A fast approximately k–nearest–neighbour search algorithm for classification tasks

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

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

Abstract. The k–nearest–neighbour (k–NN) search algorithm is widely used in pattern classification tasks. A large set of fast k–NN search algorithms have been developed in order to obtain lower error rates. Most of them are extensions of fast NN search algorithms where the condition of finding exactly the k nearest neighbours is imposed. All these algorithms calculate a number of distances that increases with k. Also, a vector–space representation is usually needed in these algorithms.
If the condition of finding exactly the k nearest neighbours is relaxed, further reductions on the number of distance computations can be obtained. In this work we propose a modification of the LAESA (Linear Approximating and Eliminating Search Algorithm, a fast NN search algorithm for metric spaces) in order to use a certain neighbourhood for lowering error rates and reduce the number of distance computations at the same time.
Keywords: Nearest Neighbour, Metric Spaces, Pattern Recognition.

* The authors wish to thank the Spanish CICyT for partial support of this work through project TIC97–0941.

1 Introduction

Non-parametric classification is one of the most widely used techniques in pattern recognition [2]. One of the simplest techniques (and one of the most popular) is to use the nearest–neighbour (NN) classifier which, given an unknown sample x, finds the prototype p in the training set which is closest to x, then it classifies x in the same class as p. The NN classifier usually obtains acceptable error rates, but it is possible to obtain better (lower) error rates using a number k of nearest neighbours. Thus, a k–nearest–neighbour (k–NN) classifier finds the k nearest neighbours of the sample x, and then, through a voting process, it classifies x in the class which has most representatives among those k nearest neighbours.

Usually, these classifiers are implemented through an exhaustive search; that is, all the distances between the sample and the prototypes in the training set are calculated. When the representation space is an Euclidean space, this exhaustive search is usually not very time–consuming. On the other hand, when working on a metric space[1] in which the temporal cost of calculating the distance between two prototypes is high (as for instance the edit distance when classifying handwritten characters [8]), the exhaustive search becomes impractical.

Several algorithms (AESA [7], LAESA [5] and TLAESA [4], among others) have been developed which find the nearest neighbour in a metric space with a low number of distance calculations. Also, the AESA algorithm has been extended [1] to find the k nearest neighbours with a low number of distance calculations. In this paper, we present an extension of the LAESA algorithm that uses at most k neighbours to classify the sample. Although this extension does not find exactly the k nearest neighbours, the error rates obtained are very close to those of a classifier that uses the exact k nearest neighbours.

In the next section we will introduce the LAESA algorithm and the proposed modification, the Approximating k–LAESA (Ak–LAESA) algorithm. The following sections describe the experiments and the results obtained, which show that the Ak–LAESA obtains error rates close to those of a k–NN classifier, while at the same time calculates a reduced number of distances. Finally we present the conclusions and we outline some future work.

2 The *Ak*–LAESA algorithm

The Ak–LAESA algorithm is derived from the LAESA algorithm [5, 6]. This latter algorithm relies on the triangle inequality to prune the training set. It uses a lower bound of the real distance between each prototype and the sample, which is used to eliminate all those prototypes that cannot be closer to the sample than a given one. The LAESA algorithm has two main steps:

1. Preprocessing step:
   This step is carried out before the classification begins. First, it selects a set

[1] A metric space is a representation space which has some kind of metric defined.

of a number of prototypes[2] called *base prototypes*. Then, it calculates and stores the actual distances between each base prototype and all the other prototypes in the training set (including the base prototypes).

2. Classification step:
   During this step, the distances calculated in the preprocessing step are used to obtain a lower bound of the distance between each prototype and the sample. Using this lower bound of the distance, the algorithm iteratively finds a good candidate to nearest neighbour, calculates the actual distance between this candidate and the sample and prunes the training set[3], until there is only one prototype: the nearest neighbour.

The *Ak*–LAESA (see figure 1) is a simple but powerful evolution of the LAESA algorithm that simply stops the search for the nearest neighbour when the number of remaining (not pruned) prototypes is less than a number k. Then, it classifies the sample by voting among those prototypes. Our experiments show that the number of prototypes used in the voting is in general smaller than k, and that those prototypes are not exactly the k nearest neighbours. Despite of this, error rates for this algorithm are very close (different in less than 1%) to those of a classifier using the exact k nearest neighbours. Also, the algorithm calculates a lower number of distances than a classifier using the k nearest neighbours, and also calculates fewer distances than the LAESA algorithm itself.

3 Experiments

In [6] we presented some results of the *Ak*–LAESA algorithm on synthetic data, using 2 classes. In this work, the algorithm for generating clustered data appeared in [3] has been used to generate synthetic data from 4 classes and with two values for the dimensionality of the generated data: 6 and 10. This algorithm generates random synthetic data from different classes (clusters) with a given maximum overlap between them. Each class follows a Gaussian distribution with the same variance and different randomly chosen means. The overlap was set to 0.04 and the variance to 0.05, in order to obtain low error rates (less than 8% for the NN classifier).

Once we had the synthetic data prepared, several partitions of training and test set with different sizes were made. For training sets, the sizes used were: 1024, 2048, 3072, 4096, 6144 and 8192. The test set size was 512 for all training set sizes. All experiments were repeated 16 times with different training and test sets of each size, in order to obtain statistically significative results.

In the first experiment we compare *Ak*–LAESA error rates with those of the NN classifier and various k-NN classifiers. The experiment has been repeated for $k = 7$ and $k = 17$, using data of different dimensionalities (see figure 2). As we

[2] This number depends exclusively on the dimensionality of data (see [5] for more details).

[3] All the prototypes whose lower bound of the distance is bigger than the actual distance between the candidate and the sample can be safely eliminated.

```
Input: P ⊂ E, n = |P|;                          { finite set of training prototypes }
   B ⊆ P, m = |B|;                              { set of Base prototypes }
   D ∈ ℝ^{n×m}; { precomputed n × m array of interprototype distances }
   x ∈ E;                                       {test sample }
   k ∈ ℕ;                                       { maximum number of neighbours to use }
Output: c ∈ ℕ;                                  { class assigned to the sample x }
Functions: d : E × E → ℝ;                       { distance function }
   VOTING : E → ℕ;                              { voting function }
Variables: p, q, s, b ∈ P;
   G ∈ ℝ^n;                                     { lower bounds array }
   p* ∈ E;                                      { best candidate to nearest neighbour }
   d*, d_xs, g_p, g_q, g_b ∈ ℝ;
   nc ∈ ℕ;                                      { number of computed distances }
   stop : Boolean;                              { used to stop the search }
begin
 d* := ∞; p* := indeterminate; G := [0];
 s := arbitrary_element(B); nc := 0; stop :=false;
 while |P| > 0 and not stop do
   d_xs := d(x, s); nc := nc + 1;               { distance computing }
   P := P − {s};
   if d_xs < d* then
      d* := d_xs; p* := s                       { updating d*, p* }
   endif
   b := indeterminate; g_b := ∞; q := indeterminate; g_q := ∞;
   for every p ∈ P do                           { eliminating and approximating loop }
    if s ∈ B then
       G[p] :=max(G[p], |D[p, s] − d_xs|);      { updating G, if possible }
    endif
    g_p := G[p];
    if p ∈ B then                               { approximating: selecting from B }
      if g_p < g_b then g_p := g_b; b := p endif
    else                                        { p ∉ B }
      if g_p >= d* then
         P := P − {p}                           { eliminating from P − B }
      else                                      { approximating: selecting from P − B }
         if g_p < g_q then g_p := g_q; q := p endif
      endif
    endif
   endfor
   if b ≠indeterminate then s := b
   elsif |P| < k then stop :=true               { stop the search }
   else s := q endif
 endwhile
 P := P ∪ {p*};                                 { retrieve the best candidate to NN }
 c :=VOTING(P);
end
```

Fig. 1. Algorithm *Ak*–LAESA

could expect, Ak–LAESA has lower error rates than the NN classifier (also lower than some k–NN classifiers), but they are slightly higher than those of the k–NN classifiers, using the same value of k. Another point is that the Ak–LAESA does not always use k prototypes to classify the sample: it uses *at most* k. This feature could explain the difference between Ak–LAESA error rates and k–NN classifiers error rates, for a fixed value of k. The average number of neighbours used by Ak–LAESA was calculated, in order to perform a more accurate comparison (see figure 6). The numbers were 5.27 for $k = 7$ when dimensionality was 6, and 6.78 prototypes with dimensionality 10. For $k = 17$, the numbers were 9.22 and 15.42, respectively. Figure 2 shows a comparison of the Ak–LAESA with a k–NN classifier that used aproximately the same number of prototypes (6, 7, 10 and 16 respectively), and also with a k–NN classifier with $k = 7$ and $k = 17$. The error rates of a NN and a 3–NN classifiers also have been plotted.

Even though Ak–LAESA error rates are slightly higher than those of the k–NN classifiers (but lower than those of a NN classifier), our experiments show that the number of distance calculations of the Ak–LAESA does not depend on the size of the training set (see figure 3), while the number of distance calculations of a k–NN classifier (when using the exhaustive search) is exactly the size of the training set. When the temporal cost of calculating the distance between two prototypes is high, the Ak–LAESA is faster than the k–NN classifier, and obtains error rates very close to those of the k–NN classifier.

The second experiment was performed to show that the Ak–LAESA calculates less distances than the LAESA algorithm. Also, as it happens with the LAESA algorithm, the number of distance calculations does not depend on the training set size. Figure 3 shows the average number of distance calculations of the LAESA algorithm, and the Ak–LAESA with $k = 3$ and $k = 7$.

The aim of the third experiment was to show the behaviour of Ak–LAESA when the value of k increases, and a comparison with the behaviour of a k–NN classifier was also made. As shown in figure 4, the error rate starts (with dimensionality 6) at a value close to 7% in both classifiers. As the value of k increases, the rate tends to a 4% for the k–NN classifier and to a 5% for Ak–LAESA. Figure 5 plots the average difference between Ak–LAESA error rates and the k–NN classifier error rates, showing that the k–NN classifier error rates are lower, but the difference is (with dimensionality 6 and 10) less than 1%.

Finally, an experiment was developed to find out how many prototypes used by the Ak–LAESA algorithm to classify were actually among the k nearest neighbours (see figure 6). The tables in figure 6 show, for two different values of k:

1. The average number of prototypes used by the Ak–LAESA in the voting process,
2. How many of these prototypes are among the k nearest neighbours (and also the percentage), that is, how many of the final prototypes used by the Ak–LAESA are one of the k nearest neighbours.
3. How many are among the $2k$ nearest neighbours, and
4. How many are among the $4k$ nearest neighbours.

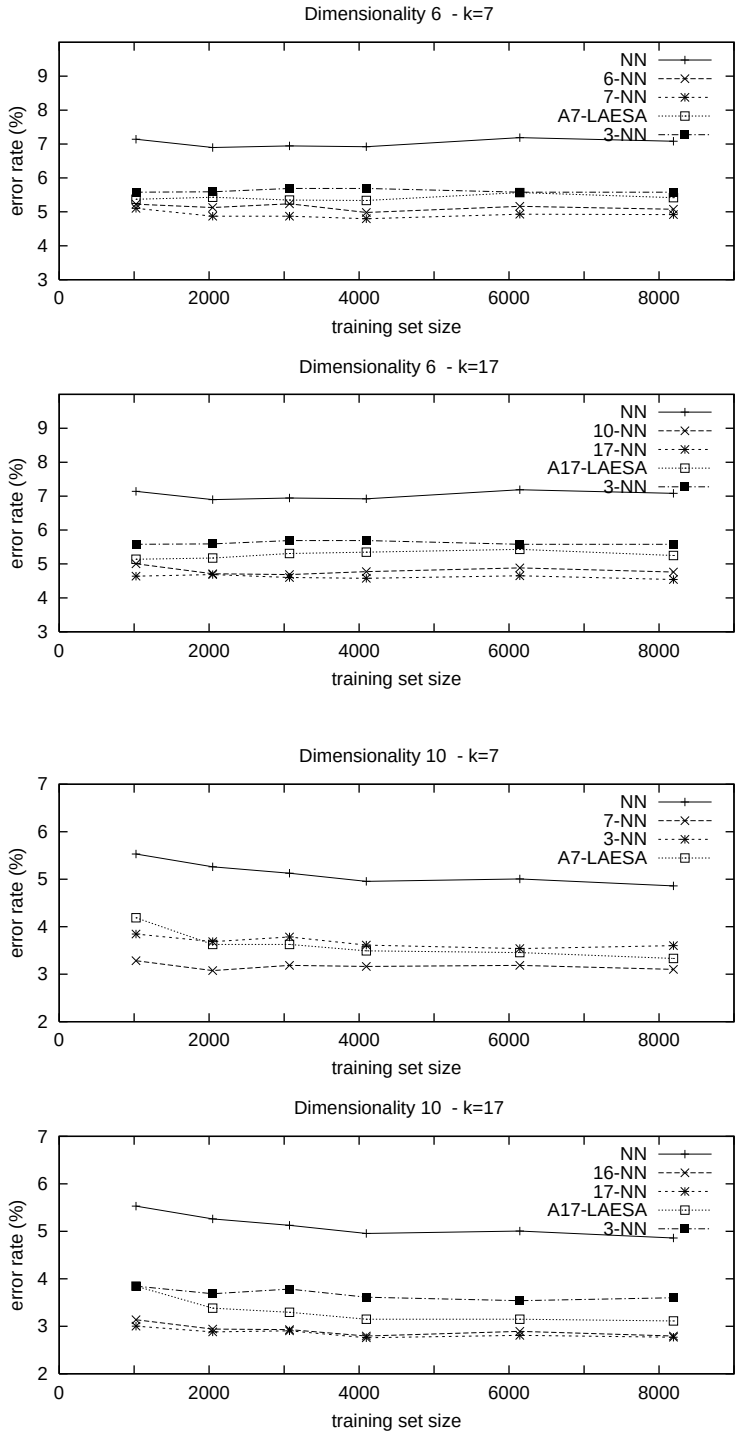

Fig. 2. Error rates of the Ak–LAESA classifier compared to k–NN classifiers when the training set size increases.

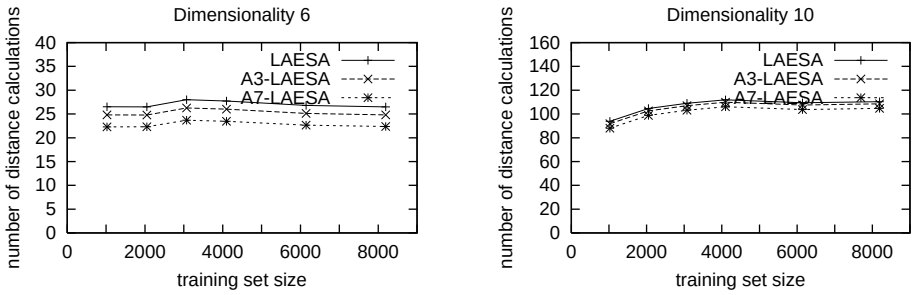

Fig. 3. Average number of distance calculations of the LAESA and Ak–LAESA algorithms, as the training set size increases.

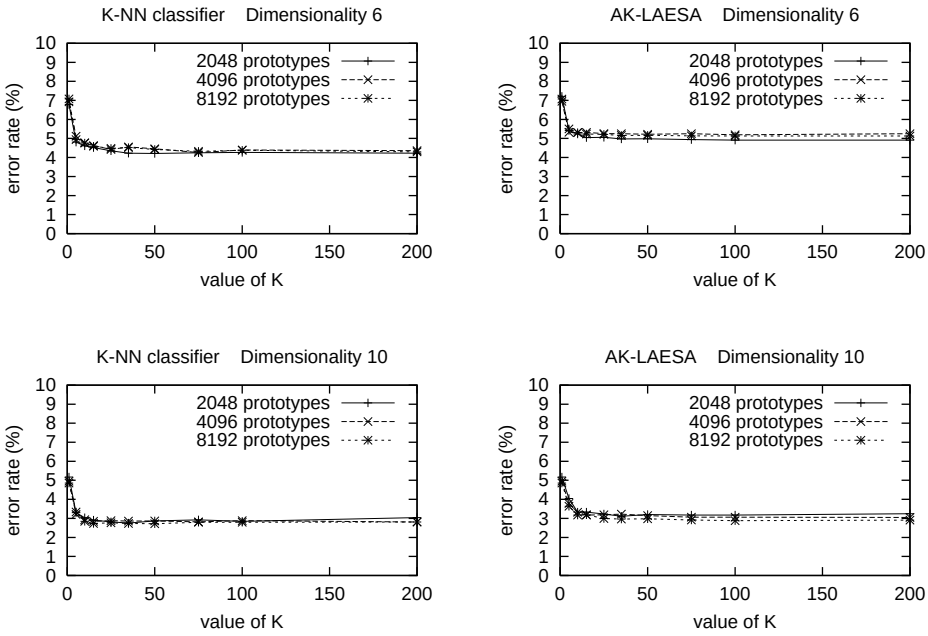

Fig. 4. Error rates of the Ak–LAESA classifier compared to k–NN classifier, as the value of k increases.

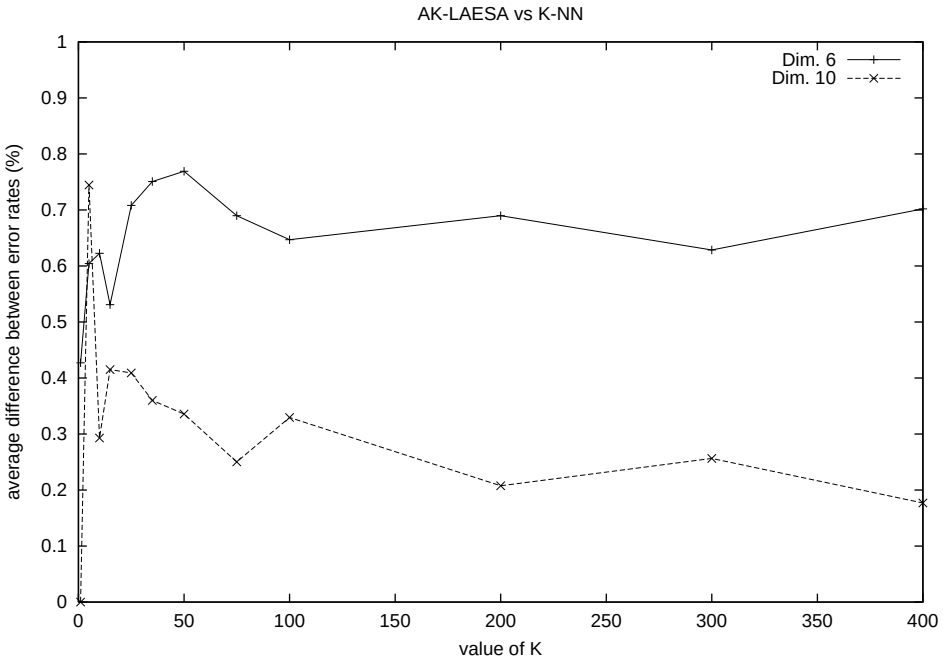

Fig. 5. Average difference between error rates of the Ak–LAESA classifier a k–NN classifier, as the value of k increases.

The results show that the number of prototypes used by the Ak–LAESA which are among the k nearest neighbours decreases as dimensionality increases, while at the same time Ak–LAESA error rates improve (see figure 4).

Dimensionality 6				
Value of k	Prototypes Used	Among k NN	Among $2k$ NN	Among $4k$ NN
7	5.27	2.33 (44%)	3.04 (58%)	3.74 (71%)
17	9.22	5.14 (56%)	6.49 (70%)	7.61 (83%)

Dimensionality 10				
Value of k	Prototypes Used	Among k NN	Among $2k$ NN	Among $4k$ NN
7	6.78	1.42 (21%)	1.86 (27%)	2.56 (38%)
17	15.42	3.61 (23%)	5.57 (36%)	8.15 (53%)

Fig. 6. Prototypes used by the Ak–LAESA which are among the k nearest neighbours.

4 Conclusions and future work

We have developed a fast classifier based on the LAESA algorithm [5] which obtains error rates close to those of a k–NN classifier, while calculating a low number of distances. Also the temporal and spatial complexities of the Ak–LAESA algorithm are the same than those of the LAESA algorithm. The Ak–LAESA error rates (and its behaviour as k increases) are very close to those of a classifier that uses the k nearest neighbours. The Ak–LAESA performance seem not to decrease as the size of the training set grows. Also, this behaviour of the Ak–LAESA seems to improve as dimensionality of data increases.

As for the future, we will explore the behaviour of the Ak–LAESA with data of dimensionalities higher than those used for these experiments. Also, we plan to apply this algorithm to real data tasks.

Currently, the base prototypes selection algorithm of the Ak–LAESA has been borrowed from the LAESA algorithm, and we have also used the LAESA's optimal number of base prototypes for each value of the dimensionality. We think that a different number of base prototypes or a different selection algorithm for base prototypes can improve the error rates of the Ak–LAESA, specially with low dimensionality data.

Acknowledgements: The authors wish to thank Jorge Calera-Rubio and Mikel L. Forcada, for their invaluable help in writing this paper.

References

1. Aibar, P., Juan, A., Vidal, E.: Extensions to the approximating and eliminating search algorithm (AESA) for finding k-nearest-neighbours. New Advances and Trends in Speech Recognition and Coding (1993) 23–28
2. Duda, R., Hart, P.: Pattern Classification and Scene Analysis. Wiley (1973)
3. Jain, A.K., Dubes, R.C.: Algorithms for clustering data. Prentice-Hall (1988)
4. 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
5. 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
6. Moreno-Seco, F., Oncina, J., Micó, L.: Improving the LAESA algorithm error rates. In: Proceedings of the VIII Symposium Nacional de Reconocimiento de Formas y Análisis de Imágenes, Bilbao (1999) 413–419
7. Vidal, E.: New formulation and improvements of the Nearest-Neighbour Approximating and Eliminating Search Algorithm (AESA). Pattern Recognition Letters (1994) **15** 1–7
8. Wagner, R.A., Fischer, M.J.: The String-to-String Correction Problem. Journal of the Association for Computing Machinery (1974) **21**(1) 168–173