Testing Some Improvements of the Fukunaga and Narendra's Fast Nearest Neighbour Search Algorithm in a Spelling Task

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.
One of the most known fast nearest neighbour algorithms was proposed by Fukunaga and Narendra. The algorithm builds a tree in preprocess time that is traversed on search time using some elimination rules to avoid its full exploration.
This paper tests two new types of improvements in a real data environment, a spelling task. The first improvement is a new (and faster to build) type of tree, and the second is the introduction of two new elimination rules.
Both techniques, even taken independently, reduce significantly both: the number of distance computations and the search time expended to find the nearest neighbour.

1 Introduction

The Nearest Neighbour Search method consists on finding the nearest point of a set to a given test point using a distance function [3].

To avoid the exhaustive search many effective algorithms have been developed [1]. Although some of such algorithms as K-dtrees, R-trees, etc. depend on the way the points are represented (vectors usually), in this paper we are going to focus on algorithms that does not make any assumption on the way the points are represented making them suitable to work in any metric space.

The most popular and refereed algorithm of such type was proposed by Fukunaga and Narendra (FNA) [4]. Although some recently proposed algorithms are more efficient, the FNA 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 [2, 6, 9].

⋆ The authors thank the Spanish CICyT for partial support of this work through project TIC2003–08496-CO4, GV04B-541, GV04B-631, and the IST Programme of the Europen Community, under the PASCAL Network of Excelence, IST–2002-506778. This publication only reflects the authors' view.

Recently we proposed some improvements in this algorithm [11] that reduce significantly the number of distance computations. However, those improvements were tested only with data represented in a vector space. In this work the algorithm is checked in a spelling task where the points are represented by strings and the distance function is the edit distance. Also we compare our proposal with Kalantari and McDonalds method as well as with FNA.

2 The Fukunaga and Narendra Algorithm

The FNA is a fast search method that uses a tree structure. Each node p of the tree represents a group of points S_p, and is characterised by a point $M_p \in S_p$, (the representative of the group S_p), and its distance R_p of the farthest point in the set (the radius of the node). The tree is built using recursively the c-means clustering algorithm.

When a new test point x is given, its nearest neighbour n is found in the tree using a first-depth strategy. Among the nodes at the same level, the node with a smaller distance $d(x, M_p)$ is searched earlier. In order to avoid the exploration of some branches of the tree, the FNA uses a prune rule.

Rule: if n is the nearest neighbour to x up to the moment, no $y \in S_p$ can be the nearest neighbour to x if

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

This rule will be referenced as the Fukunaga and Narendra's Rule (FNR) (see fig. 1 for a graphical interpretation).

The FNA defines another rule in order to avoid some distance computations in the leaves of the tree. In this work only binary trees with one point on the leaves are considered. On such case the rule related to leaf nodes becomes a special case of the FNR and will not be considered on the following.

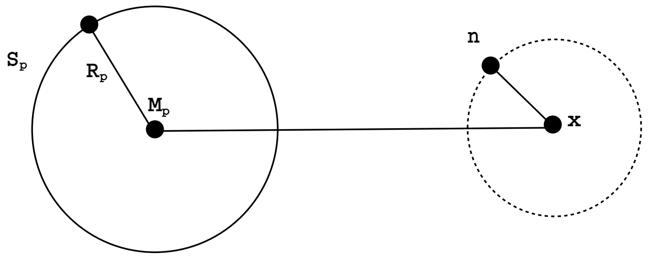

Fig. 1. Original elimination rule used in the algorithm of Fukunaga and Narendra (FNR).

3 The Search Tree

In previous works [11] some approximations were developed as an alternative to the use of the c-means algorithm on the construction of the tree. The best behaviour was obtained by the method called *Most Distant from the Father tree* (MDF). In this work this strategy is compared with c-means strategy[1] and with the incremental strategy to build the tree proposed by Kalantari and McDonalds [5], since this last strategy builds a binary tree similar to ours. Given a set of points, the MDF strategy consists on

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

This strategy reduces the computation of some distances in the search procedure as the representative of the left node is the same than the representative of its father. Each time a expansion of the node is necessary, only one new distance should be computed. Note that the construction of this tree is much faster than the construction of the FN tree where the c-means algorithm is used recursively.

While in the MDF method the average time complexity is $O(n\log(n))$, in the case that c-means algorithm is used, the average complexity is $O(n^2\log(n))$.

4 The New Elimination Rules

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

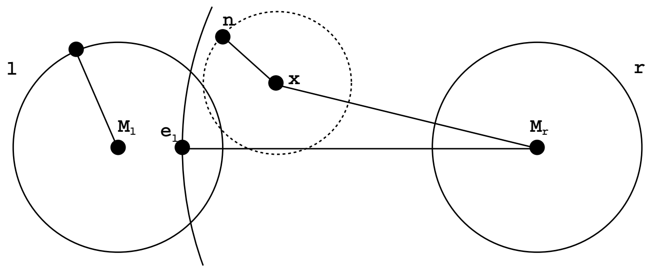

Fig. 2. Sibling based rule (SBR).

[1] As data are strings, the mean of a set of points can't be obtained. In this case the median of the set is used (c-medians).

4.1 The Sibling Based Rule (SBR)

The main idea of this rule is to use the distance from the representative to the nearest prototype of the sibling node. If this distance is too big, the sibling node can be safely ignored. Kamgar-Parsi and Kanal [10] proposed, for the FN algorithm, a similar rule (KKR), but the distance from the mean to the nearest prototype in the node was used. Note that in our case the representative is always a prototype in the node, then this distance is always zero. Moreover, in our case the rule allows the pruning of the sibling node, in the KKR case is the own node that can be pruned.
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 the sibling node ℓ.

Definition 1. ***Definition of SBR:*** *given a node r, a test sample x, an actual nearest neighbour n, and the nearest point to the representative of the sibling node ℓ, e_ℓ, the node ℓ can be pruned if the following condition is fulfil (fig. 2):*

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

Unlike the FNR, SBR can be applied to eliminate node ℓ without compute $d(M_\ell, x)$. That permits to avoid some distance computations in the search procedure.

4.2 Generalised Rule (GR)

This rule is an iterated combination of the FNR and the SBR. Given a node ℓ, a set of points $\{t_i\}$ is defined in the following way:

$$\begin{aligned} G_1 &= S_\ell \\ t_i &= \text{argmax}_{p \in G_i} d(p, M_\ell) \\ G_{i+1} &= \{p \in G_i : d(p, M_r) < d(t_i, M_r)\} \end{aligned}$$

where M_r is the representative of the sibling node r, and G_i are auxiliary sets of points needed in the definition (fig. 3). In preprocessing time, the distances $d(M_r, t_i)$ are stored in each node ℓ. In the same way, this process is repeated for the sibling node.

Definition 2. **Definition of GR:** *given two sibling nodes ℓ and r, a test sample x, an actual nearest neighbour n, and the list of point $t_1, t_2, \ldots, t_s$, the node ℓ can be pruned if there is an integer i such that:*

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

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

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 condition (2) is equivalent to FNR rule when $i = 0$.

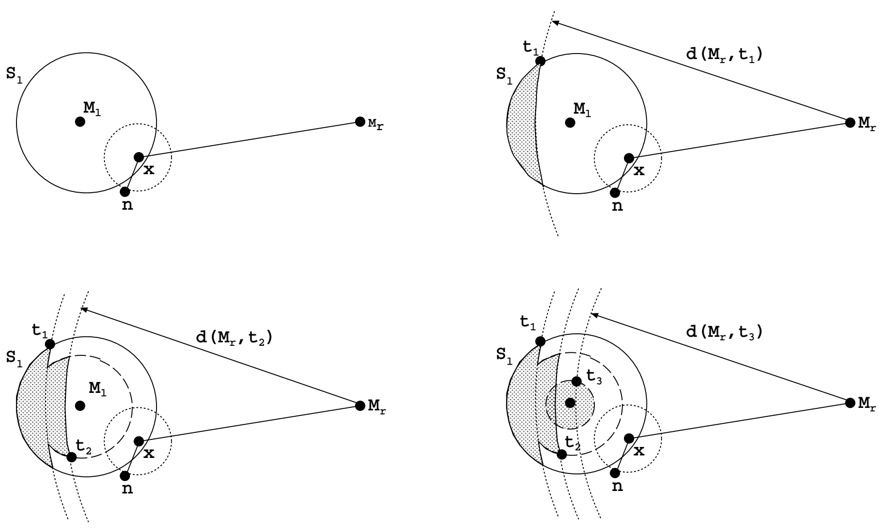

Fig. 3. Generalised rule (GR).

5 Experiments

To show the performance of the algorithm some tests were carried out on a spelling task. A database of 38000 words of a Spanish dictionary was used. The input test of the speller was simulated distorting the words by means of random insertion, deletion and substitution operations over the words in the original dictionary. The edit distance was used to compare the words.

Increasing size of dictionaries (from 1000 to 38000, 9 different sizes) was obtained extracting randomly words of the whole dictionary. The test points were 1000 distorted words obtained from randomly selected dictionary words. To obtain reliable results the experiments were repeated 10 times. The averages and the standard deviations are showed on the plots. The distance computation is referenced per test point, and the search time per test set.

In order to study the contribution of the elimination rules FNR, FNR+SBR and GR a first set of experiments were carried out using the original *c*-means tree construction of the FN algorithm (fig. 4).

As was expected, the addition of the SBR reduces slightly the number of distance computations and the search time, but GR reduces them drastically (to less than one half).

A second set of experiments were carried out in order to compare the MDF method of tree construction to the original *c*-means method (using the FNR) and to the incremental strategy proposed by Kalantari and Mc-Donalds (KM).

Figure 5 illustrates the average number of distance computations and the search time using the *c*-means, KM and MDF tree construction methods. It can be observed that the MDF reduces to less than one half the number of distance computation and the search time.

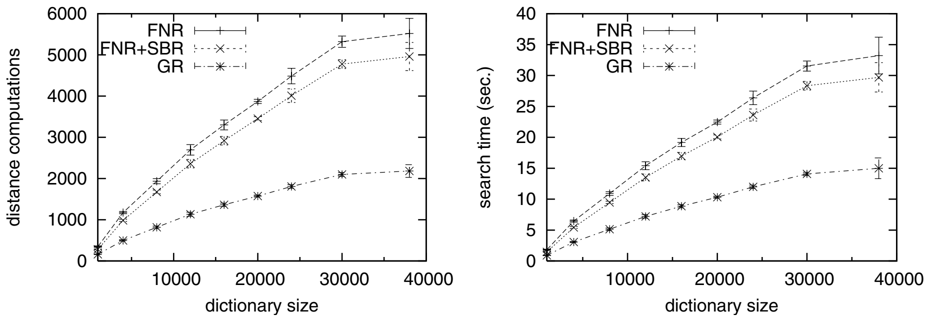

Fig. 4. Comparison of FNR, FNR+SBR and GR elimination rules using the c-medians tree construction.

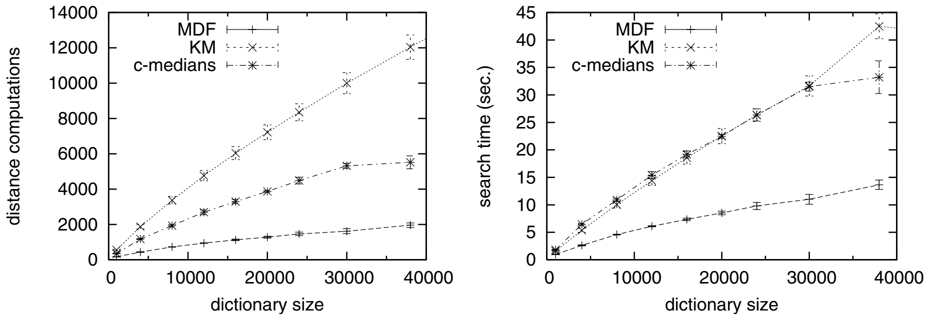

Fig. 5. Comparison of *c*-medians, KM and MDF methods to build the tree using the FNR elimination rule.

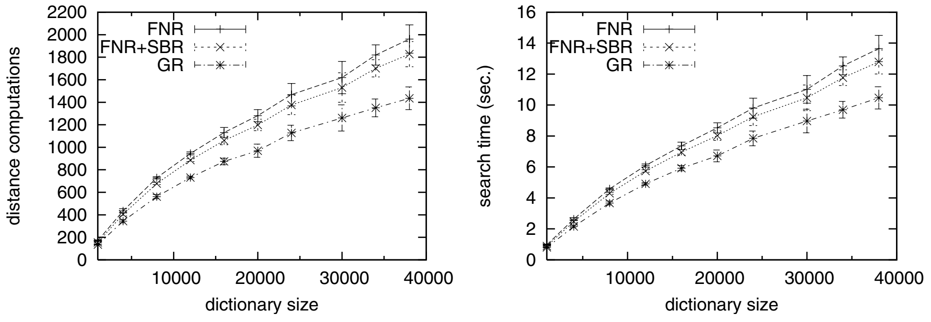

Fig. 6. Comparison of FNR, FNR+SBR and GR using a tree constructed with MDF.

Once stated that the MDF method is superior that the *c*-means method, a third set of experiments were carried out in order to study the contribution of FNR, FNR+SGR and GR with a tree constructed following MDF method.

As figure 6 shows, the number of distance computations and the search time decrease using the new rules although now the reductions are not so impressive that in the previous cases.

Nevertheless, comparing the distance computations and the search time of the original algorithm (fig. 4, FNR) to the algorithm using GR and MDF (fig. 6,GR), it can be observed that applying both techniques at the same time the distance computations and the search time can be reduced one third.

6 Conclusions

In this work two improvements of the Fukunaga and Narendra fast nearest neighbour algorithm was tested in a spelling correction task.

The first improvement is a new method to build the decision tree used in the FN algorithm. On one hand, to build the tree with this method it is much faster than with the original one and, on the other hand, the use of this method reduces the number of distance computations and the search time to one half in our experiments. The use of the KM way to build the tree increases the number of distance computations even more than with the original method. The second modification is the introduction of two new elimination rules. The use of the GR rule reduces to one half the number of distance computations and the search time. Both improvements can be applied together reaching reductions to one third in the distance computations and the search time.

On this work the generalised elimination rule was implemented in a quite naive way by means of an iterative procedure. Now we are interested in implementing this rule using a more adequate algorithm to obtain further search time reductions.

We believe that these approximations can be extended to other nearest neighbour search algorithms based on a tree structure.

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. 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
7. Alinat, P.: Periodic progress report 4, ROARS project ESPRIT II - Number 5516. Thomson Technical Report TS ASM 93/S/EGS/NC/079 (1993)
8. 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).
9. Geofrey I. Webb: OPUS: An Efficient Admissible Algorithm for Unordered Search. Journal of Artificial Intelligence Research 3 (1995) 431-465.

10. Kamgar-Parsi, B., Kanal, L.: An improved branch and bound algorithm for computing k-nearest neighbors. Pattern Recognition Letters (1985) **3** 7–12
11. E. Gómez-Ballester, L. Micó, J. Oncina: Some improvements in tree based nearest neighbour algorithms. Progress in Pattern Recognition, Speech and Image Analysis. Proceedings of the 8th Iberoamerican Congress on Pattern Recognition. Havana, Cuba. Springer Verlag, Lecture notes in Artificial Intelligence, pp. 456–46 (2003)