Some Improvements in Tree Based Nearest Neighbour Search Algorithms

Eva Gómez-Ballester, Luisa Micó, and Jose Oncina*

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

Abstract. Nearest neighbour search is one of the most simple and used technique in Pattern Recognition. In this paper we are interested on tree based algorithms that only make use of the metric properties of the space. One of the most known and refereed method in this class was proposed by Fukunaga and Narendra in the 70's.
This algorithm uses a tree that is traversed on search time and uses some elimination rules to avoid the full exploration of the tree.
This paper proposes two main contributions: two new ways for constructing the tree and two new elimination rules. As shown in the experiment section, both techniques reduce significantly the number of distance computations.

Keywords: Nearest Neighbour, Metric Spaces, Elimination rule, Pattern Recognition.

1 Introduction

Nearest neighbour search (NNS) is a simple technique very popular in problems related with classification. The NNS method consists on finding the nearest point from a prototype set to a given sample point using a distance function [3].

To avoid the exhaustive search many algorithms have been developed in the last thirty years [1]. One of the most popular and refereed algorithm was proposed by Fukunaga and Narendra [4].

This algorithm builds, on preprocess time, a tree that is traversed in search time using some elimination rules to avoid the exploration of some branches.

The algorithm does not make any assumption on the way the points are coded. It can be used in any metric space, that is, the distance function has to fulfil the following conditions:

- $d(x, y) \geq 0 \quad (= 0 \text{ if } x = y)$.
- $d(x, y) = d(y, x)$ (symmetry).
- $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality).

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

Although some recently proposed algorithms are more efficient, the Fukunaga and Narendra algorithm is a basic reference in the literature and in the development of new rules to improve the main steps of the algorithm that can be easily extended to other tree based algorithms [9] [10] [2] [6] [7].

In this paper we propose two new ways of building the tree and two new elimination rules.

2 The Fukunaga and Narendra Algorithm

The Fukunaga and Narendra algorithm is a fast search method that use a hierarchical clustering to build a search tree where all the prototypes are stored. In particular, it uses a divisive strategy splitting the training data into l subsets. Moreover each subset is divided into l subsets again, and applying recursively this procedure a search tree is built. Fukunaga and Narendra proposed to use the c-means at each step. Each node p of the tree represents a group of samples, and is characterised by the following parameters:

- S_p set of samples
- N_p number of samples
- M_p representative of S_p
- $R_p = \max_{x_i \in S_p} d(x_i, M_p)$, (the radius of the node)

When an unknown sample x is given, the nearest neighbour is found by searching in the tree by first-depth strategy. Among the nodes at the same level, the node with a smaller distance $d(x, M_p)$ is searched earlier. Let n be the current nearest neighbour to x among the prototypes considered up to the moment, the following two rules permit to avoid the search in the subtree p:

rule for internal nodes: no $y \in S_p$ can be the nearest neighbour to x if

$$d(x, n) + R_p < d(x, M_p)$$

rule for leaf nodes: $y \in S_p$ cannot be the nearest neighbour to x if

$$d(x, n) + d(y, M_p) < d(x, M_p)$$

In this work only binary trees with one point on the leafs[1] are considered. On such case the second rule becomes a special case of the first one. This rule will be refereed as the Fukunaga and Narendra's rule (FNR).

3 The Searching Tree

Two approximations have been developed as alternative to the use of the well known c-means algorithm that was recursively used by Fukunaga in the construction of the tree structure.

The first clustering strategy is called *Most Separated Points* (MSP), and consists on:

[1] In the Fukunaga and Narendra algorithm, leaf nodes can have more than one point.

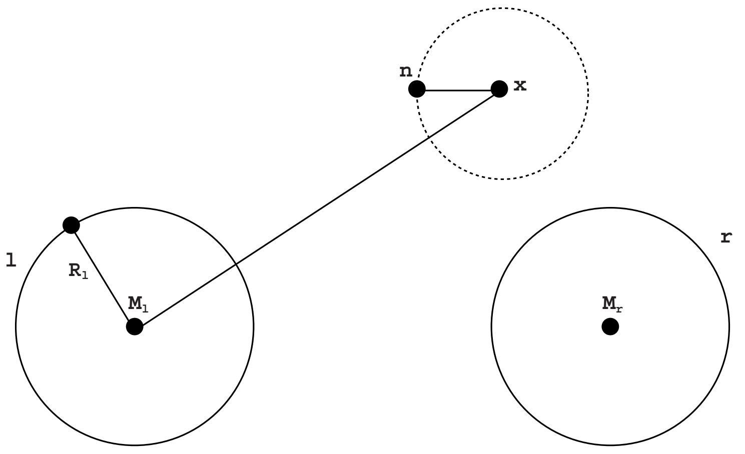

Fig. 1. Original elimination rule used in the algorithm of Fukunaga and Narendra (FNR).

- use as representative of the two children of each node the two more separated prototypes,
- classify the rest of the prototypes in the node of the nearest representative,
- recursively repeat the process until each final node has only a prototype, the representative.

The second clustering strategy is called *Most Separated Father Point* (MSFP)^2, and consists on:

- randomly select a prototype as the representative of the root node,
- in the following level, use as representative of one of the nodes the representative of the father node. The representative of the sibling node is the farthest prototype among all the prototypes belonging to the father node.
- classify the rest of the prototypes in the node of the nearest representative,
- recursively repeat the process until each leaf node has only one point, the representative.

Of course, the second strategy is not as symmetric as the first one and it will produce deeper trees. On the other hand, this strategy permits to avoid the computation of some distances in the search procedure as one of the representatives is the same than the father, each time that it is necessary to expand a node, only one new distance computation is needed.

4 The New Rules

The elimination rules defined by Fukunaga and Narendra only make use of the information between the node to be prune and the hiperespherical surface cen-

[2] note that this tree can be built in $O(n\log(n))$, where n is the prototype set size

tred in the test sample with radius the distance to the nearest point considered up to the moment.

In the proposed new rules, to eliminate a node l, also information related with the sibling node r is used.

4.1 The Sibling Based Rule (SBR)

A first proposal requires that each node r stores the distance between the representative of the node, M_r, and the nearest point, e_ℓ, in S_ℓ.

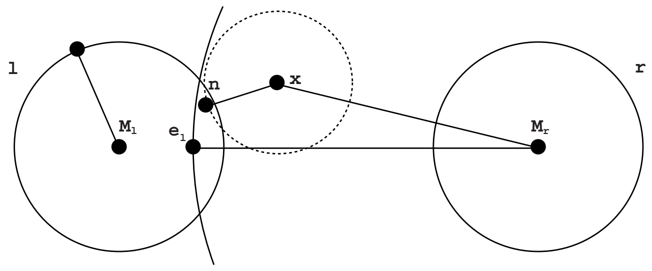

Fig. 2. Sibling based rule (SBR).

Definition of SBR: given a node r, a test sample x, an actual candidate to be the nearest neighbour n, and the nearest point to the representative of the sibling node ℓ, e_ℓ, the node ℓ can be prune if the following condition is fulfil:

$$d(M_r, e_\ell) > d(M_r, x) + d(x, n)$$

It is interesting to see that this rule don't need to know the distance between x and M_ℓ. That will permit to avoid some distance computations in the search procedure[3].

4.2 Generalised Rule (GR)

This rule is an iterated combination of the FNR and the SBR. Let l be a node, to apply this rule, first a set of points $\{l_i\}$ is defined in the following way:

$$G_1 = S_\ell$$
$$\ell_i = \text{argmax}_{p \in G_i} d(p, M_\ell)$$

[3] In the search procedure, each time a node expansion is needed, the distances between each representative of the children to the test sample is calculated. After that, the elimination rules are applied.

Now, as the new rule SBR don't use $d(x, M_\ell)$, ℓ can be eliminated before computation of $d(x, M_\ell)$.

$$G_{i+1} = \{p \in G_i : d(p, M_r) < d(\ell_i, M_r)\}$$

In preprocessing time, the distances $d(M_r, \ell_i)$ are stored in each node ℓ.

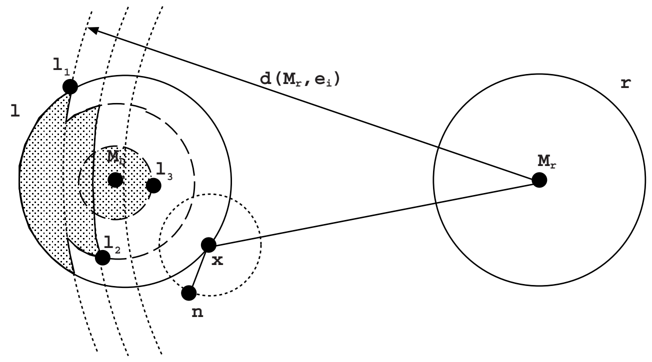

Fig. 3. Generalised rule (GR).

Definition of GR: given two sibling nodes l and r, a test sample x, an actual candidate to be the nearest neighbour n, and the list of point $\ell_1, \ell_2, \ldots, \ell_s$, the node ℓ can be prune if there is an integer i such that:

$$d(M_r, \ell_i) \geq d(M_r, x) + d(x, n) \tag{1}$$

$$d(M_\ell, \ell_{i+1}) \leq d(M_\ell, x) - d(x, n) \tag{2}$$

Cases $i = 0$ and $i = s$ are also included not considering equations (1) or (2) respectively. Note that condition (1) is equivalent to SBR rule when $i = s$ and conditioni (2) is equivalent to FNR rule when $i + 1 = 1$.

5 Experiments

Some experiments with synthetic and real data were carried out to study the behaviour of the algorithm.

The prototypes in the synthetic experiment set were extracted from a 6-dimensional uniform distribution in the unit hypercube. The Euclidean distance was used.

All the experiments were repeated with 10 different sets of prototypes, 1000 samples were used as test.

A first set of experiments were carried out in order to study the behaviour of the algorithm when using c-means (the proposed by Fukunaga and Narendra), MSP and MSFP clustering algorithm to build the tree (fig. 4).

The experiments show that *c*-means and MSP have a very similar behaviour[4] and that MSFP is neatly superior. This is because the saving of one distance computation at each level compensates the fact that the trees are deepest.

A second set of synthetic experiments were carried out to show the behaviour of the algorithm when using the elimination rules FNR, FNR+SBR and GR using the MSFP clustering algorithm (see fig. 5).

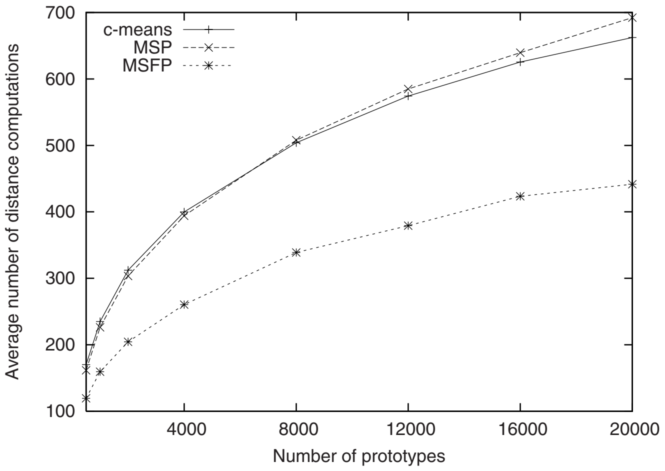

Fig. 4. Influence on the Fukunaga and Narendra algorithm when c-means, MSP and MSFP clustering algorithms are used in the tree building process with a 6-dimensional uniform distribution.

As was expected the addition of the SBR reduces slightly the number of distance computations but the GR[5] reduces it drastically.

Some experiments using real data have also been made. In particular, PHONEME database from the ROARS ESPRIT project [8] was used. The PHONEME database consists of 5404 5-dimension vectors from 2 classes. The set was divided in 5 subsets, using 4 sets as prototypes and 1 set as samples. A leaving one out technique was used.

The results plotted in figure 6 show the average number of distance computations as the size of the training set increases. All the combinations of the tree clustering algorithms and FNR and GR are shown. The behaviour using this data seems similar to the obtained results with artificial data, as figure illustrates.

[4] note that MSP is much faster than *c*-means

[5] note that the FNR and the SBR are special cases of the GR

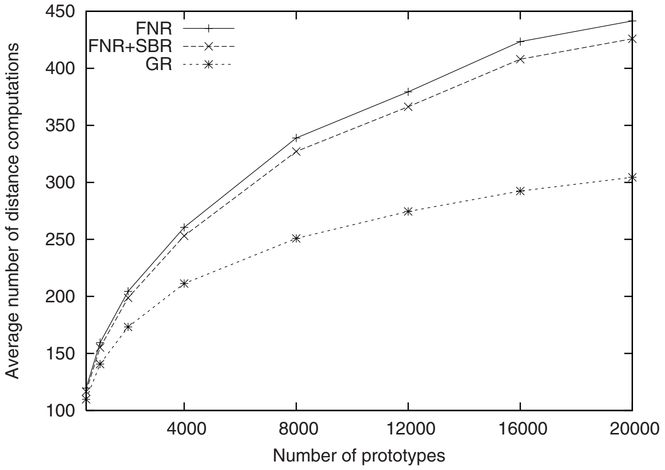

Fig. 5. Influence on the Fukunaga and Narendra algorithm when MSPF is used to build the tree and the rules FNR, FNR+SPR and GR are used in the search.

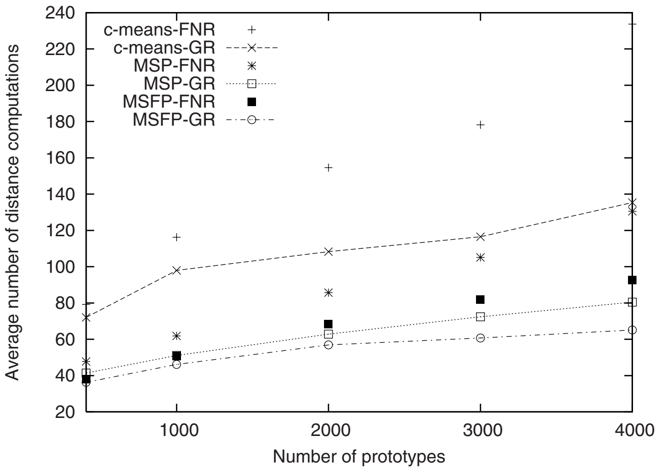

Fig. 6. Average number of distance computations by sample in relation to the size of the training set for the PHONEME database.

6 Conclusions

In this paper we have developed a series of improvements based on the algorithm proposed by Fukunaga and Narendra. These algorithm builds a tree in preprocess time to speed the nearest neighbour search.

On the one hand, two new methods to build the tree has been proposed. This tree is quicker to build and allows the search algorithm to find the nearest neighbour with less distance computations.

On the other hand, two new elimination rules are proposed to speed up the nearest neighbour search. The experiments suggest that high speed ups can be obtained.

In the future, we plan to apply these approximations to other nearest neighbour search algorithms based on a tree structure. We are also interested in testing these techniques in general metric spaces.

References

1. Belur V. Dasarathy: Nearest Neighbour(NN) Norms: NN Pattern Classification Techniques. IEEE Computer Society Press (1991).
2. Chen, Y.S., Hung, Y.P., Fuh, C.S.: Fast Algorithm for Nearest Neighbour Search Based on a Lower Bound Tree. Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada, (2001), **1** 446–453
3. Duda, R., Hart, P.: Pattern Classification and Scene Analysis. Wiley (1973)
4. Fukunaga, K., Narendra, M.: A branch and bound algorithm for computing k–nearest neighbours. IEEE Trans. Computing (1975) **24** 750–753
5. Kalantari, I., McDonald, G.: A data structure and an algorithm for the nearest point problem. IEEE Trans. Software Engineering (1983) **9** 631–634
6. L. Miclet, M. Dabouz: Approximate fast nearest-neighbour recognition. Pattern Recognition Letters (1983) **1**277–285.
7. 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
8. Alinat, P.: Periodic progress report 4, ROARS project ESPRIT II – Number 5516. Thomson Technical Report TS ASM 93/S/EGS/NC/079 (1993)
9. S. Omachi, H. Aso: A fast algorithm for a k-NN classifier based on branch and bound method and computational quantity estimation. Systems and Computers in Japan, vol. 31, no 6, pp. 1–9 (2000).
10. Geofrey I. Webb: OPUS: An Efficient Admissible Algorithm for Unordered Search. Journal of Artificial Intelligence Research 3 (1995) 431–465.