Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

Pattern Recognition Letters 24 (2003) 47–53

Pattern Recognition Letters

www.elsevier.com/locate/patrec

A modification of the LAESA algorithm for approximated k-NN classification

Francisco Moreno-Seco [*], Luisa Micó, Jose Oncina

Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante, E-03071 Alicante, Spain

Received 9 April 2001; received in revised form 27 March 2002

Abstract

Nearest-neighbour (NN) and k-nearest-neighbours (k-NN) techniques are widely used in many pattern recognition classification tasks. The linear approximating and eliminating search algorithm (LAESA) is a fast NN algorithm which does not assume that the prototypes are defined in a vector space; it only makes use of some of the distance properties (mainly the triangle inequality) in order to avoid distance computations.

In this work we propose an improvement of LAESA that uses k neighbours in order to approach to the accuracy of a k-NN classifier, and computes the same number of distances than the LAESA preserving the time and space complexity independent from k.

Keywords: Nearest neighbour; Metric spaces; Non-parametric classification

1. Introduction

Nearest-neighbour (NN) classification is one of the most widely used non-parametric techniques in pattern recognition (Duda and Hart, 1973). Given an unknown sample x, this technique finds the prototype p in the training set P which is closest to x, then it classifies x in the same class as p. Sometimes it is possible to reduce error rates using a number k of NNs instead of only one. Thus, a k-nearest-neighbour (k-NN) classifier finds the k-NNs of the sample x, and then, through a voting process, classifies x in the class which has most representatives among those k-NNs.

A simple implementation of those classifiers consists on an exhaustive search, computing all the distances between the sample and each prototype in the training set. When the distance computation is time expensive and/or the training set is large this technique can be impractical.

There is a wide number of fast NN and k-NN search algorithms. Most of them require the prototypes to be defined in a vector space (such as for instance kd-tree (Bentley, 1975)). In this work we are interested on algorithms that can work with any distance properly defined. That is, the distance

[*] Corresponding author. Tel.: +3-4965-903-400; fax: +3-4965-903-434.

E-mail addresses: paco@dlsi.ua.es (F. Moreno-Seco), mico@dlsi.ua.es (L. Micó), oncina@dlsi.ua.es (J. Oncina).

function $d(\cdot,\cdot)$ has to fulfill the following properties:

(1) $d(x,x) = 0$
(2) $d(x,y) = d(y,x)$
(3) $d(x,y) \leqslant d(x,z) + d(z,y)$ (triangle inequality)

Note that we are not assuming the points to have any particular structure; therefore, the prototypes can be represented by strings, graphs, tables or any other data structure.

Several algorithms such as Fukunaga and Narendra's (1975), Kalantari and McDonald's (1983), approximating and eliminating search algorithm, AESA (Vidal, 1986), linear approximating and eliminating search algorithm, LAESA (Micó et al., 1994), TLAESA (Micó et al., 1996) and RCNN (Lee and Chae, 1998) [1] among others have been developed in order to find the NN in metric spaces using a low number of distance computations (see Ramasubramanian and Paliwal, 2000 for a comparison with some of those algorithms).

AESA makes use of a table of the distances from each prototype to all the other prototypes. This makes the space complexity quadratic and the algorithm unusable in many practical situations. LAESA was introduced to avoid this need. It has a linear space complexity but increases slightly the number of distance computations (n_d). This makes the algorithm very useful when the distance computation is very time consuming and the training set is large.

The AESA was extended to find the k-NN (Aibar et al., 1993). A similar modification can be used in LAESA to obtain the k-LAESA. The problem on these versions is that the number of distances grows quickly with k.

In this paper, we present an improvement of the LAESA to use k neighbours to classify the sample in order to approximate the error rate of a k-NN classifier, while maintaining the LAESA number of distances.

This extension retains the main properties of LAESA:

- it is suitable for any metric space, e.g. it does not require a vector-space of representation,
- it has $O(n + n_d \log n)$ worst-case time and $O(n)$ space complexities with respect to the training set size n and the number of distance computations n_d, and
- it calculates the same number of distances than LAESA.

In the next section the LAESA algorithm is described along with the extension presented in this paper, the Approximating k-LAESA (Ak-LAESA). The following section describes the experiments and in the last section the conclusions are given and an outline of some future work is described.

2. The LAESA and Ak-LAESA algorithms

2.1. The LAESA algorithm

In order to avoid distance computations LAESA, [2] in a preprocessing step, computes and stores the distance from a subset of the prototypes (called *base prototypes*) B to each prototype in the training set P. Those distances are used to compute a lower bound function $g(\cdot,\cdot)$ of the distance from each prototype p to the sample x.

Let us suppose that the distance from a base prototype b to the sample is known. Then, applying the triangle inequality, $|d(p,b) - d(b,x)|$ is a lower bound of $d(p,x)$ (see Fig. 1). This bound can be generalised by taking the maximum over all the base prototypes. Then the lower bound function $g(\cdot,\cdot)$ can be defined as:

$$g(p,x) = \max_{b\in B} |d(p,b) - d(b,x)|, \tag{1}$$

[1] Although this algorithm is presented using vector prototypes, it does not require explicitly a vector space representation (it only makes use of inter-prototype distances), so it could also be used with more general metric spaces.

[2] There are some versions of the LAESA, here the simplest one, known as the EC1-LAESA, is used.

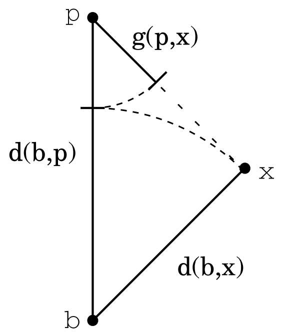


Fig. 1. The lower bound $g(p,x)$ of the distance from p to x is computed as $|d(p,b) - d(b,x)|$. The distance $d(b,p)$ is precomputed during the preprocessing step.

Note that the distances $d(b,x)\ \forall b \in B$ have to be computed before the use of the lower bound. In the LAESA, the lower bound is also used to find a candidate NN. If the lower bound is bigger than the distance from the candidate prototype to the sample, then this prototype is eliminated and a distance computation is avoided.

Obviously, this lower bound is only useful if its computation is cheaper than the distance computation times the number of the avoided distance computations. As the number of base prototypes is fixed before classification starts, the time complexity of the lower bound computation is $O(n)$, so this computation does not increase the time complexity of the algorithm.

The only remaining question is how many and which base prototypes to choose. The number of base prototypes has to be adjusted through some preliminary experiments. It has been shown that it does not depend much on the training set size but rather on the intrinsic dimensionality of the data. The best criterium so far to select the base prototypes is to choose them so that they are maximally separated. [3] The algorithm from Fig. 2 shows how to select n_{bp} base prototypes so that they are maximally separated.

The LAESA algorithm, which combines all of these ideas, can be described as in Fig. 3.

[3] In Micó et al. (1994) a greedy algorithm is proposed for selection of base prototypes *approximately* maximally separated in $O(n)$, with similar classification results.

```
select an arbitrary prototype $p$ as the first base prototype $b_1$
$B = \{ b_1 \}$
while $|B| < n_{bp}$ do
        for all $p$ in $P - B$ do
            $A[p] = 0$
            for all $b_i$ in $B$
                $A[p] = A[p] + d(b_i, p)$
            endfor
        endfor
        find $b'$ from $P - B$ such that $A[b']$ is the maximum
        $B = B \cup \{ b' \}$
endwhile
```

Fig. 2. Selection of base prototypes in the LAESA algorithm.

The $O(n + n_d \log n)$ worst-case time complexity comes from the line 3 of the algorithm, which can be implemented creating a heap ($O(n)$) and then extracting n_d elements from it ($n_d \log n$). As experiments in Micó et al. (1994) show, n_d is usually much lower than n. In practice, the time expended in this step is not actually a bottleneck compared to the time expended in distance computations or lower bound computations, so line 3 can also be implemented by sorting the array of lower bounds $O((n \log n))$ instead of implementing a heap.

2.2. The A*k*-LAESA algorithm

A*k*-LAESA is a simple but powerful modification of the LAESA algorithm. The aim of this new algorithm is to achieve classification rates similar to those of a *k*-NN classifier, using *k* neighbours that may not be the *k*-NNs and preserving the main properties of LAESA (distance computations and time and space complexities).

In LAESA each time a distance is computed its value is used to update the NN candidate. In A*k*-LAESA all the values of the distance computations in line 3b are stored, and simply the best *k* prototypes according to these stored distances are chosen to vote the class of the sample. These *k* prototypes are not necessarily the *k* nearest ones because the algorithm is stopping (line 3a) when the lower bound of the remaining prototypes is bigger than the actual distance to the nearest prototype instead of the *k*th one. Please note that A*k*-LAESA computes the same number of distances than LAESA, but makes use of the closest *k* prototypes whose distance has been computed to

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

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

Fig. 3. The LAESA algorithm.

classify the sample, instead of using only the NN. The next section shows that the Ak-LAESA produces similar classification results than using the k-NN.

3. Experiments

In order to test the algorithm two different tasks were performed. The first task involved synthetic clustered data in a Euclidean space. Of course, fast NN algorithms specialised in Euclidean distance can beat Ak-LAESA in this task. The task is included just to show the soundness of Ak-LAESA in a well-known task where the dimensionality and other characteristics of the data are under control.

The second task is devoted to test Ak-LAESA in a real data situation. In this task contour chains of handwritten digits are used as points and the edit distance is used for comparison (no Euclidean distance is possible in this case).

3.1. Synthetic data experiments

Using synthetic data, two experiments have been performed. The first one was made to compare the classification power of Ak-LAESA with respect to k-NN classifiers. The second set of experiments studies the evolution of Ak-LAESA error rates as the value of k increases, and compares it with k-NN evolution.

All the experiments used synthetic data obtained using the Jain and Dubes (1988) algorithm. Given n, d, σ^2, $n_{\min}$, I_o and c, this algorithm produces n points in a d-dimensional unit hypercube arranged in c spherically shaped Gaussian classes with a variance σ^2 having an overlap between pairs of classes lower than I_o (Bayes error). Each class has a minimum of $n_{\min}$ elements.

The parameters I_o and σ^2 were set to 0.04 and 0.05 respectively in order to obtain error rates lower than 10% with a NN classifier and $n_{\min}$ was set to n/c (all classes have equal size). All the experiments were repeated for different dimensionalities (6, 10 and 14) with similar results, but only results for dimensionality 10 have been presented in order to avoid redundancy. The algorithm seems to perform better as dimensionality grows, but this behaviour may be due to the use of Gaussian distributions in these experiments.

All the experiments were repeated 16 times for each pair of training and test set sizes. While the size of the training set was variable, the size of the test set was always 1024. The average and the standard deviation are shown in the plots. All the experiments were repeated for data from several number of classes, but only results for 4 and 8 classes will be reported here.

The Ak-LAESA has two parameters: the value of k and the number of base prototypes (see Section 2). In the first set of experiments the number of base prototypes was borrowed from previous works. In Micó et al. (1994) it was found that for a 10-dimensional Euclidean space the optimum is 48 base prototypes.

In this set of experiments $k = 11$ was chosen. The following set of experiments study how the parameter k affects the error rate.

The first set of experiments were developed to compare the error rates of A11-LAESA with NN and 11-NN, with increasing training set sizes of 256, 512, 768, 1024, 1536, 2048, 3072 and 4096. As

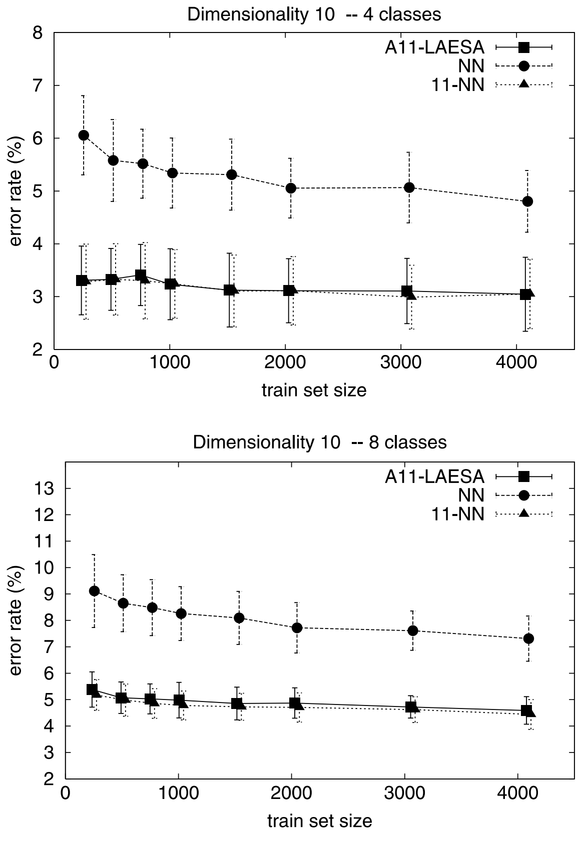


Fig. 4. Error rates of the A11-LAESA classifier using 4 and 8 classes compared to NN and 11-NN classifiers when the size of the training set increases.

Fig. 4 shows, the error rate of A11-LAESA is always very close to the rate of a 11-NN classifier. The error rate of a NN classifier has been also plotted as a reference. Fig. 5 plots the average number of distance computations of A11-LAESA for data from 4 and 8 classes.

From those plots it can be concluded that the Ak-LAESA has a very similar classification power than the k-NN computing drastically less distances ($\approx$2.5% of the total and decreasing as the training set increases).

The second set of experiments was designed to study the behaviour of Ak-LAESA as the value of k increases, and to compare this behaviour with k-NN. These experiments used sets of 2048 prototypes for training, and sets of 1024 prototypes for test, as in the previous experiment. The range

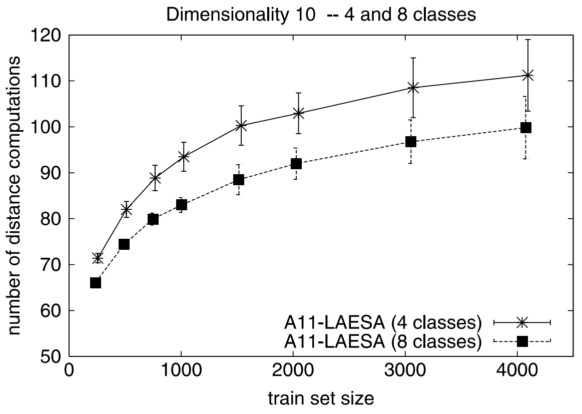


Fig. 5. Distance computations of Ak-LAESA for 4 and 8 classes.

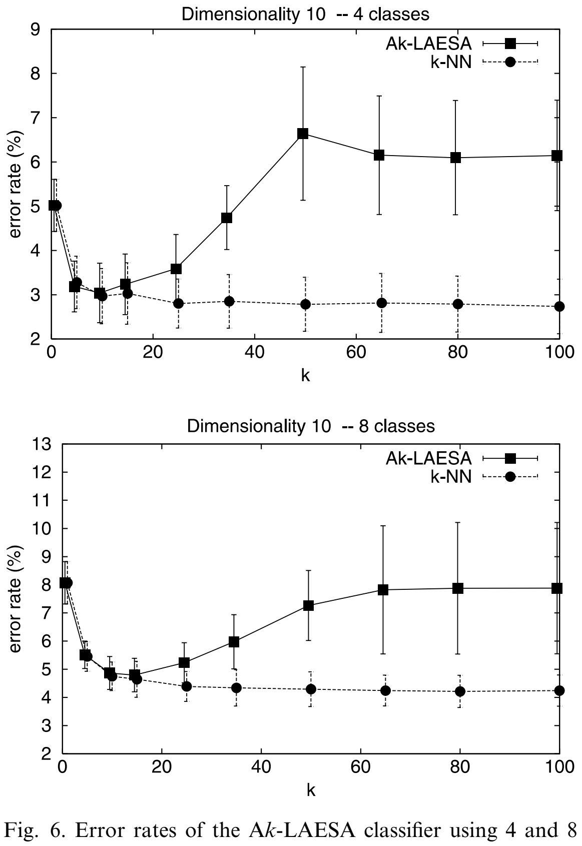


Fig. 6. Error rates of the Ak-LAESA classifier using 4 and 8 classes compared to the k-NN classifier when k increases.

of values for k was 1–100. As shown in Fig. 6, the behaviour of Ak-LAESA as the value of k increases is similar to that of k-NN for small values of k.

When the value of k becomes greater than approximately 20, the error rate of Ak-LAESA increases. This is probably due to the fact that, for 2048 prototypes, around 40 distances to non-base prototypes are computed. From these 40 non-base prototypes, k have to be selected in order to vote the class of the sample. If k is large enough the worst prototypes will enter in the voting pool. Further studies have to be done in this subject; until now one has to be careful choosing k.

3.2. Real data experiments

These experiments are a repetition of the experiments performed in (Micó and Oncina, 1998) in order to show the behaviour of LAESA. These experiments have been performed using handwritten digits from the NIST Special Database 3. Each 128 × 128 image was first reduced to a 64 × 64 image and then the contour of each image was codified as an 8-direction string. The edit distance, using a substitution weight proportional to the difference angle of the directions has been used to classify the samples.

In the same way than in the original work the *looseness* technique is also used in order to reduce the number of distance computations. Intuitively, the looseness H is a small value which is subtracted from $d(p_{\min}, x)$ in order to allow less distances to be computed. Then the comparison in line 3a of the LAESA algorithm:

if $g(p,x) > d(p_{\min}, x)$ stop the algorithm

has been changed to

if $g(p,x) > d(p_{\min}, x) - H$ stop the algorithm

In Micó and Oncina's (1998) work was determined that the optimal looseness for this problem is 0.06 and that the optimal number of base prototypes is 40.

Two different experiments have been performed: first, we have made a comparison of Ak-LAESA, with ($H = 0.06$) and without looseness ($H = 0$), and k-NN error rates for increasing values of k. The training and test sets sizes were over 8000 and 1000 strings respectively. Each algorithm has been tested with nine different training/test sets. Fig. 7 shows the behaviour of Ak-LAESA

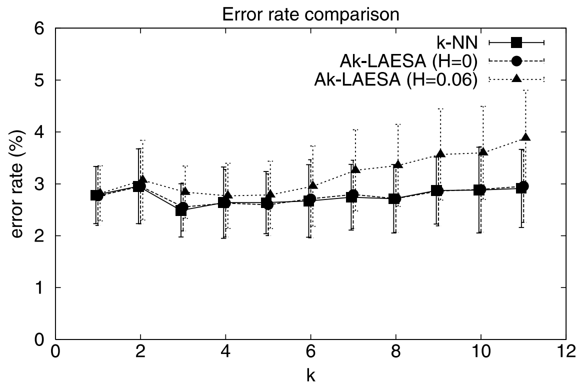
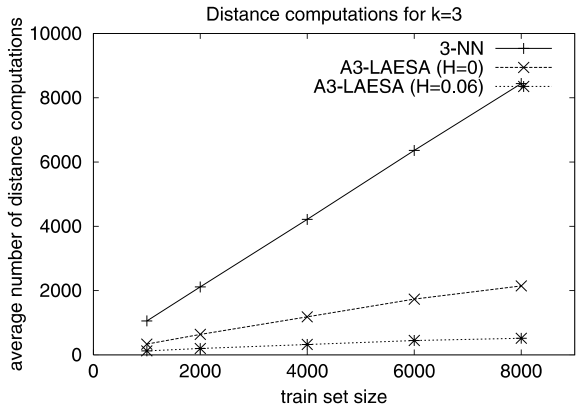


Fig. 7. Comparison of error rates for handwritten digits.



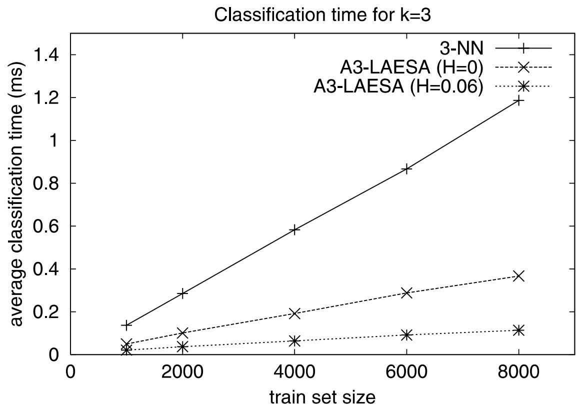


Fig. 8. Distances computed and classification times as training set increases.

without looseness, Ak-LAESA with a looseness of 0.06, and k-NN. The error rate of Ak-LAESA with looseness is higher because it is pruning good candidates to NN. Anyway, the difference is always $<1\%$ error.

The second experiment was developed to compare the number of computed distances (and classification times) of both algorithms for increasing training set sizes. These sizes range from 1000 to 8000, with 1000 test samples. Fig. 8 shows the results obtained for $k = 3$ (results for other values of k were similar). Although introducing looseness slightly increases error rates ($<1\%$ in our experiments), the important gaining in distances[4] and times makes this idea worth considering seriously.

4. Conclusions

In this work we have presented the Ak-LAESA, a fast classifier for general metric spaces (no vector space required) based on the LAESA algorithm. It obtains error rates very close to those of a k-NN classifier, while calculating a much lower number of distances (the number of distances of the Ak-LAESA algorithm is exactly the same that the computed by the LAESA algorithm). The adequate time and space complexities of LAESA have been preserved.

As for the future, we plan to study in depth the reason of the increase of the error rate whit respect to the k-NN when the k is beyond a limit. We are also interested in applying the algorithm to other real data tasks.

Acknowledgements

The authors wish to thank José Manuel Iñesta and Mikel L. Forcada for their careful reading of this paper, and Eva Gómez-Ballester for her invaluable help in processing handwritten digits from NIST database. Our thanks is also due to the Spanish CICyT for partial support of this work through project TIC2000–1703-CO3-02.

References

Aibar, P., Juan, A., Vidal, E., 1993. Extensions to the approximating and eliminating search algorithm (aesa) for finding k-nearest-neighbours. In: New Advances and Trends in Speech Recognition and Coding, pp. 23–28.

Bentley, J.L., 1975. Multidimensional binary search trees used for associative searching. Commun. ACM 18, 509–517.

Duda, R., Hart, P., 1973. Pattern Recognition and Scene Analysis. Wiley, New York.

Fukunaga, K., Narendra, M., 1975. A branch and bound algorithm for computing k-nearest neighbors. IEEE Trans. Comput. 24, 750–753.

Jain, A.K., Dubes, R.C., 1988. Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs, NJ.

Kalantari, I., McDonald, G., 1983. A data structure and an algorithm for the nearest point problem. IEEE Trans. Softw. Eng. 9, 631–634.

Lee, E.-W., Chae, S.-I., 1998. Fast design of reduced-complexity nearest-neighbor classifiers using triangular inequality. IEEE Trans. Pattern Anal. Mach. Intell. 20 (5), 562–566.

Micó, L., Oncina, J., 1998. Comparison of fast nearest neighbour classifiers for handwritten character recognition. Pattern Recogn. Lett. 19, 351–356.

Micó, L., Oncina, J., Carrasco, R.C., 1996. A fast branch and bound nearest neighbour classifier in metric spaces. Pattern Recogn. Lett. 17, 731–739.

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

Ramasubramanian, V., Paliwal, K.K., 2000. Fast nearest-neighbor search algorithms based on approximation-elimination search. Pattern Recogn. 33, 1497–1510.

Vidal, E., 1986. An algorithm for finding the nearest neighbour in (approximately) constant time. Pattern Recogn. Lett. 4, 145–157.

[4] Using looseness, Ak-LAESA distances are about 6% of the number of distances of an exhaustive k-NN search. Without looseness, this percentage becomes 25%.