An Algorithm for Finding Nearest Neighbours in Constant Average Time With a Linear Space Complexity*

Luisa Micó[1], José Oncina[1] and Enrique Vidal[2]

[1]Departamento de Sistemas Informáticos y Computación, Universidad de Alicante, Spain
[2]Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Spain

Abstract

Given a set of n points or "prototypes" and another point or "test sample", we present a new algorithm that finds a prototype that is a Nearest Neighbour of the test sample, by computing only a constant number of distances on the average. This is achieved through a preprocessing procedure that computes only a number of distances and uses an amount of memory that grow lineally with n. The algorithm is an improvement of the previously introduced AESA algorithm and, as such, does not assume the data to be structured into a vector space, making only use of the metric properties of the given distance.

1: Introduction

Finding Nearest Neighbours (NN) with respect to a certain dissimilarity measure or distance is a very usual procedure in Pattern Recognition. A great number of techniques have been introduced in the last few years to minimize the number of distance computations, since, in many cases of interest, such computations are very expensive. These techniques generally rely upon two basic properties: (1) Data can be adequately represented in a suitable vector space, and then coordinates of the samples are available; and/or (2) the dissimilarity measure is a metric. The techniques that only make use of the metric properties of the given distance are most interesting, since, being more general, they can accommodate a wider range of currently interesting Pattern Recognition applications. One example of these applications is Isolated Word Recognition [1]. All these techniques relie on a preprocessing phase that permits reducing the average computational expense of the search procedure to a sublineal function of the number of reference samples or *prototypes*. However, in general, they are not powerful enough when large sets of prototypes and/or costly dissimilarity measures are involved.

Recently, Vidal [2] proposed the AESA (Approximating and Eliminating Search Algorithm) that makes use of: (1) efficient elimination rules and (2) an appropriate search organization to allow maximum efficiency of the available rules. For N prototypes, the AESA makes use of a precomputed (by preprocessing) triangular array of $(N^2-N)/2$ distances between prototypes, which permits carrying out the NN-Search through a very small number of distance computations. Furthermore, for large N, this number of computations, tends to be independent of N (asymptotic constant time-complexity). However, the great storage complexity and preprocessing time ($O(N^2)$), severely limit the practical use of the AESA for large sets of prototypes.

In this regard, one interesting recent contribution is the one introduced by Ramasubramanian and Paliwal [3]. They use a "*spherical distance coordinate*" formulation, where a vector in K-dimensional space is represented uniquely by its distances from K+1 fixed points. This fact permits the use of a *linear (rather than quadratic)* precomputed array of (K+1)*N distances between prototypes, while keeping nearly unchanged the original, highly efficient search strategy of the AESA.

However, as previously mentioned, there are many problems of interest in Pattern Recognition, where the data cannot be properly represented in a vectorial form, for which these techniques cannot be applied. Nevertheless, the original AESA does not have this limitation since it only makes use of the metric properties of the dissimilarity measure.

We propose here a new version of the AESA, called LAESA (Linear AESA), which uses a linear array of distances, like in [3] but is strictly based on metric arguments like the original AESA. The procedure starts by selecting from the given set of prototypes a relatively small set, called "Base Prototypes" (BP), and computing the distances between these BP's and the complete set of prototypes. The choice of Base Prototypes and the resulting LAESA algorithm are presented in the following

Work supported in part by the Spanish CICYT under grant TIC 448/89

0-8186-2915-0/92 $3.00 © 1992 IEEE

sections. Also, a number of simulation experiments are reported to permit comparison with the original AESA.

2: Selection of Base Prototypes

The efficiency of the proposed search algorithm is related both to the number of selected Base Prototypes and to their emplacement. This last question was already addressed in an early work by Marvin Shapiro [4], whose results from computer simulations suggested that significant improvements could be achieved by locating reference points far away from data clusters.

Let E be the set from which all the samples are to be drawn and let d:ExE--->$\Re$ a dissimilarity measure which is assumed to fulfill the metric properties. We have adopted the following greedy strategy that tries to find a set of M BP's that are maximally separated among the given set of prototypes:

```
Algorithm BP-SELECTION
Input:  P ⊂ E, M ∈ ℕ; // finite set of prototypes and number of BP's //
Output: B ⊆ P;                          // set of Base Prototypes //
        D ∈ ℜ^(|P|×|B|);                // |P|·|B| interprototype distances //
Function  d: E x E ---> ℜ               // distance function //
Variables: A ∈ ℜ^|P|;                   // distance accumulator array //
           b, b' ∈ P;
           max ∈ ℜ;
begin
   b' := arbitrary_element(P); D := [0]; B := {b'}; A := [0];
   while |B| < M do
      max := 0; b := b';
      for every p ∈ P-B do
         D[b,p] := d(b, p);
         A[p] := A[p] + D[b,p];
         if (A[p] > max) then b' := p; max := A[p]; end_if
      end_for
      B := B ∪ {b'};
   end_while
end
```

While this procedure does not strictly guarantee the optimality (maximum separation) of the resulting set of BP's, it requires only a linear processing time and usually yields quite satisfactory results.

3: Searching Algorithm

The proposed algorithm performs both the search and elimination strategies by using the array of distances obtained by the preprocessing described above. The principles of the algorithm are described below.

Like in the original AESA, the metric properties of d:ExE--->$\Re$ are used by the elimination and search rules of the new algorithm. If q is a Base Prototype whose distance from x has already been computed and n is a prototype that is the current nearest neighbour of test sample x, the triangle inequality of d can be applied to obtain the following sufficient conditions for a prototype p not to be closer to x than n:

$$\text{(i)} \quad d(p,q) \geq d(x,q) + d(x,n)$$
$$\text{(ii)} \quad d(p,q) \leq d(x,q) - d(x,n)$$

or, equivalently,

$$|d(p,q) - d(x,q)| \geq d(x,n) \qquad (1)$$

On the other hand, using the new formulation introduced in [5], the approximation criterion used by Vidal [1] for selecting candidate prototypes that are close to the test vector x, becomes, in the case of Base Prototypes, as follows:

$$s = \underset{\forall q \in P-U}{\operatorname{argmin}} G(q) \qquad (2)$$

with

$$G(p) = \max_{\forall u \in Bu} | d(p,u) - d(x,u) |$$

where Bu is the set of Base Prototypes used so far during the searching procedure, P is the set of prototypes and U is the set of eliminated prototypes.

Using this formulation, the inequality (1) can be used to obtain the following sufficient condition for a prototype p to be eliminated without risk of loosing the true nearest neighbour:

$$G(p) \geq d(x,n) \qquad (3)$$

Using (2) and (3) the algorithm LAESA, given below, repeatedly performs five steps until all prototypes have been eliminated. These steps are: Distance computing, Updating the prototype nearest to x, Updating G, Eliminating and Approximating. The first execution starts with the computation of the distance from x to an arbitrarily selected Base Prototype. On the basis of (2), the Aproximation step successively selects non-eliminated prototypes from B as long as certain conditions are met; otherwise, a non-eliminated prototype from P-B is selected. In the algorithm, the CHOICE function helps deciding, depending on the conditions, wether a base or non-base prototype is selected. Another function called CONDITION is introduced to control the elimination of BP's. These two functions, permit an easy introduction of different strategies to manage the use and elimination of Base Prototypes. As will be shown in the next section, different behaviour of LAESA can be obtained for each of these strategies. The "Updating G" step, is only executed for Base Prototypes, since only for these prototypes are distances to other prototypes available in the precomputed matrix D.

Algorithm LAESA

```
Input:  P ⊂ E ;                          // finite set of prototypes //
        B ⊆ P ;                          // set of Base Prototypes //
        D ∈ ℜ^(|P|×|B|) ;                // precomputed |P|·|B| distances //
        x ∈ E;                           // test sample //
Output: n ∈ P; d* ∈ ℜ                    // NN prototype and its distance to x //
Function  d: E × E ---> ℜ                // distance function //
Function  CONDITION: Boolean             // controls elimination of BP's //
Function  CHOICE: B×(P-B) → P            // selection of a BP or non-BP //

Variables:  p, q, s, b ∈ P
            G ∈ ℜ^|P|                    // lower bound array //
            dxs, gp, gq, gb ∈ ℜ
begin
  d*:= ∞; n:= indeterminate; G:= [0]; s:= arbitrary_element(B);
  while |P| > 0 do
    dxs:= d(x,s); P:= P-{s};             // distance computing //
    if dxs < d* then n:= s; d*:= dxs; end_if   // updating n, d* //
    q:= indeterminate; gq:= ∞; b:= indeterminate; gb:= ∞;
    for every p ∈ P do                   // eliminating and approximating loop //
      if s ∈ B then                      // updating G, if possible //
        G[p]:= max( G[p], | D[p,s]-dxs | )
      end_if
      gp:= G[p];
      if p ∈ B then
                                         // eliminating from B //
        if ( gp ≥ d* & CONDITION ) then P:= P-{p}
        else                 // approximating: selecting from B //
          if gp<gb then gb:= gp; b:= p end_if
        end_if
      else
        if gp ≥ d* then P:= P-{p};       // eliminating from P-B //
        else             // approximating: selecting from P-B //
          if gp<gb then gq:= gp; q:= p end_if
        end_if
      end_if
    end_for
    s:= CHOICE(b, q);
  end_while
end.
```

A number of different strategies for using and eliminating Base Prototypes were studied in this work. The implementation of the function CHOICE is the same for all these strategies and simply consists of selecting a BP whenever possible; that is:

CHOICE(b,q) = **if** b ≠ indeterminate **then** b **else** q **end_if**

On the other hand, the different implementations of the function CONDITION are given hereafter, where "Bu" stands for the set of Base Prototypes that have been used (selected) in previous steps of the algorithm:

a) EC1 *//never eliminate BP's explicitly//*

CONDITION = false

b) EC2 *//only allow elimination of BP's after having selected a number of prototypes greater than one half of B//*

CONDITION = (|Bu| > |B|/2)

c) EC3 *//only allow elimination of BP's after having selected a number of prototypes greater than one third of B//*

CONDITION = (|Bu| > |B|/3)

d) EC∞ *//always allow elimination of BP's//*

CONDITION = true

e) ECELIM *//allow elimination of BP's if the last selected prototype contributed no elimination//*

CONDITION = (no prototype was eliminated in the previous step)

4: Results and discussion

The behaviour of the LAESA with the above BP management strategies was tested in a serie of experiments. A random set of 1024 prototypes was drawn from a uniform distribution in the unit 6-dimensional hypercube. With this set, the BP-selection algorithm was repeatedly executed, yielding several sets of BP's of different sizes. Then, another random set of 1000 test samples was submitted to LAESA using the different BP and management strategies. This procedure was repeated 10 times with different random number generator seeds. The results averaged over the 10 executions are shown in Fig. 1.

When all the prototypes are considered BP's, the number of distance computations for both EC∞ and ECELIM tends to be nearly the same as for the AESA. Moreover, for the ECELIM strategy in particular, the actual decision for a specific number of BP's is less critical, since only rather negligible differences in performance are observed for a very wide range in the number of Base Prototypes.

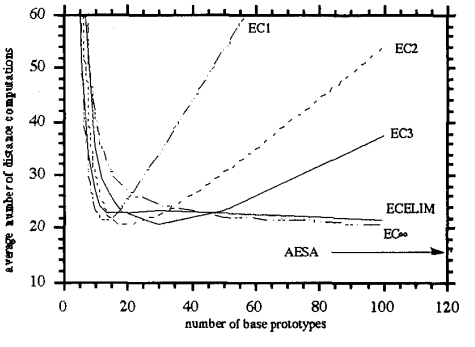

Figure 1. Average number of distance computations for the LAESA as a function of the number of Base Prototypes, for a random set of 1024 6-dimensional prototype vectors using the Euclidean Metric.

Using the first version (EC1), we have performed some experiments to study how the number of distance computations grows with the size of the prototype set [6]. The obtained results show that the average numbers of distance computations both for the AESA and LAESA are dramatically smaller than for other methods previously proposed in the literature and, when the number of prototypes is large enough, the performances tend to be independent of this number in both cases (asymptotic constant time complexity). Moreover, the average number of distance computations required by LAESA tend to be only less than 1.5 times that of the original AESA, while space requirements were smaller by orders of magnitude. The same experiments were repeated for different number of BP's and varying dimensions. The results, displayed in Fig. 2, suggest that both the "optimum" number of BP's (that leading to the smallest number of distance computations)and the average number of distance computations tend to increase rapidly with the dimension.

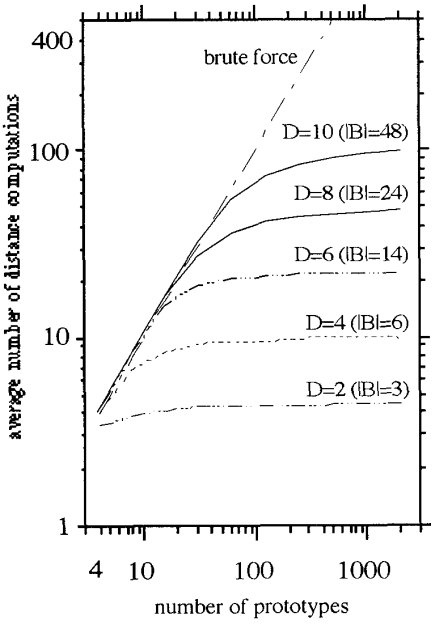

Figure 2. Average number of distance computations for LAESA with EC1 as a function of the number of prototypes and different dimension, using de Euclidean Metric.

This behaviour with uniformly distributed data is the same observed for the AESA [2]. Therefore, as it happens with the AESA, this rapid increase is not actually expected for real data, in which increasing the complexity or "intrinsic dimension" of an (appropiate) object representation does not generally lead to an increase in the distances from test samples to prototypes[1]. Finally, we performed other experiments to compare two choice for the emplacement of BP's: randomly among the complete set of prototypes and chosen through the greedy, maximum separation procedure described in section 2. The best results were achieved, as expected, with the second method.

Although the results using the LAESA are not as good when using AESA (1.3 times worse on the average), the use of AESA entails a quadratic storage complexity, whereas the storage complexity of the LAESA is linear. Therefore the use of large sets of prototypes is not a problem now.

5: References

[1] E. Vidal, H. Rulot, F. Casacuberta and J. Benedí, "On the use of a metric-space search algorithm (AESA) for fast DTW-bassed recognition of isolated words", IEEE Transactions on Acoustics, Speech, and Signal Processing., vol. 36, pp. 651-660, 1988.

[2] E. Vidal, "An algorithm for finding nearest neighbours in (approximately) constant average time complexity.", Pattern Recognition Letters, vol. 4, pp. 145-157, 1986.

[3] V. Ramasubramanian and K. K. Paliwal, "An Efficient Approximation-Elimination Algorithm for Fast Nearest-Neighbour Search Based on a Spherical Distance Coordinate Formulation.", Signal Processing V: Theories and Applications., pp. 1323-1326, 1990.

[4] M. Shapiro, "The Choice of Reference Points in Best-Match File Searching.", Artificial Intelligence/Language Processing., vol. 20, pp. 339-343, 1977.

[5] E. Vidal, "New formulation and improvements of the Nearest-Neighbour Approximating and Eliminating Search Algorithm (AESA).", To be published, 1991.

[6] L. Micó, J. Oncina and E. Vidal, "Algoritmo para encontrar el vecino más próximo en un tiempo medio constante con una complejidad espacial lineal", Tech. Report DSIC II/14-91. Universidad Politécnica de Valencia, 1991.