Fast *k* Most Similar Neighbor Classifier for Mixed Data Based on a Tree Structure

Selene Hernández-Rodríguez, J. Fco. Martínez-Trinidad, and J. Ariel Carrasco-Ochoa

Computer Science Department
National Institute of Astrophysics, Optics and Electronics
Luis Enrique Erro No. 1, Sta. María Tonantzintla, Puebla, CP: 72840, México
{selehdez,ariel,fmartine}@inaoep.mx

Abstract. In this work, a fast *k* most similar neighbor (*k-MSN*) classifier for mixed data is presented. The *k* nearest neighbor (*k-NN*) classifier has been a widely used nonparametric technique in Pattern Recognition. Many fast *k-NN* classifiers have been developed to be applied on numerical object descriptions, most of them based on metric properties to avoid object comparisons. However, in some sciences as Medicine, Geology, Sociology, etc., objects are usually described by numerical and non numerical features (mixed data). In this case, we can not assume the comparison function satisfies metric properties. Therefore, our classifier is based on search algorithms suitable for mixed data and non-metric comparison functions. Some experiments and a comparison against other two fast *k-NN* methods, using standard databases, are presented.

Keywords: Nearest Neighbors Rule, Fast k-Most Similar Neighbors Search, Mixed Data.

1 Introduction

The *k-NN* [1] rule has been a widely used nonparametric technique in Pattern Recognition. However, in some applications, the exhaustive comparison between the new object to classify and the objects in *T* becomes impractical. Many fast *k-NN* classifiers have been designed to avoid this problem; most of them were designed for numerical object descriptions and based on metric properties.

In some applications, the objects are described by numerical and non numerical features (mixed data). In this case, we can not assume the comparison function satisfies metric properties and therefore, we can not use most of the methods proposed for numerical objects. Thus, if a metric is not available but a comparison function that evaluates the similarity between a pair of objects could be defined, the objective would be to find the *k* most similar neighbors (*k-MSN*) and use them for classifying.

The *k-MSN* classifier is based on a training set *T* of *N* objects. Each object is described by *d* features, which can be numerical or non numerical. Given a new object *Q* to classify, the goal consists in finding the *k-MSN* according to a comparison function

L. Rueda, D. Mery, and J. Kittler (Eds.): CIARP 2007, LNCS 4756, pp. 407–416, 2007.

and assigning to Q the majority class of its k nearest neighbors. The exhaustive search of the *k-MSN*, as occurs with the *k-NN*, could be very expensive if T is large. Therefore, in this paper we introduce a fast *k-MSN* classifier based on a tree structure, which does not assume the comparison function satisfies any metric property.

This paper is organized as follows: Section 2 provides a brief review of fast *k-NN* classifiers based on tree structures. In Section 2.1 the comparison functions used in this work are described. In Section 3 our fast *k-MSN* classifier is introduced. In Section 4 we report experimental results obtained using our classifier. Finally, in Section 5 we present some conclusions and future work.

2 Related Work

In this section we describe some methods for solving the fast nearest neighbor classification problem.

To avoid the exhaustive search, some methods, based on tree structures, have been proposed. In a preprocessing step, the objects in T are organized in a tree structure. In the classification step, the tree is traversed to find the *k-NN*. The speed up is obtained while the exploration of some parts of the tree is avoided. One of the first fast *k-NN* classifiers, that uses a tree structure, was proposed by Fukunaga and Narendra [2].

In the classifier proposed by Fukunaga and Narendra, each node p of the tree contains four features, which are: the set of objects in the node p (S_p), the number of objects in p (N_p), the centre of the node (M_p) and finally the maximum distance between the centre of p and the objects in the node p (R_p). Given a new object Q to classify, Fukunaga's fast classifier searches the *NN* based on a branch and bound method to traverse the tree. Two pruning rules are used to decide whether a node or an object of the tree is evaluated or not. These rules are based on the triangle inequality. The first pruning rule for nodes of the tree is:

$$B + R_p < D(Q, M_p) \tag{1}$$

Where B is the distance between Q and the current *NN* and D is the distance function. The second pruning rule is applied to the objects that belong to a leaf node of the tree, in order to decide whether or not to compute the distance from the sample Q to the object from the node or not. The pruning rule for every object $o_i \in S_p$ is:

$$B + D(o_i, M_p) < D(Q, M_p) \tag{2}$$

The objects that satisfy condition (2) can not be closer than the current nearest neighbour and therefore, the distance to Q is not computed. The search process finishes when all nodes in the tree have been evaluated or eliminated by the first pruning rule. Finally, the class of the *NN* found in the search process is assigned to Q. An extension to *k-NN* is also proposed in [2], where, in the search process, B is the distance between Q and the current *k-NN* instead of the current *1-NN*. In this case, the majority class of its k nearest neighbors is assigned to Q.

In the last years, some methods have been developed to improve the Fukunaga and Narendra classifier [3,4,5,6,7,8]. The improvements proposed in [3,4,5,6,8] are exact methods to find the *k-NN*. However, finding the *k-NN* (even using a fast method) is a

slow process for some tasks, therefore in [7] Moreno-Seco proposed a fast approximate *k-NN* classifier, where Fukunaga's first pruning rule is modified in order to finish the search when the current nearest neighbor is not too far from the exact nearest neighbor:

$$(1+e)(D(Q,M_p)-R_p)>B \tag{3}$$

Where e is an error margin that allows to decrease the number of comparisons. In this process, lower classification run times are obtained, but reducing the classification accuracy. However, Moreno-Seco's classifier also relies on metric properties to avoid comparisons.

All the mentioned methods, based on tree structures, were designed to work with numerical data when the object comparison function satisfies metric properties.

2.1 Comparison Functions for Mixed Data

In this work, in order to compare objects, the function F [10] was used. Let us consider a set of objects $\{O_1, O_2, \ldots, O_N\}$, each of them described by d attributes $\{x_1, x_2, \ldots, x_d\}$. Each feature could be numerical or non numerical. The function F is defined as follows:

$$F(O_1,O_2)=1-\frac{|\{x_i \mid C_i(x_i(O_1),x_i(O_2))=1\}|}{d} \tag{4}$$

For qualitative data $C_i(x_i(O_1), x_i(O_2))$ is defined as follows:

$$C_i(x_i(O_1),x_i(O_2))=\begin{cases}1 & If\ x_i(O_1)=x_i(O_2)\ and\ neither\ x_i(O_1)\ nor\ x_i(O_2)\\ & \quad is\ a\ missing\ value\\ 0 & otherwise\end{cases} \tag{5}$$

For quantitative data $C_i(x_i(O_1), x_i(O_2))$ is defined as follows:

$$C_i(x_i(O_1),x_i(O_2))=\begin{cases}1 & If\ |x_i(O_1)-x_i(O_2)|<\sigma_i\ and\ neither\ x_i(O_1)\ nor\ x_i(O_2)\\ & \quad is\ a\ missing\ value\\ 0 & otherwise\end{cases} \tag{6}$$

Where, σ_i is the standard deviation of the attributes x_i. We also used the functions: *HVDM* and *HOEM*, described in [11] and [12] respectively, which allow us to compare objects described by mixed data. Using the functions F, *HVDM* and *HOEM* the most similar neighbor is the one that minimizes the function.

3 Proposed Classifier

In this section, an approximate fast *k-MSN* classifier which considers object described by mixed data is introduced. The classifier consists of two phases. The first one, or preprocessing phase, is the construction of a tree structure from T, using suitable strategies for mixed data. In the second phase, two fast approximate *k-MSN* search algorithms are used, which are independent of metric properties of the comparison function.

3.1 Preprocessing Phase

In this phase, the training set is hierarchically decomposed to create a tree structure. The *C-Means with Similarity Functions* algorithm *(CMSF)*, which is an extension of the *C-Means* algorithm to work with mixed data, is used. In the original *C-Means* the mean of the objects is considered as the centre of the cluster, meanwhile in *CMSF* an object, which represents the cluster, is used as the centre of it (see [10] for details).

The node 0 (root) of the tree contains the whole set *T*. In order to create the following levels of the tree, each node *n* of the tree is divided in *C* clusters (c_1,... , c_C), in such way that each cluster c_i represents a descendant node n_i of *n*. Each node is divided again and this process is repeated until a stop criterion is satisfied. In figure 1 the algorithm to construct the tree is described.

Each node *p* of the tree contains three features which are: the set of objects that belong to that node S_p, the number of objects in that node N_p and unlike Fukunaga's and Moreno-Seco's methods a representative object of the node Rep_p.

```
1.   CurrentNode = 0
2.   NodesToDivide = CurrentNode
3.   p = 1
4.   While |NodesToDivide| ≠ 0
5.        CurrentNode = NodesToDivide [1]
6.        To divide the objects belonging from CurrentNode in C clusters
7.        For every cluster c =1:C
8.             cluster(c) = node_p of the tree
9.             node_p is a child of CurrentNode
10.            Compute the features of node_p: S_p, N_p, Rep_p
11.            If a Stop Criterion is satisfied, then node_p is a leaf
12.            Else, node_p is added to the list NodesToDivide
13.            p = p+1
14.       End for every
15.       Eliminate CurrentNode from NodesToDivide
16.  End while
```

Fig. 1. Tree building algorithm

Three different stop criteria are proposed:

1. *Stop criterion based on the node size* (*SCNS*): in this criterion, if a node contains less than *SC* objects ($N_p \leq SC$), then the node is considered as a leaf. This criterion is used in some other works [3,4 and 7]. To implement this criterion, lines 7-14 of the algorithm described in figure 1, are replaced by the algorithm described in figure 2.

```
For every cluster c =1:C
    cluster(c) = node_p of the tree,    node_p is a child of CurrentNode
    Compute the features of node_p: S_p, N_p, Rep_p
    If (N_n ≤ CP) then node_p is a leaf,
    Else node_p is added to the list NodesToDivide
    p = p+1
End for every
```

Fig. 2. Stop criterion based on the node size (SCNS)

2. *Stop criterion based on non homogeneous nodes* (*SCNHN*): it is proposed to consider the number of objects in a cluster and the class of those objects. In this case, if certain percentage of the objects have the same class, then the node is considered a

leaf and is marked with the majority class (even if $N_p > SC$). To implement this criterion, lines 7-14 of the algorithm described in figure 1, are replaced by the algorithm described in figure 3. When the node is generalized by the majority class, through *SCNHN*, an error is introduced. Therefore, a third criterion is proposed.

3. *Stop criterion based on homogeneous nodes* (*SCHN*): In this case, if certain percentage of the objects in a cluster belongs to the same class, two nodes are created. Using the objects that belong to the majority class, a leaf node is created and is marked with the majority class. The rest of the objects are assigned to a second node, which is considered as a leaf if $N_p \leq SC$, otherwise, this node will be divided again.

A pseudocode of this criterion is shown in figure 4.

```
For every cluster c =1:C
    cluster(c) = nodep of the tree,      nodep is a child of CurrentNode
        Compute the features of nodep: Sp, Np, Repp
    For every j = 1: No.Classes
        PercC (j) = Percentage of the objects in nodep that belong to the class j
    End for every
    PercM = max (PercC (j), ∀ j ∈ [1, No. Classes] )
    If (PercM ≥ thres), then nodep is a leaf,   NodeCriterion(n) = true
        ClassNode (n) = the majority class of nodep
    Else If (Nn ≤ CP) then nodep is a leaf,   NodeCriterion(n) = false
        Else nodep is added to the list NodesToDivide
    p = p+1
End for every
```

Fig. 3. Stop criterion based on non homogeneous leave (SCNHL)

```
For every cluster c =1:C
    For every j = 1: No.Classes
        Percc (j) = Percentage of the objects in nodep that belong to the class j
    End for every
    PercM = max (Percc (j),
    If (PercM ≥ Thres),
        N1 = the set of objects of cluster c that belong to the majority class
        N2 = the rest of the objects
        With N1:
            Sp = N1, nodep is a leaf,      nodep is a child of CurrentNode
            Compute the features of nodep: Np, Repp
            NodeCriterion(n) = true,   ClassNode (n) = the majority class of nodep
        p = p+1
        With N2:
            Sp = N2, nodep is a child of CurrentNode
            Compute the features of nodep: Np, Repp
            If (Nn ≤ CP) then nodep is a leaf,   NodeCriterion(n) = false
            Else nodep is added to the list NodesToDivide
    Else, if (Nn ≤ CP) then nodep is a leaf,   NodeCriterion(n) = false
        Else nodep is added to the list NodesToDivide
    End if
    p = p+1
End for every
```

Fig. 4. Stop criterion based on homogeneous leave (SCHL)

3.2 Classification Phase

In this phase, in order to avoid the exhaustive tree traversal, the existing fast *k-NN* classifiers rely on pruning rules (based on metric properties). As we are looking a method applicable when the comparison function does not satisfy metric properties,

the pruning rules proposed by Fukunaga and Moreno-Seco would not be appropriate; therefore we propose to stop the search when a leaf of the tree is reached. In the first method (*k-MSN* local search) we propose to use a depth-first search and in the second method (*k-MSN* global search) we propose to use a best-first search. The proposed methods for searching the *k-MSN* are described below:

k-MSN local search method: It begins at the root of the tree, following the path of the most similar node and finishes on a leaf of the tree. A list of the *k-MSN* is stored and updated during the tree traversal. When a leaf node *l* is reached, if *l* is marked with the majority class, then only the representative object Rep_l is considered to update the *k-MSN*. In other case, a local exhaustive search in the node is done and the list of *k-MSN* is finally updated. (see pseudocode in figure 5).

Current Node = node 0
While *Current Node* ≠ Leaf
  DesNodes = descendant nodes of the *Current Node*.
  $MostSimilarNode = \min_{\forall\ node_p \in DesNodes} (F(Q, Rep_p))$
  Current Node = *MostSimilarNode* Update *k-MSN* considering *MostSimilarNode*
End while
If *NodeCriterion*(*Current Node*) = true,
  $sim = F(Q, Rep_{CurrentNode})$ Update *k-MSN* considering $Rep_{CurrentNod}$
Else, perform an exhaustive search with the objects in the *Current Node* and update *k-MSN*
Class(*Q*)=majority class of the *k-MSN*

Fig. 5. *k-MSN* local search method

When a leaf is reached, if the list does not have *k MSN*'s, then the tree traversal makes backtracking to explore nodes closer to *Q*, in order to find *k MSN*'s.

k-MSN global search method: It begins at the root of the tree, comparing *Q* against the descendant nodes of the root, which are added to a list (*List_tree_traversal*). After that, *List_tree_traversal* is sorted in such way that the most similar node to *Q* is in the first place. The most similar node is eliminated from *List_tree_traversal* and its descendant nodes are compared against *Q*, and added to *List_tree_traversal*, which is sorted again. In this search it is possible to reconsider nodes in levels of the tree already traversed if the first node of *List_tree_traversal* belongs to a previous level in the tree (see pseudocode in figure 6). During the tree traversal, another list (*List_k-MSN*) containing the *k* current *MSN* is stored and updated.

Current Node = node 0
List_tree_traversal = {0}
While *Current Node* ≠ Leaf
  Current Node = *List_tree_traversal* [1] Delete *Current Node* from *List_tree_traversal*
  DesNodes = descendant nodes of the *Current Node*.
  Compute the similarity between *Q* and the nodes in *DesNodes*
  Add the nodes to *List_tree_traversal* Update *List_k-MSN*
  Order *List* in such way that the most similar object to *Q* is the first element of the list.
End while
If *NodeCriterion*(*Current Node*) = true,
  $sim = F(Q, Rep_{CurrentNode})$ Update *List_k-MSN*, considering $Rep_{CurrentNod}$
Else, perform an exhaustive search with the objects in the *Current Node* and update *k-MSN*
Class(*Q*)=majority class of the *k-MSN*

Fig. 6. *k-MSN* global search method

When a leaf is reached, if *List_k-MSN* does not contain *k-MSN*, then the first element in *List_tree_traversal* is considered to follow a new route in the tree traversal. The process stops when *List_k-MSN* contains *k-MSN*.

However, using both search methods (*k-MSN* global and *k-MSN* local), it is quite difficult not to have *k MSN*'s during the tree traversal, in applications where the training set is large.

After finding *k-MSN*, the majority class is assigned to the new sample *Q*.

4 Experimental Results

In this section, we report the obtained results of applying the proposed fast approximate *MSN* classifier over 9 datasets from the *UCI* repository [15]. Three of these datasets are numerical (Glass, Iris and Wine), two are non numerical (Hayes and Bridges) and four are mixed (Hepatitis, Zoo, Flag, Echocardiogram). In all the experiments 10-fold-cross-validation was used.

In order to compare our method, Fukunaga's [2] and Moreno-Seco's [7] classifiers were adapted. The adaptation consists in the use of the same tree structure proposed in Section 3 and the same function suitable to work with mixed data, instead of a distance function. There are other methods based in tree structures [13, 14]. However, it is not possible to adapt these methods to work with similarity functions because these methods involve techniques such as *PCA* which are only applicable to numerical data.

The compared five fast *k-NN (k-MSN)* classifiers used in the experimentation were:

1. The exhaustive *k-NN* classifier (using a dissimilarity function)
2. Adapted Fukunaga's *k-NN* classifier
3. Adapted Moreno-Seco's *k-NN* classifier
4. The proposed classifier using *k-MSN* local search
5. The proposed classifier using *k-MSN* global search

To compare the classifiers, the accuracy and the number of comparisons between objects were considered.

Before using adapted Moreno-Seco's classifier, some tests with different values of the error margin *e* (*e*=1, 10 and 20) were proved. These experiments were carried out for the three stop criteria described in Section 3.1: *SCNS* (with *SC*=1, 5 and 20), *SCNHL* (with percentage threshold=50, 70 and 100) and *SCHL* (with percentage threshold=50, 70 and 100), were proved. In every experiment, different values of *C* (*C*=3, 5, no. of classes, 4* no. of classes, (.1)* no. of objects and (.3)* no. of objects) in the *CMFS* algorithm, were also proved. In the classification phase, all of the experiments were repeated using *k* = 1, 3 and 5 *MSN*. During the tree construction and the classification phase, the three dissimilarity functions: *HVDM, F* and *HOEM* were tested.

In all of these experiments, while the error margin *e* grows, the number of object comparisons decreases. In the next experiments, adapted Moreno-Seco's classifier was used with *e*=20, because this value needed the smallest number of comparisons with a slightly accuracy reduction.

The parameter *C* of the *CMSF* algorithm corresponds to the number of branches of the nodes in the tree. In order to select the best value of *C*, *C*=3, 5, no. of classes, 4* no. of classes, (.1)* no. of objects and (.3)* no. of objects, were proved. In the tree

construction phase, the three stop criteria described in Section 3.1: *SCNS* (with *SC* = 1, 5 and 20), *SCNHL* (with percentage threshold=50, 70 and 100) and *SCHL* (with percentage threshold=50, 70 and 100), were proved. In the classification phase, all of the experiments mentioned before were repeated using k = 1, 3 and 5 *MSN*. During the tree construction and the classification phase, all experiments mentioned before, the three dissimilarity functions: *HVDM, F* and *HOEM* were tested.

According to these experiments, the accuracy does not vary too much with the different values of *C*. However, the number of comparisons between objects increases for the adapted Fukunaga's classifier when *C* grows. In the next experiments *C*=3 was used, because there is not a big variation of the accuracy and the number of objects comparisons is reduced for adapted Fukunaga's and Moreno-Seco's classifiers.

Another important parameter in the tree construction algorithm is the stop criterion. To evaluate the stop criterion, all datasets mentioned before were used. In this experiment, the *HVDM* function and 1-*MSN* were considered.

First, the performance of the classifiers, using the first stop criterion (*SCNS*), was evaluated according different values of the *SC* parameter. From this experiments, *SC*=20 was chosen, since the biggest objects comparison percentage reduction is achieved.

The performance of the classifiers, using the criterion based on non homogeneous nodes (*SCNHN*) was evaluated with a percentage threshold equal to 50, 70, 80 and

Table 1. Evaluation of the different classifiers using *HVDM* function, with k = 1, 3 and 5 *MSN*

General averages	Exhaustive *k-NN* classifier		Adapted Fukunaga's *k-NN* classifier		Adapted Moreno-Seco's *k-NN* classifier		Proposed classifier using *k-MSN* local search		Proposed classifier using *k-MSN* global search	
	%Acc	%Com	%Acc	%Com	%Acc	%Com	%Acc	%Com	%Acc	%Com
k=1										
Hep	82,67	100	81,33	126,58	80,71	88,59	82,67	15,75	82,63	13,52
Zoo	97,09	100	94,18	68,25	94,18	23,90	93,09	12,46	93,09	16,76
Flag	53,05	100	50,39	50,38	50,39	44,12	47,84	10,56	48,37	17,41
Echo	83,41	100	81,10	110,29	81,81	75,29	84,89	13,54	84,89	18,50
Hayes	85,71	100	85,25	56,06	84,95	23,25	85,71	14,85	85,71	12,97
Brid	61,82	100	56,09	68,20	54,00	51,55	55,09	9,54	55,09	15,46
Glass	70,06	100	70,06	41,03	67,29	37,80	66,30	7,65	66,30	6,16
Iris	94,67	100	94,67	24,95	92,67	21,37	94,00	14,55	94,00	11,17
Wine	95,49	100	95,49	40,00	95,49	30,73	94,41	11,05	94,41	7,93
General Avg.	**80,44**	**100**	**78,73**	**65,08**	**77,94**	**44,07**	**78,22**	**12,22**	**78,28**	**13,32**
k=3										
Hep	82,67	100	81,29	126,58	81,33	88,59	82,67	15,75	83,92	13,52
Zoo	96,09	100	94,09	68,25	91,09	23,90	95,09	12,46	95,09	16,76
Flag	53,53	100	53,53	50,38	53,53	44,12	50,95	10,56	51,47	17,41
Echo	82,64	100	79,62	110,29	79,56	75,29	80,38	13,54	80,38	18,50
Hayes	85,71	100	85,71	56,06	84,95	23,25	85,71	14,85	85,71	12,97
Brid	57,64	100	56,91	68,20	53,91	51,55	55,91	9,54	55,91	15,46
Glass	68,70	100	68,70	41,03	68,11	37,80	68,66	7,65	68,66	6,16
Iris	94,00	100	94,00	24,95	93,33	21,37	94,00	14,55	94,67	11,17
Wine	95,52	100	95,52	40,00	95,46	30,73	94,41	11,05	93,86	7,93
General Avg.	**79,61**	**100**	**78,82**	**65,08**	**77,92**	**44,07**	**78,64**	**12,22**	**78,85**	**13,32**
k=5										
Hep	84,54	100	83,25	126,58	85,25	88,59	78,83	15,75	78,17	13,52
Zoo	94,09	100	90,18	68,25	88,18	23,90	92,09	12,46	92,09	16,76
Flag	53,66	100	53,63	50,38	53,55	44,12	52,58	10,56	53,11	17,41
Echo	86,43	100	85,66	110,29	85,60	75,29	86,43	13,54	86,43	18,50
Hayes	86,43	100	86,43	56,06	84,89	23,25	85,77	14,85	85,77	12,97
Brid	60,09	100	58,09	68,20	56,09	51,55	56,18	9,54	56,27	15,46
Glass	63,55	100	63,55	41,03	62,64	37,80	63,53	7,65	63,53	6,16
Iris	95,33	100	95,33	24,95	94,00	21,37	94,67	14,55	94,67	12,52
Wine	97,19	100	97,19	40,00	93,82	30,73	94,41	11,05	94,41	7,93
General Avg.	**80,15**	**100**	**79,26**	**65,08**	**78,22**	**44,07**	**78,28**	**12,22**	**78,27**	**13,47**

100. The accuracy is slightly reduced using *SCNHN* (with percentage threshold=50, 70 and 80) in comparison with the criterion based on the node size (*SCNS*, with *SC*=20). However, the comparisons percentage is reduced using *SCNHN*. Using *SCNHN* and *SCHN*, when the percentage threshold grows, both, the accuracy and the objects comparisons percentage increase.

When the percentage threshold is 100, using *SCHN* and *SCNHN* the obtained accuracy and the comparisons percentage are the same, because in both cases, a leaf is marked with the majority class only when all of the objects in the node belong to the same class. In the next experiments, *SCHN* with percentage threshold =100 (which is the same as using *SCNHN*, with percentage threshold =100) and *SC*=20, were used.

In table 1, the obtained accuracy (**%Acc**) and the percentage of objects comparisons (**%Com**) are shown, for *k*=1,3, 5 in *k-MSN* process. The number of comparisons performed by exhaustive search is considered as the 100 percent of comparisons. Using *k*=1, the higher accuracy in average is obtained with the exhaustive *NN* classifier.

The experiments with different *k-MSN* were repeated using *F* and *HOEM* functions. The performance of the different classifiers is similar for *F* and *HOEM* functions. Using *HOEM* function, Fukunaga's classifier is an exact method, it happens because *HOEM* function is a metric. However, for all classifiers the lower accuracy is obtained using *HOEM* function.

In figure 7 a graphic of the accuracy against the comparisons percentage using the different fast *NN* (*MSN*) classifiers, with *k*=1, is shown. From this graph, we can see that all the classifiers obtained similar accuracy but the classifiers proposed in this work (using *MSN* local and *MSN* global search) did the smaller number of comparisons.

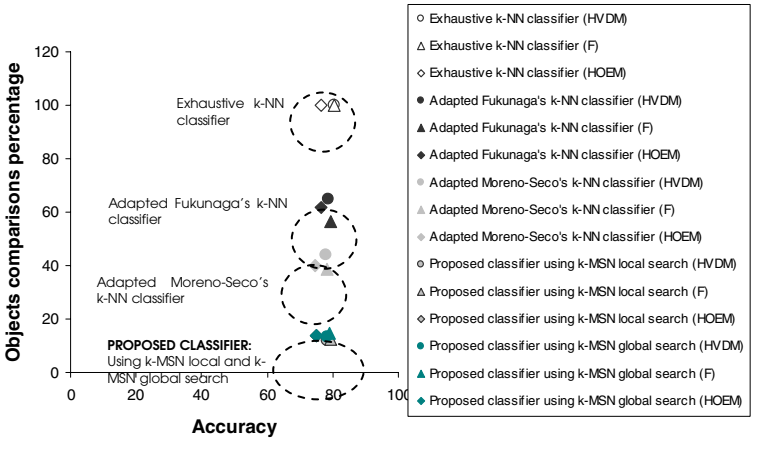

Fig. 7. Accuracy against the comparisons percentage using the different classifiers and three different comparison functions

5 Conclusions

In practical problems, it is frequent to find non numerical object descriptions or even mixed data (numerical and non numerical). Therefore, it is important to use methods that allow us to work with these kind of features.

In this work, an approximated fast *k-MSN* classifier for mixed data was proposed. In order to compare our method, Fukunaga's and Moreno-Seco's classifiers were implemented using the same object comparison functions for mixed data. Based on our experimental results, in comparison with Fukunaga's and Moreno-Seco's classifiers, our method (using *MSN* local and *MSN* global search), obtained a big reduction on the number of comparisons between objects with only a slightly accuracy reduction.

As future work, we plan to look for a pruning rule non based on metric properties, which allow us to reduce the number of objects comparisons, but doing an exhaustive tree traversal during the search process.

References

1. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. Trans. Information Theory 13, 21–27 (1967)
2. Fukunaga, K., Narendra, P.: A branch and bound algorithm for computing k-nearest neighbors. IEEE Trans. Comput. 24, 743–750 (1975)
3. Kalantari, I., McDonald, G.: A data structure and an algorithm for the nearest point problem. IEEE Trans. Software Eng. 9, 631–634 (1983)
4. Omachi, S., Aso, H.: A fast algorithm for a k-nn Classifier based on branch and bound method and computational quantity estimation. Systems and Computers in Japan 31(6), 1–9 (2000)
5. Gómez-Ballester, E., Mico, L., Oncina, J.: Some Improvements in Tree Based Nearest Neighbor Search Algorithms. In: Sanfeliu, A., Ruiz-Shulcloper, J. (eds.) CIARP 2003. LNCS, vol. 2905, pp. 456–463. Springer, Heidelberg (2003)
6. Gómez-Ballester, E., Mico, L., Oncina, J.: Some approaches to improve tree-based nearest neighbor search algorithms. Pattern Recognition Letters 39, 171–179 (2006)
7. Moreno-Seco, F., Mico, L., Oncina, J.: Approximate Nearest Neighbor Search with the Fukunaga and Narendra Algorithm and its Application to Chromosome Classification. In: Sanfeliu, A., Ruiz-Shulcloper, J. (eds.) CIARP 2003. LNCS, vol. 2905, pp. 322–328. Springer, Heidelberg (2003)
8. Mico, L., Oncina, J., Carrasco, R.: A fast Branch and Bound nearest neighbor classifier in metric spaces. Pattern Recognition Letters 17, 731–739 (1996)
9. MacQueen, J.B.: Some Methods for classification and Analysis of Multivariate Observations. In: Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297. University of California Press, Berkeley (1967)
10. García-Serrano, J.R., Martínez-Trinidad, J.F.: Extension to C-Means Algorithm for the use of Similarity Functions. In: 3rd European Conference on Principles and Practice of Knowledge Discovery in Database Proceedings, Prague, Czech, pp. 354–359 (1999)
11. Wilson, D.R., Martínez, T.: Reduction techniques for instance based learning algorithms. Machine Learning. 38, 257–286 (2000)
12. Wilson, D., Martínez, T.: Improve heterogeneous Distance Functions. Journal of Artificial Intelligence Research 6, 1–34 (1997)
13. McNames, J.: A Fast Nearest Neighbor Algorithm Based on a Principal Axis Search Tree. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(9), 964–976 (2001)
14. Yong-Sheng, C., Yi-Ping, H., Chiou-Shann, F.: Fast and versatile algorithm for nearest neighbor search based on lower bound tree. Pattern Recognition Letters (2006)
15. Blake, C., Merz, C.: UCI Repository of machine learning databases. In: Department of Information and Computer Science, University of California, Irvine, CA (1998), http://www.uci.edu/mlearn/databases/