A new classification rule based on nearest neighbour search

Francisco Moreno-Seco, Luisa Micó and Jose Oncina
Dept. Lenguajes y Sistemas Informáticos
University of Alicante
03080 Alicante, Spain
{paco,mico,oncina}@dlsi.ua.es

Abstract

The nearest neighbour (NN) classification rule is usually chosen in a large number of pattern recognition systems due to its simplicity and good behaviour. As the problem of finding the nearest neighbour of an unknown sample is also of interest in other scientific communities (very large databases, data mining, computational geometry, ...), a vast number of fast nearest neighbour search algorithms have been developed during the last years. In order to improve classification rates, the k-NN rule is often used instead of the NN rule, but it yields higher classification times. In this work we introduce a new classification rule applicable to many of those algorithms in order to obtain classification rates better than those of the nearest neighbour (similar to those of the k-NN rule) without significantly increasing classification time.

1. Introduction

The nearest neighbour (NN) rule [2, 5] classifies an unknown *sample* into the class of its nearest neighbour, according to some similarity measure (a *distance*). Its statistical properties have been extensively studied and firmly stated [5]. Given a distance, it is very simple to build up a classifier based on this rule, which often becomes unbeatable (or at least hard to beat) by other types of classifiers.

The easiest implementation of the NN rule consists on computing all the distances between the sample and each object (*prototype*) in the training set, and then the nearest neighbour is the prototype whose distance is the minimum; this is often referred as *exhaustive search* or *brute force search*. However, in some applications where the training set is large and/or the distance's computing time is high this approach becomes unpractical. During the last years, a large number of fast NN search algorithms have aroused from various scientific communities. Many of them focus on reducing search time when the training set is large, like for instance the k-d tree [3, 4], the Fukunaga and Narendra's branch and bound algorithm [6], the vp-tree [15] and the GNAT [1]. Other algorithms such as those from the AESA family [13, 14, 9, 8] focus mainly on reducing the number of distance computations to address classification tasks where the distance is very time consuming, as for instance when the objects are represented by strings and the distance is the Levenshtein distance (also called Edit distance).

The k nearest neighbours (k-NN) rule classifies the sample into the majority class of its k nearest neighbours. Although it is possible to find the k nearest neighbours using an exhaustive search, the usual way is to extend a fast NN search (like those mentioned above) to find the k nearest neighbours; however, the number of distance computations and the classification time both increase in a quantity which depends on the value of k.

In this work we present a new classification rule that uses k neighbours to classify the sample. This rule is based on fast NN search algorithms and obtains error rates similar to those of the k-NN rule without increasing distance computations nor classification time. A high percentage of the neighbours used by the new rule are in fact among the k nearest neighbours, and that percentage increases with the dimensionality, which closes the rule's error rates to those of the k-NN rule.

2 The approximation-elimination framework

Many fast NN search algorithms are based on an approximation-elimination framework [12], which consists mainly in two steps: first, a candidate to NN is selected using an approximation function, and second, the training set is pruned using the distance to the current nearest neighbour. Then, the search process ends when there are no selectable prototypes. More precisely, the approximation-elimination scheme may be summarized as follows:

1. Using a certain approximation function, select a candi-

date to nearest neighbour, compute its distance to the sample and update the current nearest neighbour.

2. Eliminate all those prototypes that cannot be closer than the current nearest neighbour to the sample, i.e., those prototypes which are outside a hyphersphere centered on the sample with radius the distance to the current nearest neighbour.

3. If there are still prototypes in the training set which have not been eliminated or selected, go to step 1.

The approximation-elimination scheme may also be formulated in algorithmic notation:

$d_{nn} := \infty$
until the training set P is empty **do**
    $p_i := argmin_{p \in P}$Aprox(x, p) *// Approximation*
    $d := d(x, p)$
    if $d < d_{nn}$ **then**
(a) $nn := p$; $d_{nn} := d$ *// Update the current NN*
(b) $P := P - \{ \ q : q \notin E(x, d_{nn}) \ \}$ *// Elimination*
    endif
enddo

The search for the k nearest neighbours may be performed under this scheme by simply updating the list of the k nearest neighbours in line labelled (a) in the algorithm above (instead of updating only the nearest neighbour), and using the distance to the k-th nearest neighbour (instead of d_{nn}) to prune the training set in line labelled (b). These two modifications allow any algorithm which fits under the approximation-elimination framework to be extended to find the k nearest neighbours.

3 A new classification rule

As mentioned above, the search for the k nearest neighbours implies a computational overhead that depends on the value of k. On the other hand, the classification rates of the k-NN rule are usually much better than those of the NN rule, and that is why the k-NN rule is applied in many real tasks instead of the NN rule.

During the search for the nearest neighbour (under an approximation-elimination framework) a number of prototypes are selected and usually discarded. Only if their distance to the sample is the minimum in that step of the algorithm they are stored as the current nearest neighbour, but if a prototype closer to the sample is selected in a following step, they will be discarded. However, if the approximation function is properly selected (as happens in many algorithms), the selected prototypes are very close to the sample in the final steps, so they could be used to improve the NN rule classification rates without the extra cost of finding exactly the k nearest neighbours.

The k *nearest selected neighbours* (k-NSN) rule is defined as follows: given an approximation-elimination based NN search algorithm, classify the unknown sample into the majority class of the k nearest prototypes selected by that algorithm as candidates to nearest neighbour, that is, those prototypes whose distance to the sample is computed during the NN search. The implementation of this rule is very simple: each time a distance is computed (line (a)), update the list containing the k minimum distances found so far, i.e, the distances to the k nearest selected neighbours, which is exactly the same that should be done to find exactly the k nearest neighbours; the difference is that in line (b) the distance used to prune in the k-NSN rule is d_{nn}, and that is why the k-NSN rule computes exactly the same distances as the NN rule. At the end of the search phase, the sample is classified into the majority class of the k-NSN.

The k-NSN rule requires little extra computational effort over a standard NN search (storing the k nearest selected neighbours), which is negligible in comparison with the cost of computing the extra distances required to find exactly the k-NN. The k-NSN rule fills the gap between the NN rule and the k-NN rule, but its performance depends highly on the underlying NN search algorithm.

4 Experiments and results

The idea behind the k-NSN rule was first applied to the LAESA algorithm [11], and then was extended to other algorithms (k-d tree, Fukunaga and Narendra's, vp-tree, GNAT, AESA and TLAESA) in [10]. This late work reports some preliminary results of the k-NSN rule with several synthetic and real-data tasks, which show that this rule yields similar classification rates than those of the k-NN rule, without extra time overhead.

In this work we present some results on synthetic data of increasing dimensions, varying from 10 to 50. The datasets used in the experiments have been obtained using the algorithm for generating clustered data in [7]. This algorithm generates points from a number c of Gaussian classes, with a fixed variance and maximum interclass overlap (the means are randomly chosen). For our experiments we have used 8 equally probable classes, with a variance of 0.01 and an overlap of 0.12. For each dimensionality, we have generated a set of points from which we have extracted 16 pairs of training and test sets of 4096 and 1024 prototypes respectively.

The error rates of the k-NSN rule (applied to several algorithms) in comparison with those of the k-NN rule, as the value of k increases, are plotted in figure 1 (the error bars plotted are those from the k-NN rule), for data of dimensions 10 and 20. In both cases the differences between the k-NSN error rates and the k-NN error rates are minimum, and become lower for dimension 20, which suggests that those differences should decrease as dimensionality increases. On another side, the results show that the fewer distances an algorithm compute, the worse its error rates are with the k-NSN rule (this is specially evident with the algorithms from the AESA family).

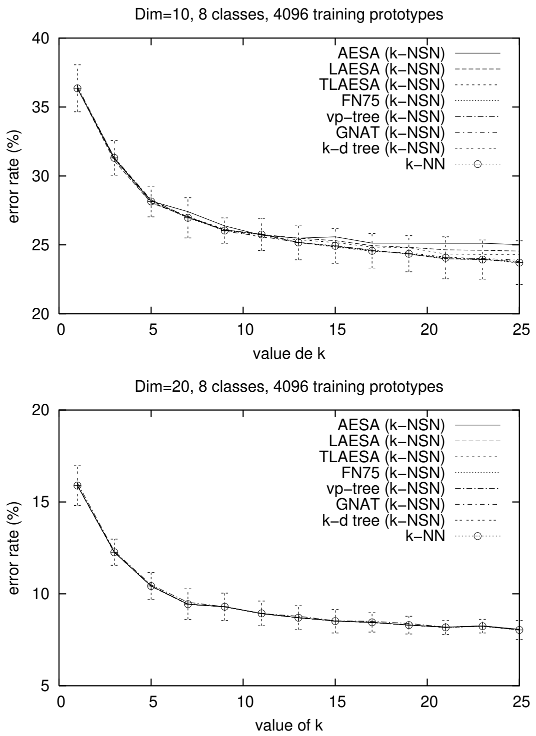

Figure 1. Error rates of the k-NSN rule for data of dimensions 10 (top) and 20 (bottom).

4.1 Relation between the k nearest neighbours and the k nearest selected neighbours

The results showed above seem to suggest that many of the nearest selected neighbours are in fact among the k nearest neighbours, or at least are not much farther than them from the sample (and thus should belong to the same class). To verify this hypothesis, we have developed some experiments in order to measure the percentage of k-NSN that are among the k-NN, and also to study the relation between the distances from the sample of the kth NSN ($d_{k-\text{NSN}}$) and the kth NN ($d_{k-\text{NN}}$), that is, the value of $d_{k-\text{NSN}}/d_{k-\text{NN}}$; this relation has to be always greater than or equal to 1, by definition. The experiments have been made with $k = 15$ and using several sizes for the training set, with 10 dimension data (see figure 2). The results show that both measures reach an stable state when the training set has a reasonable value, but do not improve (get close to 100 and 1, respectively) as the training set increases.

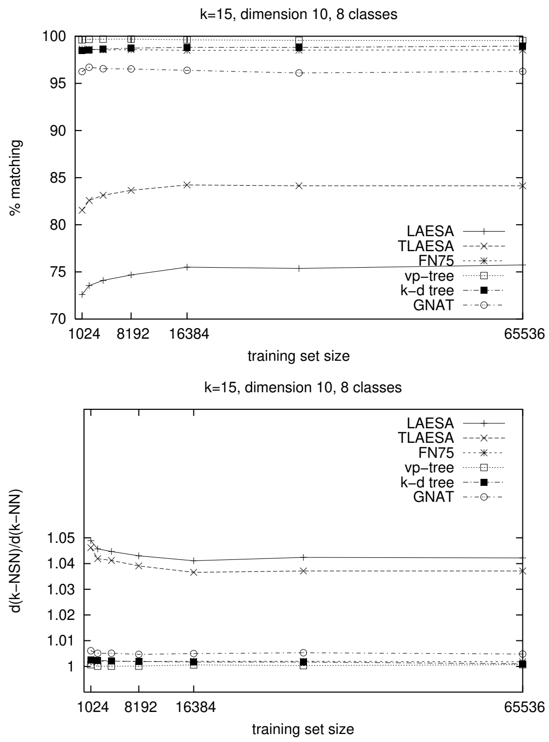

Figure 2. Percentage of the k-NSN among the k-NN (top), and relation between the distance of the last k-NSN and that of the last k-NN (bottom), for dimension 10.

The same experiment has been repeated with a fixed training set size of 4096, and with data of increasing dimensions (see figure 3). As these plots show, when the dimension becomes high the relation between the distances of the last k-NSN and the last k-NN reaches the limit value, 1, which indicates that the k-NSN are exactly at the same distance that the k-NN , that is, they may be different prototypes but are at the same distance to the sample (this usually happens when using different algorithms to find the k-NN, also).

5 Conclusions and future work

A new classification rule based on fast NN search algorithms has been developed and tested. This rule (the k-NSN rule) allows the classifier to obtain error rates very close to those of the k-NN rule with almost the same classification time as the NN rule, that is, without the extra computing time required by the k-NN rule. In this work we have also

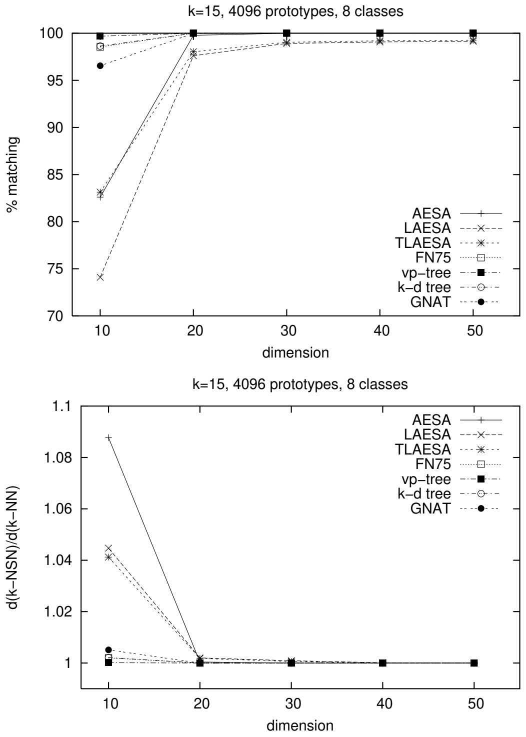

Figure 3. Percentage of the k-NSN among the k-NN (top), and relation between the distance of the last k-NSN and that of the last k-NN (bottom), as dimension increases.

analyzed the relation between the k nearest selected neighbours and the k nearest neighbours, showing that when the data dimension becomes high they match completely or at least are at the same distance from the sample. From this analysis we can conclude that many fast NN search algorithms do find the k nearest neighbours (for reasonable values of k) in their search for the nearest neighbour, and thus classification rates may be improved without significantly increasing classification time.

As for the future we plan to study the application of the k-NSN rule to other fast NN search algorithms and to real-data classification tasks; for these tasks we plan also to reproduce the analysis of the relation between the k nearest selected neighbours and the k nearest neighbours.

References

[1] S. Brin. Near neighbor search in large metric spaces. In *Proceedings of the* 21^{st} *VLDB Conference*, pages 574–584, 1995.

[2] R. Duda and P. Hart. *Pattern Recognition and Scene Analysis*. Wiley, 1973.

[3] J. Friedman, F. Baskett, and L. Shustek. An algorithm for finding nearest neighbors. *IEEE Transactions on Computers*, 24(10):1000–1006, 1975.

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

[5] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[6] K. Fukunaga and M. Narendra. A branch and bound algorithm for computing k–nearest neighbors. *IEEE Transactions on Computing*, 24:750–753, 1975.

[7] A. Jain and R. Dubes. *Algorithms for clustering data*. Prentice-Hall, 1988.

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

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

[10] F. Moreno-Seco, L. Micó, and J. Oncina. Extending fast nearest neighbour search algorithms for approximate k-nn classification. In *Proceedings of the 1st Iberian Conference on Pattern Recognition and Image Analysis (Port d'Andratx, Spain). Lecture Notes in Computer Science, F.J. Perales et al (Eds.) vol. 2652, Springer-Verlag.*, pages 589–597, Berlin, 2003.

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

[12] V. Ramasubramanian and K. K. Paliwal. Fast nearest-neighbor search algorithms based on approximation-elimination search. *Pattern Recognition*, 33:1497–1510, 2000.

[13] E. Vidal. An algorithm for finding the nearest neighbour in (approximately) constant time. *Pattern Recognition Letters*, 4:145–157, 1986.

[14] E. Vidal. New formulation and improvements of the nearest-neighbour approximating and eliminating search algorithm (AESA). *Pattern Recognition Letters*, 15:1–7, 1994.

[15] P. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, 1993.