithms fit in an approximation and elimination framework, and all are suited for general metric spaces except kd-tree, which requires point coordinates. The algorithms of AESA family (AESA, LAESA, TLAESA) focus on reducing the number of distance computations, thus are best suitable for expensive distances. The vp-tree and GNAT were developed to face large training sets and/or high dimensionality of data, and thus the number of distance computations is important but it is not its main goal.

Two sets of experiments have been performed: first, a set of synthetic data experiments to test the performance of the rule in a widely known environment. Then, some tests have been performed with several real data tasks. In both cases our main goal was to study the error rates of these algorithms using the k-NSN rule and to compare them with the k-NN error rates.

[1] The simplest implementation is to keep sorted an array of k elements each time a distance is computed. The extra time complexity over the NN search is $O(ck)$, where c is the number of computed distances. Although it is possible to reduce this time complexity with a heap, this overhead is almost negligible when compared to the overhead of computing c distances.

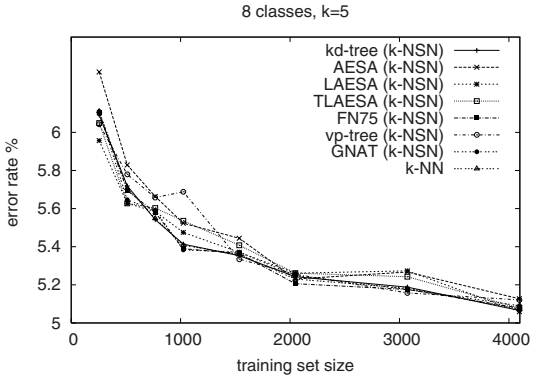

Fig. 1. Comparison between k-NN and k-NSN classifiers, for $k = 5$

3.1 Experiments with Synthetic Data

For these experiments we have generated Gaussian data from 4 and 8 classes of dimensionality 10 using the algorithm for generating clustered data in [8]. Tests have been performed for several values of k: 5, 11 and 17, and with training sets of growing size (from 256 to 4096). Test set had always 1024 prototypes. Also, 16 different train/test sets of each size were generated in order to obtain more sound results.

Figure 1 shows the error rates of these classifiers with data from 8 classes for $k = 5$ (results for 4 classes data were similar), and figure 2 plots the same results for $k = 17$ but including the NN error rate just to show the difference with respect to k-NN and k-NSN rates. These results show that the differences in error rates are negligible for k-NSN and k-NN classifiers, and are better than those of an NN classifier.

In order to study the behaviour of the k-NSN rule as the value of k increases another experiment was performed keeping the train and test sizes to 2048 and 1024 respectively. All the algorithms were run with 16 different train/test sets, and the average results are shown in figure 3. As can be seen in that figure, even for high values of k most of the k-NSN classifiers still obtain classification rates comparable to those of the k-NN classifier. Algorithms from AESA family seem to be very sensitive to an increase in the value of k. This may be due to the fact that they compute very few distances with respect to the others (i.e. they select less candidates to nearest neighbour). However, this question should be studied more carefully and we plan to do it in the future. Finally, note that with (almost) the same computational effort of finding the nearest neighbour, the error rates obtained with k-NSN are much better than NN rates and comparable to those of a k-NN classifier.

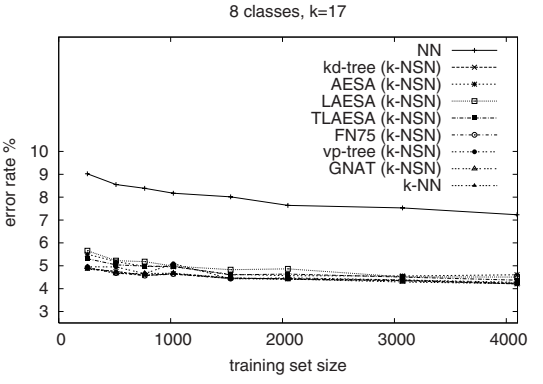

Fig. 2. Nearest neighbour error rate compared to k-NN and k-NSN

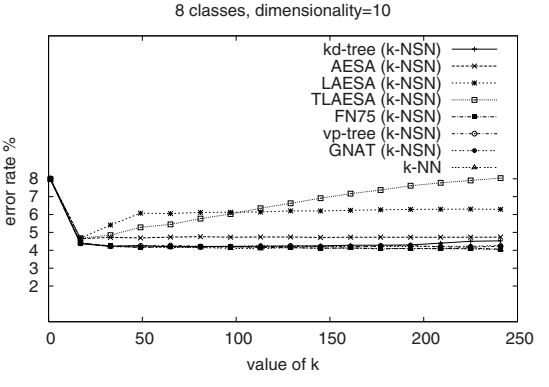

Fig. 3. Comparison between k-NN and k-NSN classifiers as k increases

3.2 Experiments with Real Data

We have performed experiments with two different data sets: first we have tested our rule with chromosome data [9, 7, 6] and then with the PHONEME database from the ROARS ESPRIT project [1].

The chromosome database contains 4400 samples coded as strings, and we have chosen to use the Levenshtein distance [17, 19] for this task (the kd-tree has not been tested with this database due to this feature). The database has been divided into two sets of 2200 samples each, and two experiments have been performed using one of them for training and the other one for test. Figure 4 shows the average error rates of k-NN and k-NSN classifiers as the value of k increases. There is a parameter for LAESA and TLAESA (see [13, 12] for more details), the number of *base prototypes*, which has been set to 40, which it is not probably its optimal value. However, the search for this optimal value is

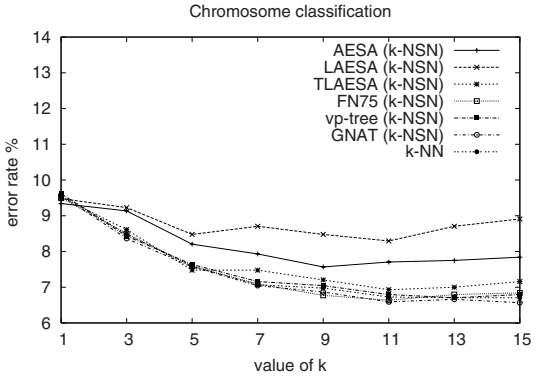

Fig. 4. Error rate comparison in chromosome classification

beyond the scope of this paper. Table 2 shows the average classification time per sample in seconds (on a 1.5 GHz PC under Linux); it is also shown in this table the average time for an extension of LAESA to find the k-NN, named k-LAESA [10], in order to allow a more fair comparison than with exhaustive search k-NN classifier. The value of k for the k-NSN algorithms does not appear because the average times are very similar for all values of k (as expected).

The PHONEME database consists of 5404 five-dimension vectors from 2 classes. Five different partitions have been made to obtain train/test sets of 4300/1000 samples approximately. The results plotted in figure 5 are the average of the five different runs, and show the error rates of k-NN and k-NSN classifiers as the value of k increases. The best results are obtained by LAESA and TLAESA. This may happen because the number of *base prototypes* was set to 40, which probably is higher than the optimum. Both LAESA and TLAESA compute more distances than all other algorithms.

Table 2. Ordered table of average classification time of chromosomes

Algorithm/rule	Time (secs.)
AESA (k-NSN)	0.024
TLAESA (k-NSN)	0.029
LAESA (k-NSN)	0.029
k-LAESA (k=5)	0.044
GNAT (k-NSN)	0.044
FN75 (k-NSN)	0.045
k-LAESA (k=15)	0.048
vp-tree (k-NSN)	0.060
k-NN (exhaustive)	0.091

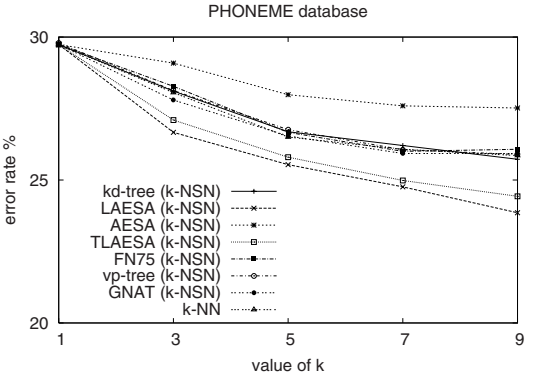

Fig. 5. Error rate comparison in phoneme classification

From this two sets of experiments we can conclude that for low values of k (the most often used) the error rates of the k-NSN rule are always better than those of a NN with the same computational cost, and it nearly reaches or even improves the k-NN error rate for some tasks. Thus, this new rule may be interesting for some real tasks because it may obtain better results than k-NN classifiers and improves NN error rates with (almost) no extra computational effort.

4 Conclusions and Future Work

A new NN based classification rule (the k-NSN rule) has been developed. Our experiments show that classification results similar to those of the k-NN rule are obtained using this rule with very little extra computational effort with respect to a NN classifier. The k-NSN rule is applicable to any approximation-elimination NN search algorithm. Whenever a fast approximation-elimination NN search algorithm is applicable, it may be easily modified to classify using the k-NSN rule and thus it may obtain error rates lower than those of NN, without the extra overhead of searching for the k-NN. Moreover, the time performance of k-NSN classifiers does not depend on the value of k. We have tested this rule with various well known NN fast search algorithms: kd-tree, Fukunaga and Narendra's, vp-tree, GNAT. We have also tested the rule with the algorithms of AESA family, which compute a very low number of distances.

There is still a lot of work to do to explore the possibilities and range of application of the k-NSN rule. As for the future, we plan to:

- study the evolution of k-NSN error rates as the value of k become higher than those tested in this work, and compare them with k-NN,
- extend the NN search algorithms we have implemented to find the k-NN, and then make a comparison with k-NSN rule studying error rates and time performance,

- test the performance of the k-NSN rule as the dimensionality or the number of classes increase, and
- apply the k-NSN rule to other approximation-elimination NN search algorithms.

Acknowledgments

The authors wish to thank Juan S. Sánchez and Alfons Juan for providing us the PHONEME database and the chromosomes database, respectively. We also would like to thank José Manuel Iñesta and the anonymous referees for their valuable comments.

References

[1] Alinat, P.: Periodic progress report 4, ROARS project ESPRIT II - Number 5516. Thomson Technical Report TS ASM 93/S/EGS/NC/079 (1993) 593

[2] Brin, S.: Near Neighbor Search in Large Metric Spaces. Proceedings of the 21^{st} VLDB Conference (1995) 574–584 590, 591

[3] Duda, R., Hart, P.: Pattern Classification and Scene Analysis. Wiley (1973) 589

[4] Friedman, J. H., Bentley, J. L., Finkel, R. A.: An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software (1977) **3** 209–226 590, 591

[5] Fukunaga, K., Narendra, M.: A branch and bound algorithm for computing k–nearest neighbors. IEEE Trans. Computing (1975) **24** 750–753 590, 591

[6] Granum, E., Thomason, M. G.: Automatically inferred Markov network models for classification of chromosomal band pattern structures. Cytometry (1990) **11** 26–39 593

[7] Granum, E., Thomason, M. G., Gregor, J.: On the use of automatically inferred Markov networks for chromosome analysis. In Automation of Cytogenetics, C. Lundsteen and J. Piper, eds., Springer-Verlag (1989) 233–251 593

[8] Jain, A. K., Dubes, R. C.: Algorithms for clustering data. Prentice-Hall (1988) 592

[9] Lundsteen, C., Phillip, J., Granum, E.: Quantitative analysis of 6985 digitized trypsin G-banded human metaphase chromosomes. Clinical Genetics (1980) **18** 355–370 593

[10] Moreno-Seco, F., Micó, L., Oncina, J.: Extending LAESA fast nearest neighbour algorithm to find the k nearest neighbours. Structural, Syntactic, and Statistical Pattern Recognition. Lecture Notes in Computer Science, T. Caelly et al (Eds.) vol. 2396, Springer-Verlag (2002) 691–699 594

[11] Moreno-Seco, F., Micó, L., Oncina, J.: A modification of the LAESA algorithm for approximated k-NN classification. Pattern Recognition Letters (2003) **24** (1-3) 47–53 590

[12] Micó, L., Oncina, J., Carrasco, R. C.: A fast branch and bound nearest neighbour classifier in metric spaces. Pattern Recognition Letters (1996) **17** 731–739 591, 593

[13] Micó, L., Oncina, J., Vidal, E.: A new version of the nearest neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing-time and memory requirements. Pattern Recognition Letters (1994) **15** 9–17 590, 591, 593

[14] Nene, S., Nayar, S.: A Simple Algorithm for Nearest Neighbor Search in High Dimensions. IEEE Transactions on Pattern Analysis and Machine Intelligence (1997) **19**(9) 989–1003 590

[15] Ramasubramanian, R., Paliwal, K. K.: Fast nearest-neighbor search algorithms based on approximation-elimination search. Pattern Recognition **33** (2000) 1497–1510 590

[16] Vidal, E.: New formulation and improvements of the Nearest-Neighbour Approximating and Eliminating Search Algorithm (AESA). Pattern Recognition Letters (1994) **15** 1–7 590, 591

[17] Wagner, R. A., Fischer, M. J.: The String-to-String Correction Problem. Journal of the Association for Computing Machinery (1974) **21**(1) 168–173 593

[18] Yianilos, P. N.: Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. ACM-SIAM Symposium on Discrete Algorithms (1993) 311–321 590, 591

[19] Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal of Computing (1989) **18** 1245–1262 593