Improving the Linear Approximating and Eliminating Search Algorithm (LAESA) error rates

Francisco Moreno-Seco, José Oncina, Luisa Micó*

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante
E-03071 Alicante, Spain.
e-mail: `{paco,oncina,mico}@dlsi.ua.es`

Abstract. The LAESA algorithm finds the nearest neighbour in a metric space with a low number of distance calculations. We have developed a new variant of this algorithm that improves its classification error rates. The main idea is to prematurely stop the search for the nearest neighbour when the number of non-eliminated training prototypes is less than or equal to a certain number K (thus, we have called the algorithm *Approximating* K LAESA, AK–LAESA). The new algorithm preserves the space and time complexities and the number of distance calculations of the LAESA algorithm. Actually, the new algorithm is slightly faster than the LAESA algorithm and obtains lower error rates.

Keywords: Nearest Neighbour, LAESA, Pattern Recognition.

1 Introduction

One of the typical tasks in Pattern Recognition is to classify an unknown sample as belonging to one of several classes. The most widely used technique to accomplish this task is to classify the sample as belonging to the class of its nearest neighbour. There are some applications in which the computation of the distance between two elements requires too much time, and thus this classification technique becomes impractical. Recently, some algorithms have been developed to find the nearest neighbour in linear time while highly reducing the number of distance computations [3, 4, 5], thus allowing for a practical use of this classification method in those applications where the computation of the distance is very time–consuming. One of these algorithms is LAESA [4], which finds the nearest neighbour to a given sample, while maintaining the average number of distance computations bounded by a constant (experimentally determined) that is independent of the size of the training set.

The classical nearest neighbour classifier usually obtains acceptable results, but it is possible to obtain better error rates using K neighbours to classify the sample instead of

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

using only one [2]. Recently, Aibar et al. proposed an extension of the AESA algorithm that uses several neighbours to classify a sample [1].

In this article we describe an improvement of the LAESA algorithm that also uses several neighbours[1] to classify the sample (AK–LAESA). Our experiments show that the new algorithm obtains lower error rates than the LAESA algorithm[2], while maintaining its key feature, that is, that the average number of distance computations shows to be bounded by a constant (experimentally) independent from the training set size. The AK–LAESA algorithm does not find exactly the K nearest neighbours. Instead, it uses some of them to classify the sample, and also uses some other neighbours very close to the sample that are not exactly the nearest neighbours.

2 The LAESA algorithm

The LAESA algorithm [4] relies on the triangle inequality to eliminate those prototypes from the training set which cannot possibly be the nearest neighbour. The algorithm is divided in two phases: a preprocessing phase, which is performed before the classification begins, and the search phase, which uses the data obtained in the preprocessing phase to find the nearest neighbour.

During the preprocessing phase, a number n_{bp} of base prototypes[3] is selected from the training set, and the distances between each of these base prototypes and all the other training prototypes (including the other base prototypes) are computed and stored. These base prototypes are selected in such a way that they are *maximally separated*. It has been shown that if base prototypes are maximally separated the number of distance computations is reduced in the following steps.

When the algorithm begins to classify, it starts to search for the nearest neighbour using those distances previously calculated as a lower bound of the real distance between the training prototypes and the test sample, thus eliminating the training prototypes whose distance to the sample (which is not actually computed) will be higher than the distance from the nearest neighbour (up to that moment) to the sample. The search phase performs a number of steps repeatedly until all training prototypes have been eliminated except the nearest neighbour. These steps can be sumarized as follows:

1. Compute the distance between a selected prototype s (initially one of the base prototypes) and the sample x, and eliminate this prototype.

2. Update the nearest neighbour obtained so far, and the lowest distance d_* to the sample.

3. For each prototype p not already eliminated,

   (a) Update, if possible, the lower bound $G[p]$ of the distance between the prototype p and the sample x, which is estimated using the distance between the selected prototype s and the prototype p ($D[p,s]$, calculated during the preprocessing phase), and the distance between s and x:

$$G[p] := max(G[p], |D[p,s] - d(x,s)|) \tag{1}$$

[1] The method used to obtain these neighbours is simpler than in [1].

[2] This result was expected, as the new algorithm uses more than one neighbour to classify the sample.

[3] The number of *base prototipes* depends on the dimensionality of data and must be adjusted experimentally.

This step is performed only if s is one of the base prototypes.

(b) Eliminate p if the lower bound of its distance to x is greater or equal to the lowest distance d_* computed so far (this step is called the *eliminating* step):

$$\textbf{if}\ \ G[p] \geq d_*\ \ \textbf{then}\ \ \text{eliminate}\ \ p\ \ \textbf{endif} \tag{2}$$

(c) Select a new prototype so that its lower bound is the minimum of all lower bounds. The algorithm is usually set to select a base prototype until all base prototypes have been eliminated. This is the *approximating step.*

The LAESA algorithm obtains the nearest neighbour while computing an average number of distances which has been experimentally proved to be constant with respect to the size of the training set.

3 The AK–LAESA algorithm

This algorithm is a new version of the LAESA algorithm which preserves exactly the same preprocessing phase and modifies the search phase described in the previous section. After *approximating and eliminating* steps have been completed, the new algorithm compares the number of remaining prototypes R with a constant number K. If the number of remaining prototypes is less or equal than K, it stops the search and classifies the sample by voting between these remaining prototypes, that is, it classifies the sample as belonging to the class that has more prototypes in the set of remaining prototypes.

During the development of this algorithm we observed that the proposed modification should only be performed if all base prototypes have been eliminated. If we let the algorithm stop when base prototypes are still alive, it can use some prototypes that are too far from the sample to classify it[4], thus obtaining very poor error rates when the number of training prototypes is low. When the number of training prototypes is increased, all base prototypes are eliminated and error rates decrease.

4 Experiments

We have performed a set of experiments using randomly generated data from two equally probable Gaussian classes, with a slightly different mean vector. All experiments have been developed using from 20 to 50 different training sets of increasing sizes, using the same test set for all the training sets in each experiment.

In the first experiment (see figure 1), the number R of remaining prototypes (which is always less than K) increases with the dimensionality of the data D, but hardly approximates to K. This experiment was performed using 50 different training sets of 1000 prototypes each one, classifying the same test set (also with 1000 samples). The experiment was repeated variying the dimensionality of data from 4 up to 10.

The second experiment was carried out to compute, given a fixed K, the average number of remaining prototypes (see figure 2) and the average number of distance calculations (see figure 3). Two conclusions may be derived from these experiments: first, the number R of remaining prototypes does not depend on the size of the training set T and tends to a constant value for each possible value of the dimensionality; second,

[4] Actually, some base prototypes are used to classify the sample, and as they are selected to be *maximally separated*, they will usually be very far from the sample.

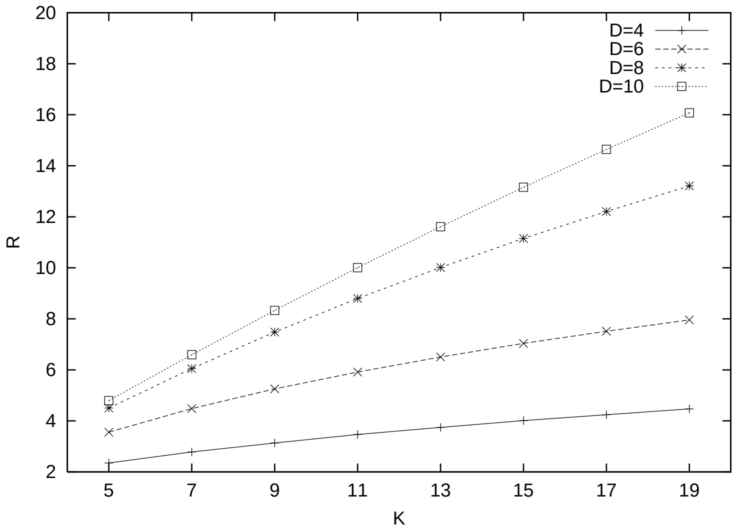

Figure 1: Number of prototypes remaining R for each K in the AK–LAESA, using training sets of 1000 prototypes.

the average number of distance calculations also tends for each dimension to a constant value that does not depend on the size of the training set. The latter behaviour is also observed with the LAESA algorithm. In this experiment we have set K to 17, and we have used 20 different training sets whose sizes range from 50 to 3000 prototypes; we used all these training sets to classify the same set of 100 samples. As in the previous experiment, we performed this one varying the dimensionality from 4 to 10.

The third experiment was carried out to determine whether the behaviour of this new classifier was similar to the behaviour of a K–NN classifier, comparing the average error rates of both types of classifiers. As shown in figure 4, the error rate of the AK–LAESA classifier is similar to that of the 1–NN classifier when there is a low number of training prototypes, and seems to approximate to the error rate of the K–NN classifier when the number of prototypes increases; this approximation becomes better as the dimensionality increases. This experiment was carried for $K = 17$, using 30 different training sets of increasing size (from 50 to 3000 prototypes) to train the AK–LAESA classifier, the 1–NN classifier and the K–NN classifier. All these classifiers were tested using a set of 1000 samples.

Finally, we have performed an experiment to compare the behaviour of the AK–LAESA with the behaviour of the LAESA[5], with respect to the number of distance calculations. As shown in figure 5, the AK–LAESA computes a lower number of distances than the LAESA, but, as we have figured out, it is not a significant reduction. To carry out this experiment we have used 20 different training sets of increasing sizes (from 50 to 3000 prototypes), testing both algorithms with a set of 100 samples. We have used data of dimensionalty 8, and we have varied K from 5 to 17.

5 Conclusions and future work

We have developed an algorithm that retains the most important feature of the LAESA algorithm —the number of distance calculations is bounded by a constant that does

[5]When K is set to 1, the AK-LAESA becomes the LAESA algorithm.

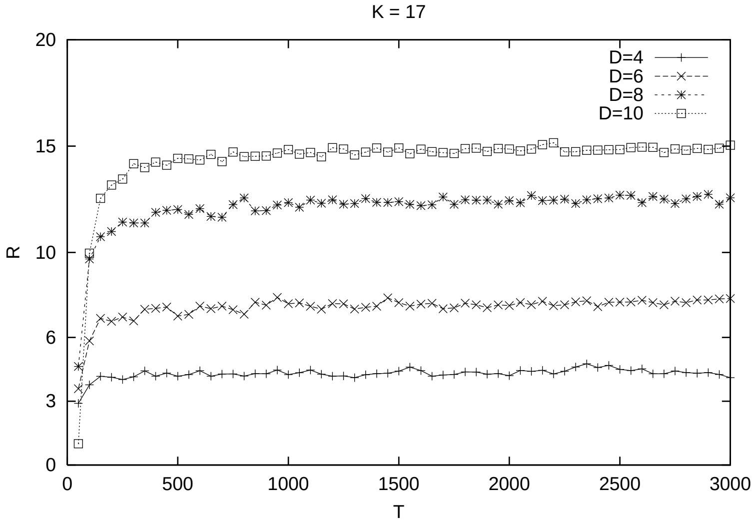

Figure 2: In the AK–LAESA, with $K = 17$, average number of prototypes remaining R as the size of the training set increases.

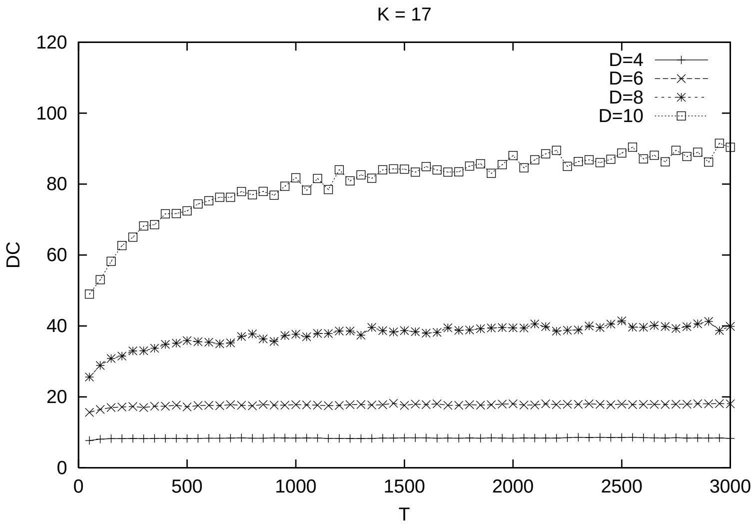

Figure 3: Average number of distance calculations DC in the AK–LAESA, with $K = 17$.

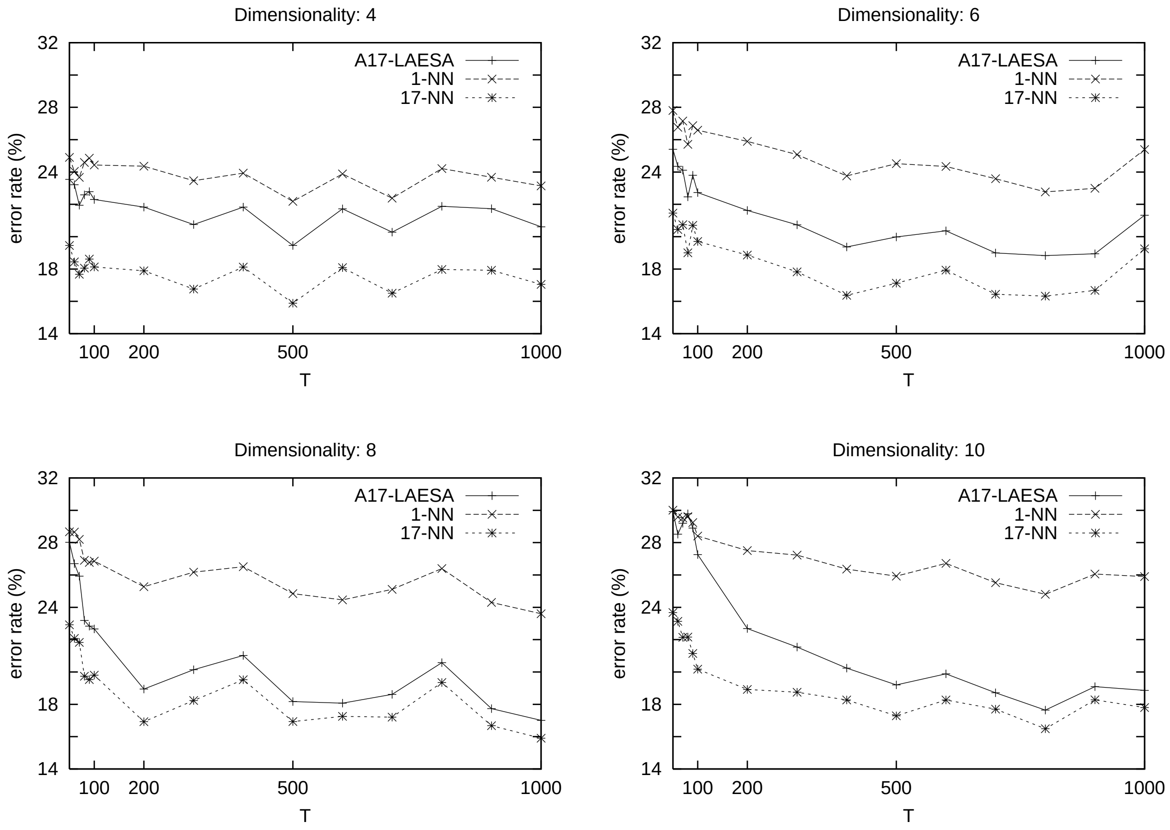

Figure 4: Error rates of the AK–LAESA classifier compared to NN–classifiers, with data of variying dimensionality.

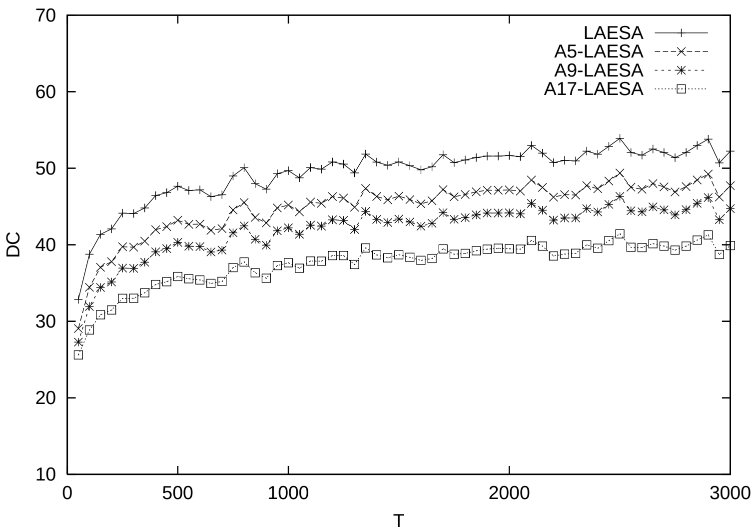

Figure 5: Comparison of LAESA and AK–LAESA algorithms with respect to the average number of distance calculations DC.

not depend on the size of the training set— while improving its error rates, specially with data of high dimensionality and large training sets. The new algorithm seems very suitable to very time-consuming tasks that also require low error rates.

The AK–LAESA algorithm simply stops the search of the nearest neighbour when the number of prototypes not eliminated is less or equal than a constant number K. The remaining prototypes were not in almost any case the K nearest neighbours, but some of them they were. These lead to error rates near to those obtained using the K–NN classifier, with a significant reduction in the average number of distance calculations. We will continue to explore new improvements to this algorithm that will lead us to error rates as close as possible to that of the K-NN classifiers, maintaining the number of distance calculations bounded by a constant independent from the training set size.

Acknowledgements: The authors wish to thank Mikel L. Forcada for his careful reading of this manuscript.

References

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

[2] R. Duda, P. Hart. "Pattern Classification and Scene Analysis". Wiley (1973).

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

[4] L. Micó, J. Oncina, 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, (1994) **15**, pp. 9–17.

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