Extending LAESA Fast Nearest Neighbour Algorithm to Find the *k* Nearest Neighbours

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. Many pattern recognition tasks make use of the k nearest neighbour (k–NN) technique. In this paper we are interested on fast k–NN search algorithms that can work in any metric space i.e. they are not restricted to Euclidean–like distance functions. Only symmetric and triangle inequality properties are required for the distance.
A large set of such fast k–NN search algorithms have been developed during last years for the special case where $k = 1$. Some of them have been extended for the general case. This paper proposes an extension of LAESA (Linear Approximation Elimination Search Algorithm) to find the k-NN.

1 Introduction

The k nearest neighbour problem consists in finding the k nearest points (prototypes) from a database to a given point sample using a dissimilarity function $d(\cdot, \cdot)$. This problem appears often in computing problems and, of course, in pattern recognition tasks [2].

Usually, a brute force approach is used but, when the database is large and/or the dissimilarity function is computationally expensive, this approach results in a real bottleneck.

In this paper we are interested in fast k-NN algorithms that can work in any metric space *i.e.* the algorithm is not restricted to work with Euclidean–like dissimilarity functions, and no assumption is made about the point's data structure. It is only required that the dissimilarity function fulfils the following conditions:

– $d(x, y) = 0 \Leftrightarrow x = y$.
– $d(x, y) = d(y, x)$ (symmetry).
– $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality).

That is, the dissimilarity function defines a *metric space*, and thus can be properly called a *distance* function.

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

© Springer-Verlag Berlin Heidelberg 2002

Such algorithms can efficiently find the k–NN when the points are represented by structures like strings, trees or graphs and the distance functions can be some variants of the edit distance ([9], [10]).

Many general metric space k–NN fast search algorithms have been developed trough these years for the special case where $k = 1$ (Fukunaga and Narendra's [3], Kalantary and McDonald's [5], AESA [8], LAESA [7], TLAESA [6], ...). One of these algorithms has been extended for the general case (k-AESA [1]).

This paper proposes an extension of LAESA (Linear Approximation Elimination Search Algorithm) fast 1–NN search algorithm to cope with the k–NN problem.

2 The LAESA

As an evolution of AESA algorithm, LAESA is a branch and bound algorithm based on the triangle inequality.

In a preprocessing step a set of n_{bp} *base prototypes* are selected and their distances to the rest of prototypes are stored in a table.

When searching the nearest neighbour of a sample s, first a lower bound ($g[\cdot]$) of the distance from each prototype (p) to the sample is computed. This lower bound distance is based on the triangle inequality and can be computed as follows:

$$g[p] = MAX_{i=1}^{n_{bp}}(|d(b_i, p) - d(s, b_i)|) \tag{1}$$

where $d(b_i, p)$ is the precomputed distance between p and the base prototype b_i, and $d(s, b_i)$ is the actual distance between the sample s and b_i.

After that, the prototype set is traversed in ascending order of $g[\cdot]$ until a prototype with a distance to the sample lower than the lower bound distance is found. Then the traversal stops and the nearest prototype to the sample found so far is outputed as the nearest neighbour.

The performance of the algorithm depends on the number of base prototype and the way they are selected. In [7] it was shown that selecting the prototypes so that they are maximally separated is a good choice. A pseudo-algorithmic description of LAESA is shown in figure 1.

LAESA was devised to work with very time consuming distances, then, in practice, the time cost of the algorithm is dominated by the number of distance computations (n_d). In [7] it was shown that in most usual cases n_{bp} is practically independent of the database size (n) and can be set to a number much smaller that n. Nevertheless, worst-case time complexity can be expressed as $O(n + n_d \log n)$, but since n_d in practice does not grow with n, the expended time grows linearly with n. Please note that n_d includes the distances to base prototypes, so n_d is always bigger than n_{bp}. The space complexity of LAESA can be expressed as $O(n_{bp}n)$.

```
Preprocessing (given the number $n_{bp}$ of base prototypes)
  1. Select the $n_{bp}$ base prototypes $B$ maximally separated
  2. Compute and store the distances $d(b,p)$ $\forall b \in B$ $\forall p \in P$

Classification (input: the sample $s$; output: the nearest neighbour of $s$, $p_{\min}$)
  1. compute and store the distances $d(b,s)$ $\forall b \in B$
  2. $p_{\min} = \mathrm{argmin}_{b \in B} d(b,s)$ and compute the lower bound $g[p]$ $\forall p \in P$
  3. for all $p$ in $P$ in ascending order of $g[p]$
     (a) if $g[p] > d(p_{\min}, s)$ stop the algorithm
     (b) compute $d(p,s)$; if $d(p,s) < d(p_{\min}, s)$ then $p_{\min} = p$
```

Fig. 1. The LAESA

3 Extending LAESA to *k*–LAESA

Instead of stopping the algorithm when the lower bound distance of the current prototype is bigger than its distance to the sample (line 3a), the algorithm is stopped when the lower bound distance is bigger that the kth nearest neighbour found so far.

K–LAESA must store the k nearest neighbours found up to the moment. As k is lower than n the space and time complexity does not change.

As can be expected, our experiments show that the number of distance computations increases as the value of k increases. Despite of this, the total number of distance computations remains much lower than exhaustive k–NN search, thus k–LAESA can be very useful when distance computations are very expensive.

4 Experiments

K–LAESA is intended for tasks where a very time consuming distance is required. The actual bottleneck in these tasks is the number of distance computations (n_d). Thus, the experiments reported here are focused exclusively on the number of distance computations for several tasks.

In a first set of experiments, 4 and 10 dimension spaces along with the Euclidean distance were used. Of course, there are some specially designed fast k–NN algorithms for such spaces that can beat LAESA and k–LAESA (remind that LAESA was devised for very time consuming distances). Those experiments are included just to show the behaviour of LAESA in a well known metric space.

Second, some experiments with misspelled words and the edit distance were performed to show k–LAESA's behaviour in its application field.

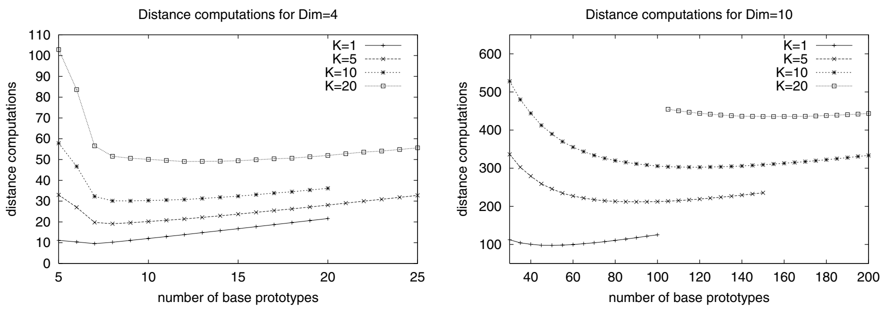

Fig. 2. Searching for optimal *number of base prototypes* in uniform distributions

Table 1. Optimal values of n_{bp} for uniform distribution data

VALUE	Dimensionality	
OF k	4	10
1	7	50
5	8	90
10	9	120
20	12	155

4.1 Experiments with the Euclidean Distance

For these experiments the well-known uniform distribution on the 4 and 10 dimension unit hipercube has been chosen as a reference.

First the optimal number of base prototypes has to be found. Then the evolution of the number of distance computations (n_d), for different values of k (1, 5, 10, 20), is studied when the number of prototypes grows.

As shown in figure 2, the optimal values of n_{bp} depends on the dimension and on the value of k. Those values (table 1) are used on the following experiments.

Next, the number of distance computations was studied as the database grows (1024, 2048, 4096, 6144 and 8192 prototypes). The test set was a collection of 1024 samples. For each database size, the experiment was repeated for 16 different pairs train/test set in order to obtain sounder results. In figure 3 it can be observed that n_d grows very slightly with respect to the database size, but, as figure 4 shows, n_d grows as the value of k increases (only the results for 2048 prototype database is plotted).

4.2 Experiments with the Edit Distance

In these experiments a dictionary of more than 60000 words was used. To obtain test words, one edition mistake (insertion, deletion or substitution) was equiprobably introduced in each word. Only results for $k = 1$ and $k = 5$ will be reported here.

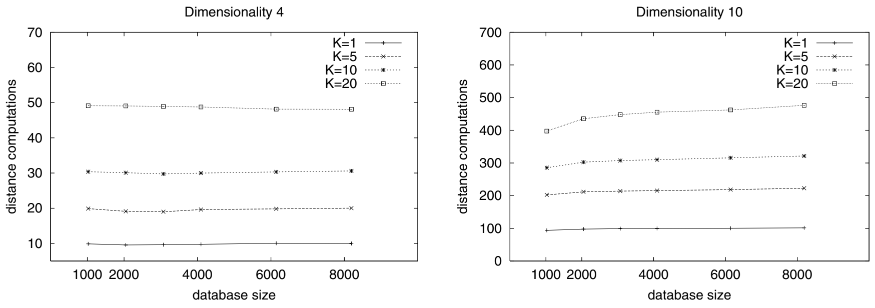

Fig. 3. Distance computations for uniform distributions

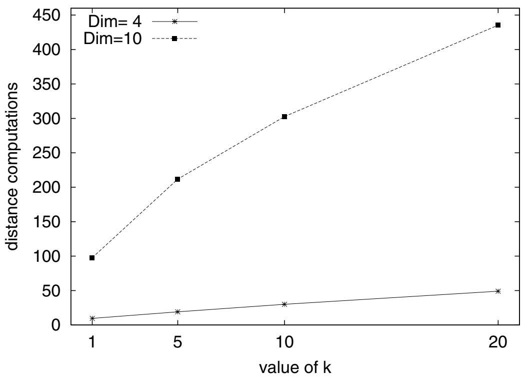

Fig. 4. Distance computations as k increases (uniform distributions)

As in previous experiments, exhaustive experiments were developed to obtain the optimal value of n_{bp} for each value of k; the results are plotted in figure 5. The optimal values obtained were 102 base prototypes for $k = 1$ and 510 base prototypes for $k = 5$. Then, experiments with databases of increasing sizes and 1000 samples were performed. The number of distance computations (n_d) obtained in these experiments is plotted in figure 6, which confirms that the increasing in n_d depends more on the value of k than on the size of the database.

5 Conclusions

We have developed an extension of LAESA to find the k nearest neighbours. This new algorithm (k–LAESA) is intended for tasks where the distance computation is very time consuming. No special data structure is required for points, only the distance is required to fulfill the symetric and the triangle inequality properties.

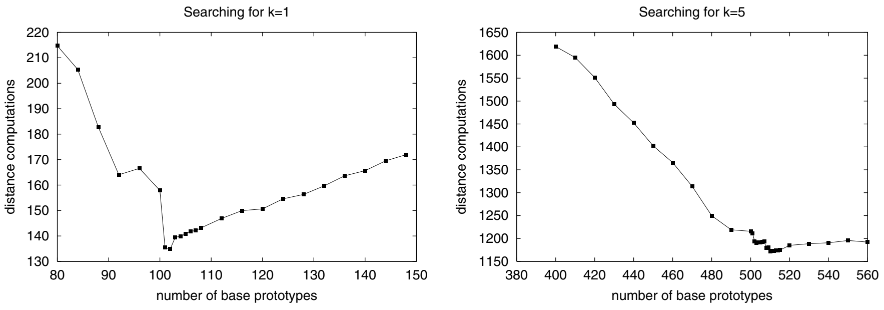

Fig. 5. Searching optimal n_{bp} for distorted words

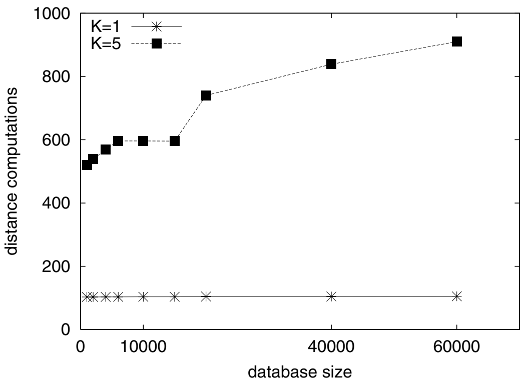

Fig. 6. Distance computations for distorted words

The experiments reported in this work show that the number of distance computations grows with the value of k, but remains always much lower than k–NN exhaustive search. This number of distance computations seems to grow very slowly with the database size. Also, the space required by this algorithm is almost linear with the database size. K–LAESA is good alternative when the distance computation is very time consuming and the database is large.

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) 23–28 719
2. Duda, R., Hart, P.: Pattern Classification and Scene Analysis. Wiley (1973) 718
3. Fukunaga, K., Narendra, M.: A branch and bound algorithm for computing k–nearest neighbors. IEEE Trans. Computing (1975) **24** 750–753 719
4. Jain, A. K., Dubes, R. C.: Algorithms for clustering data. Prentice-Hall (1988)

5. Kalantari, I., McDonald, G.: A data structure and an algorithm for the nearest point problem. IEEE Trans. Software Engineering (1983) **9** 631–634 719
6. Micó, L., Oncina, J., Carrasco, R. C.: A fast branch and bound nearest neighbour classifier in metric spaces. Pattern Recognition Letters (1996) **17** 731–739 719
7. 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** 9–17 719
8. Vidal, E.: New formulation and improvements of the Nearest-Neighbour Approximating and Eliminating Search Algorithm (AESA). Pattern Recognition Letters (1994) **15** 1–7 719
9. Wagner, R. A., Fischer, M. J.: The String-to-String Correction Problem. Journal of the Association for Computing Machinery (1974) **21**(1) 168–173 719
10. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal of Computing (1989) **18** 1245–1262 719