“Resumen de Tesis Doctoral”
Fast Most Similar Neighbor (MSN) classifiers for Mixed Data
Clasificadores Rápidos basados en el algoritmo del Vecino más Similar (MSN) para Datos Mezclados

Selene Hernández Rodríguez(Graduated)**, José Fco. Martínez Trinidad**(Advisor)**,**
Jesús Ariel Carrasco Ochoa(Advisor)
National Institute of Astrophysics, Optics and Electronics
Luis Enrique Erro # 1, Santa María Tonantzintla, C.P. 72840, Puebla, México.
selehdez@ccc.inaoep.mx, fmartine@inaoep.mx, ariel@inaoep.mx

Abstract. The *k* nearest neighbor (*k-NN*) classifier has been extensively used in Pattern Recognition because of its simplicity and its good performance. However, in large datasets applications, the exhaustive *k-NN* classifier becomes impractical. Therefore, many *fast k-NN classifiers* have been developed; most of them rely on metric properties (usually the triangle inequality) to reduce the number of prototype comparisons. Hence, the existing fast *k-NN* classifiers are applicable only when the comparison function is a metric (commonly for numerical data). However, in some sciences such as Medicine, Geology, Sociology, etc., the prototypes are usually described by qualitative and quantitative features (mixed data). In these cases, the comparison function does not necessarily satisfy metric properties. For this reason, it is important to develop fast *k* most similar neighbor (*k-MSN*) classifiers for mixed data, which use non metric comparisons functions. In this thesis, four fast *k-MSN* classifiers, following the most successful approaches, are proposed. The experiments over different datasets show that the proposed classifiers significantly reduce the number of prototype comparisons.

Keywords: Nearest neighbor rule, fast nearest neighbor search, mixed data, non-metric comparison functions.

Resumen. El clasificador *k* vecinos más cercanos (*k-NN*) ha sido ampliamente utilizado dentro del Reconocimiento de Patrones debido a su simplicidad y buen funcionamiento. Sin embargo, en aplicaciones en las cuales el conjunto de entrenamiento es muy grande, la comparación exhaustiva que realiza *k-NN* se vuelve inaplicable. Por esta razón, se han desarrollado diversos clasificadores rápidos *k-NN*; la mayoría de los cuales se basan en propiedades métricas (en particular la desigualdad triangular) para reducir el número de comparaciones entre prototipos. Por lo cual, los clasificadores rápidos *k-NN* existentes son aplicables solamente cuando la función de comparación es una métrica (usualmente con datos numéricos). Sin embargo, en algunas ciencias como la Medicina, Geociencias, Sociología, etc., los prototipos generalmente están descritos por atributos numéricos y no numéricos (datos mezclados). En estos casos, la función de comparación no siempre cumple propiedades métricas. Por esta razón, es importante desarrollar clasificadores rápidos basados en la búsqueda de los *k* vecinos más similares (*k-MSN*) para datos mezclados que usen funciones de comparación no métricas. En esta tesis, se proponen cuatro clasificadores rápidos *k-MSN*, siguiendo los enfoques más exitosos. Los experimentos con diferentes bases de datos muestran que los clasificadores propuestos reducen significativamente el número de comparaciones entre prototipos.

Palabras clave: Regla del vecino más cercano, búsqueda rápida del vecino más cercano, datos mezclados, funciones de comparación no métricas

1 Introduction

The *k-NN* classifier (Cover & Hart, 1967) uses a training set (*T*) of prototypes, whose class is known *a priori*. To decide the class of a new prototype, the *k-NN* classifier performs an exhaustive comparison between the prototype to classify and the prototypes in the training set, assigning to the new prototype a class, according to the classes of its *k* nearest neighbors in *T*. However, when the training set is large, the exhaustive comparison is expensive and sometimes inapplicable. Thus, many fast *k-NN* classifiers have been designed; different reviews appear in (Nene & Nayar, 1997; Ramasubramanian *et al.*, 2000; and Yong-Sheng *et al.*, 2007).

The objective of a fast *k-NN* classifier is to reduce the number of comparisons trying to keep the classification accuracy obtained by *k-NN*. Speeding up the *k-NN* classifier is required because some applications demand a rapid response on large datasets, for example online stock analysis, air traffic control, network traffic management, intrusion detection, etc. Also, fast *k-NN* classifiers are useful for problems with high dimensionality where the comparison function could be very expensive (Mico *et al*., 1994; Denny & Franklin, 2006), under this context, reducing the number of comparisons could be very important. For these reasons, although nowadays the computers are very fast, the development of fast *k-NN* classifiers is currently an active research area (Adler & Heeringa, 2008; Panigrahi, 2008). Nevertheless, most of the fast *k-NN* classifiers proposed in the literature have been designed for numerical prototype descriptions compared through a metric function. Moreover, in some sciences such as Medicine, Geology, Sociology, etc., the prototypes are usually described by numerical and non numerical features (mixed data) and the comparison function does not satisfy metric properties.

Thus, if a metric is not available but a comparison function that evaluates the similarity between a pair of prototypes can be defined, given a new prototype *Q* to classify, the objective is to find the *k* most similar neighbors to *Q* in a training set *T* (with *N* prototypes, where each prototype is described by *d* attributes, which can be numerical or non numerical), and assign to *Q* a class (based on its *k* most similar neighbours). However, the exhaustive search of the *k-MSN*, as occurs with *k-NN*, could be very expensive if *T* is large. For this reason, it is important to develop fast *k* most similar neighbor (*k-MSN*) classifiers for mixed data and non metric comparisons functions.

In this thesis, four fast *k-MSN* classifiers are proposed. The first uses a tree structure, the second and the third are based on a new Approximating-Eliminating approach for mixed data. Finally, the last fast *k-MSN* classifier proposed in this thesis uses a tree structure and an Approximating-Eliminating approach.

2 Proposed fast k-MSN classifiers

According to the strategy used to avoid prototype comparisons, fast *k-NN* classifiers can be broadly divided as:

Tree based classifiers, which hierarchically decompose the training set in a tree and use a tree traversal search algorithm to find the *k-NN* elements, while pruning rules to avoid the exhaustive tree traversal are used.

Approximating-Eliminating classifiers, which start comparing the new prototype to classify against the prototypes in the training set, supporting on some eliminating criteria to avoid the comparison against every element in the training set.

Hybrid classifiers, which use both approaches.

In this thesis, some fast *k-MSN* classifiers following these lines are proposed, in order to work with mixed data.

2.1 Tree k-MSN

The first proposed classifier, Tree k-MSN (Hernández-Rodríguez-a *et al.,* 2007; Hernández-Rodríguez-b *et al.*, 2007), consists of two phases. The first one, or preprocessing phase, builds a tree structure from the training set (*T*). In the second phase, two search algorithms, which are independent of metric properties of the comparison function, are proposed for classifying a new prototype.

Preprocessing phase: in this phase, the training set is hierarchically decomposed to create a tree structure (*TS*). At the beginning, the root of the tree contains the whole training set. In order to create the following levels of the tree, each node *n* of the tree is divided in *C* clusters, in such a way that each cluster represents a descendant node of *n*. Each descendant node is divided again and this process is repeated until a stop criterion is satisfied.

Since our algorithm is designed to allow mixed data, instead of using the C-Means algorithm for building the tree structure, as in the FN classifier, the C-Means with Similarity Functions algorithm (*CMSF*) (García-Serrano & Martínez-Trinidad, 1999), is used. *CMSF* allows creating *C* clusters and computing as representative element of each cluster a prototype belonging to the cluster (i.e., a prototype contained in *T*); besides *CMSF* allows using any similarity function.

Each node *p* of the tree contains three features: S_p the set of prototypes that belong to *p*; N_p the number of prototypes in *p* and unlike FN and MS classifiers, Rep_p a representative prototype of the node, which is on average the most similar to the rest of prototypes in the node.

A node is marked as a leaf when a stop criterion is satisfied. In this thesis we used a stop criterion based on the node size (*SC1*), which is used in (Fukunaga & Narendra, 1975; Kalantari & McDonald, 1983; Mico *et al.*, 1996; Omachi & Aso, 2000; McNames, 2001; D'Haes *et al.*, 2002; Gomez-Ballester *et al.*, 2006) and we introduce two new stop criteria (*SC2* and *SC3*), which take into account not only the number of prototypes of the node, but also the class distribution of these prototypes. The three stop criteria are the following:

1. Stop criterion 1 (SC1). This criterion is based on the node size. According to this criterion, if the number of prototypes contained in a node is less than a predefined threshold ($N_p \leq NoP$), then the node is marked as a leaf. The objective of this criterion is to obtain leaves with a few prototypes.

However, when most of the prototypes contained in a node belong to the same class, dividing this node could lead to unnecessary prototype comparisons during the classification stage, between the prototype to classify and the representative prototypes of the nodes. Because all descendant nodes, that would be created, also would belong to the same class. Since the objective is to classify a new prototype trying to avoid prototype comparisons, we propose a second stop criterion during the tree construction:

2. Stop criterion 2 (SC2). If most of the prototypes in a node belong to the same class, then the node is considered as a leaf and it is marked with the majority class, even if the set is not small enough according to the first stop criterion ($N_p > NoP$). In order to decide how many prototypes in the node must belong to the same class, for generalizing the class of a node, a percentage threshold (*PercThres*) is used. In the nodes where this criterion is not satisfied, only the size of the node is considered to create leaf nodes (*SC1*).

When the node is generalized by the majority class, through *SC2*, if *PercThres*=100%, it means that all prototypes in the node belong to the same class (the generalized class of the node). However, when *PercThres*<100%, an error is introduced, because some prototypes in the node do not belong to the majority class. Therefore, we introduce a third criterion:

3. Stop criterion 3 (SC3). If certain percentage (*PercThres*) of the prototypes in a node belongs to the same class, two nodes are created. Using the prototypes that belong to the majority class, a leaf node is created and it is marked with the majority class. The rest of the prototypes are assigned to a second node. In the second node, the size is considered to decide if the node is a leaf (if $N_p \leq NoP$) or if the node will be divided again. In the nodes where *SC3* criterion is not satisfied, only the size of the node is considered to create leaf nodes (*SC1*).

Using *SC2* and *SC3* the number of prototype comparisons (during the classification stage) is reduced, because if during the tree traversal a leaf node (marked with the majority class) is reached, then only the representative prototype of the node, with the corresponding majority class, is used to update the list of the *k* most similar neighbors (only one comparison), instead of comparing the prototype to classify against all the prototypes contained in the leaf.

Classification phase: in this phase, in order to avoid an exhaustive tree traversal, fast *k-NN* classifiers rely on pruning rules (based on metric properties). As we are looking for a method applicable when the comparison function does not satisfy metric properties, pruning rules based on the triangular inequality cannot be used; therefore, we propose to stop the search when a leaf of the tree is reached. In the first search algorithm (*DF* search), we propose to use a depth-first search strategy and in the second search algorithm (*BF* search), we propose to use a best-first search strategy. The two proposed algorithms for searching the *k-MSN* are described below:

1. DF search: It begins at the root of the tree, following the path of the most similar node and finishes when a leaf is reached. As each node of the tree is represented by a prototype of the training set, with known class, a list of the *k-MSN* is stored and updated during the tree traversal. When the first leaf node *l* is reached, if *l* is marked with the majority class, then only the representative prototype Rep_l is used to update the *k-MSN* (because most of the prototypes in the node belong to the same class). If the node is not marked with the majority class, then an exhaustive search in the node is done and the list of *k-MSN* is updated. After a leaf is processed, if the list of *k-MSN* does not have *k* elements, then the tree traversal follows backtracking steps to explore nodes closer to *Q*, until *k* most similar neighbours are found.

2. BF search: 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 a way the most similar node to *Q* is in the first place. The most similar node (first element) is eliminated from *List_tree_traversal* and its descendant nodes are compared against *Q*, and added to *List_tree_traversal*, which is sorted again. The search finishes when the first element of *List_tree_traversal* is a leaf. 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.

During the tree traversal, another list (*List_k-MSN*) containing the *k* current *MSN* is stored and updated.

After a leaf is processed (in a similar way than in the local search), if *List_k-MSN* does not contain *k* elements (*MSN*), then the first element in *List_tree_traversal* is considered to follow a new route. The process stops when *List_k-MSN* contains *k* elements (*MSN*). However, using both search strategies (*DF* and *BF*), in practical problems where the training set is large, it is quite difficult that *List_k-MSN* does not have *k* elements (*MSN*) when the first leaf is reached. After finding *k-MSN*, the majority class is assigned to the new sample *Q*.

2.2 AEMD

The second fast *k-MSN* classifier proposed in this thesis, AEMD (Hernandez-Rodríguez-c *et al.*, 2008), is based on a new Approximating-Eliminating approach for Mixed Data. AEMD also consists of two phases: preprocessing and classification.

*Preprocessing phase:*in this stage, AEMD computes and stores the next information which is used during the classification phase to reduce the number of comparisons between prototypes:

1. Similarity binary array (SimArray). In this thesis, we proposed computing and storing an array of similarities per attribute among the prototypes in the training set (*T*), where *SimArray*[P_a,P_b,x_i]=1 if the prototypes P_a and P_b are similar regarding the attribute x_i, $i\in[1,d]$ and otherwise *SimArray*[P_a,P_b,x_i]=0, $P_a,P_b\in T$. In order to evaluate the similarity per attribute between two prototypes, different approaches can be applied. In this thesis, the following criteria were used:

(1)

If the attribute x_i is not numeric:

(2)

If the attribute x_i is numeric:

(3)

Where σ_i is the standard deviation of the attribute x_i in *T*. The required space to store *SimArray* is |*T*| * |*T*| * *d*, but each element is a bit, therefore |*T*| * |*T*| words of *d* bits are needed for storing *SimArray*.

2. Similarity threshold (SimThres). This value is used during the tree traversal algorithm to decide if a representative prototype of a node of the tree can be used to prune nodes in the tree. In this thesis *SimThres* is computed as follows: let Set_c be the set of prototypes that belong to class *c* (*c*=1,…, number of classes in *T*) and *ClassAvgSim* be defined as follows:

(4)

SimThres is computed as the average value of similarity for all the classes:

$$SimThres = average(ClassAvgSim(c)) \qquad (5)$$

3. A representative prototype per class (RP_c*).* Taking advantage of the class information, we propose to use a representative prototype (RP_c) for each class in the training set. These prototypes are used to obtain a first approximation of the *k* most similar neighbors during the classification phase, before performing *TS* tree traversal algorithm. In this thesis, to compute RP_c, let Set_c be the set of prototypes that belong to class *c*. For each P_i in Set_c:

(6)

Where *Sim* is a similarity comparison function. Thus, the representative prototype for each class is the one that maximizes *ASim* function:

(7)

Where *i*=1...|Set_c|, *c*=1,...,*NoClasses* and *NoClasses* is the number of classes in *T*.

Classification phase (Fig. 1)*:* given a new prototype *Q* to classify, *SM*, RP_c and *SimThres* (computed during the preprocessing phase) are used to avoid prototype comparisons. This phase is based on Approximating-Eliminating steps for mixed data, which are not based on the triangle inequality. This stage is as follows:

Initial approximating step. At the beginning of the algorithm, the prototype *Q* is compared against the class representative prototypes per class (RP_c), to obtain a first approximation to the *k* most similar neighbors and, in particular, the current most similar neighbor ($Current_{MSN}$). After that, all RP_c are eliminated from *T*.

If $Sim(Q,Current_{MSN}) \geq$ *SimThres*, then the prototype $Current_{MSN}$ is used to eliminate prototypes from *T* (*Eliminating step*). In other case, the *Approximating step* is performed.

Eliminating step. $Current_{MSN}$ is used to eliminate prototypes from *T*. First, a binary representation (*BR*) containing the similarity per attribute (using $C_i(X_i(P_a),X_i(P_b))$ criterion), between *Q* and $Current_{MSN}$ is used. Thus, $BR_i(Q,Current_{MSN})=1$, if *Q* and $Current_{MSN}$ are similar in the attribute x_i and $BR_i(Q,Current_{MSN})=0$, in other case.

Using *BR*, those prototypes in *T*, which are not similar to $Current_{MSN}$ at least, in the same attributes in which $Current_{MSN}$ is similar to *Q*, are eliminated from *T* (using $SimArray(Current_{MSN},P_a)$).

After the *Initial approximation* and the *Eliminating steps*, if *T* is not empty, the approximation step is performed.

Approximating step. In this step, a new prototype $MSN' \in T$ is randomly selected, compared against *Q*, eliminated from *T* and used to update the current *k* most similar neighbors. If $Sim(Q,MSN') < SimThres$, a new *MSN'* is randomly selected (*Approximating step*). Otherwise, if $Sim(Q,MSN') \geq SimThres$, the prototype *MSN'* is used to eliminate prototypes from *T* (*Eliminating step*).

This process is repeated until the set *T* is empty. After finding the *k* most similar neighbors, the majority class is assigned to *Q*.

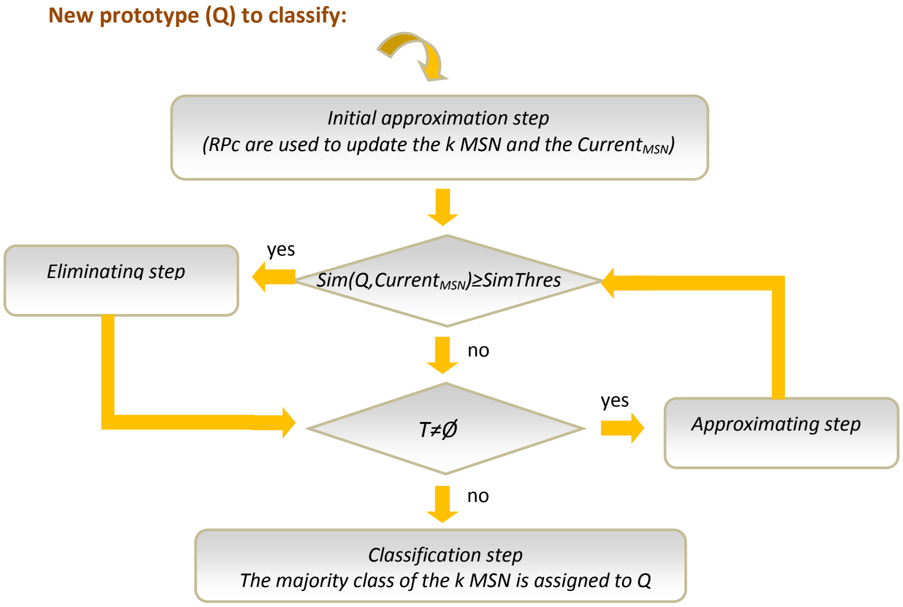
New prototype (Q) to classify:
Initial approximation step
(RPc are used to update the k MSN and the Current$_{MSN}$)
yes
Eliminating step
Sim(Q,Current$_{MSN}$)≥SimThres
no
T≠Ø
yes
Approximating step
no
Classification step
The majority class of the k MSN is assigned to Q

Fig. 1. Diagram of the classification stage of AEMD

2.3 LAEMD

The third fast *k-MSN* classifier LAEMD is a modification of AEMD, which aims to reduce the storage space required by AEMD, following the ideas of LAESA (Mico et al., 1994). In the preprocessing phase, LAEMD selects a subset of base prototypes (*BP*) from *T*, and the distances between the prototypes in *T* and the prototypes in *BP* are computed and stored in *SimArray*, which is smaller than *SimArray* (used in AEDM), since $|BP|<<|T|$. In LAESA, the *BP* selection consists in finding the farthest prototype in average to the remaining prototypes (not yet selected as *BP*) and this process is repeated until a predefined number (*m*) of *BP* have been selected.

Since, an important step for the performance of LAESA classifiers is the *BP* selection algorithm, in this thesis two *BP* selection algorithms (Hernández-Rodríguez-d *et al.*, 2008) are introduced:

1. BP selection using class information (BPClass). In this algorithm, taking advantage of the class information, the *BP* set is created by selecting roughly the same number of elements from each class, in order to obtain a balanced subset. To select the prototypes for a class, such prototypes which are the most similar, on average, to the rest of the prototypes from the same class are selected; this process is repeated for each class to select the *BP* set.

2. BP selection using representative prototypes of the tree TS (BPNodesTS). In this case, the TS tree structure is used to select some prototypes from the training set. The set of base prototypes (*BP*) is composed by the representative prototypes of the nodes of *TS* tree. In this case, if the number of nodes in the tree is bigger than *m*, then only the representative prototypes of the first *m* nodes, found by a breath search, are selected. Breadth search is used, in order to consider the nodes from the first levels of the tree, since the nodes in the first levels of the tree represent more prototypes than nodes in deeper levels.

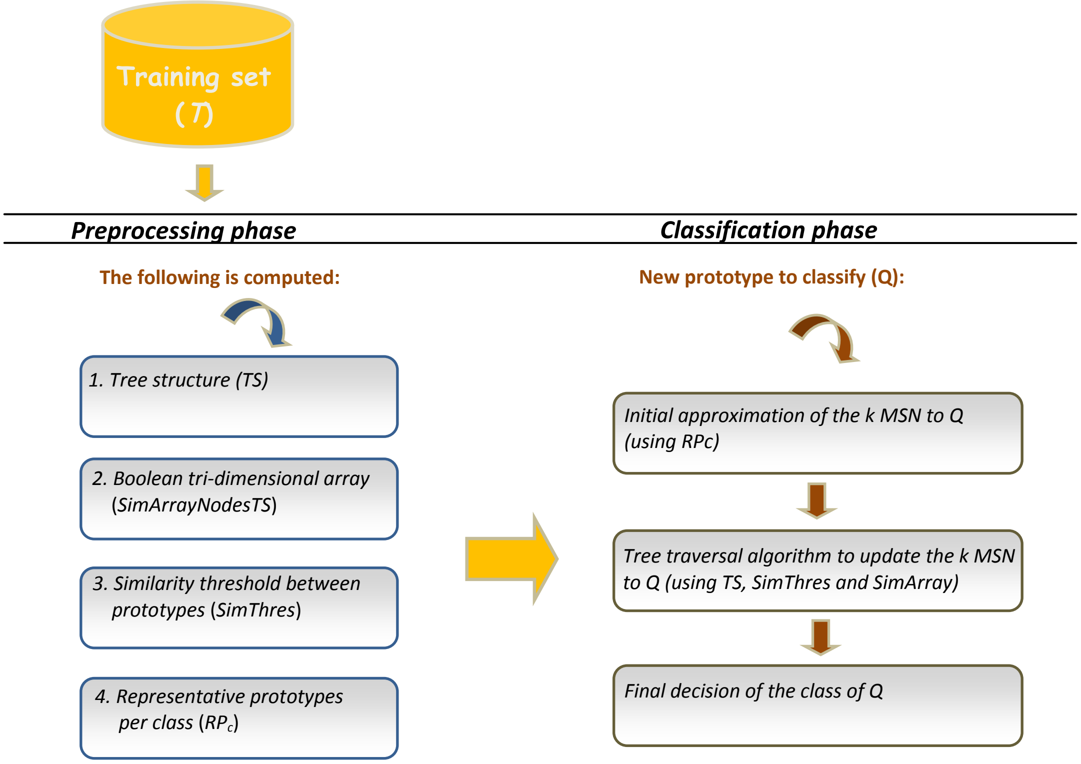
Training set
(T)
Preprocessing phase
Classification phase
The following is computed:
1. Tree structure (TS)
2. Boolean tri-dimensional array (SimArrayNodesTS)
3. Similarity threshold between prototypes (SimThres)
4. Representative prototypes per class (RPc)
New prototype to classify (Q):
Initial approximation of the k MSN to Q (using RPc)
Tree traversal algorithm to update the k MSN to Q (using TS, SimThres and SimArray)
Final decision of the class of Q

Fig. 2. General diagram of Tree LAEMD classifier

2.4 Tree LAEMD

Tree LAEMD (Hernández-Rodríguez-e *et al.*, 2008) is based on a hybrid approach, which uses a tree structure and new Approximating-Eliminating steps, for mixed data and any non metric comparison function (Fig. 2).

Preprocessing phase: in this phase we proposed to compute and store a tree structure, a Boolean tri-dimensional array, a representative prototype per class and a similarity threshold, as follows:

1. Tree structure (TS), described in Section 2.1.

2. Boolean tri-dimensional array (SimArrayNodesTS). This array stores the similarity between the representative prototypes of the nodes in *TS* tree. This array is smaller than the one used in AESA, because the number of nodes in *TS* tree is smaller than the number of elements in the training set. Besides, this array is used during the classification phase to prune nodes during the tree traversal.

In this case, $SimArrayNodesTS[Rep_a,Rep_b,x_i]$=1, if the representative prototypes Rep_a and Rep_b (of the nodes *a* and *b*) are similar regarding the attribute x_i (i=1,…,d, where *d* is the number of attributes in the prototypes) and otherwise *SimArrayNodesTS*[Rep_a,Rep_b,x_i]=0. In order to evaluate the similarity per attribute between two prototypes, the criteria described in Section 2.2, were used.

3. Similarity threshold between prototypes (SimThres). This value is used during the tree traversal algorithm to decide if a representative prototype of a node of the tree *TS* can be used to prune nodes in the tree and it is computed as described in Section 4.1.

4. A representative prototype per class (*RP_c*). These values are computed as described in Section 2.2.

Classification phase: given a new query *Q* to classify *TS*, *SimThres*, *SimArrayNodesTS*, and *RP_c*, computed during the preprocessing phase, are used to avoid prototype comparison, as follows:

1. Initial approximating step. In this step, the representative prototypes per class (*RP_c*) are compared against *Q* to obtain a first approximation of the *k* most similar neighbors.

2. Tree traversal step. In order to update the *k* most similar neighbors, two algorithms to traverse the tree (Approximating step) are proposed:

* Depth First Search, *DFSAE*

* Best First Search, *BFSAE*

During the tree traversal algorithm, if a representative prototype (*Rep*) of a node is similar enough to *Q* (*Sim(Q,Rep)>SimThres*), then all nodes whose representative prototypes are dissimilar to *Rep* are pruned, using the information about the similarity between representative prototypes stored in *SimArrayNodesTS* (Eliminating step).

3. Classification step. Finally, the majority class of the *k MSN* is assigned to *Q*.

3 Experimental result section

In order to evaluate the performance of the proposed classifiers (Tree k-MSN, AEMD, LAEMD and Tree LAEMD), they are compared against the exhaustive *k-NN* algorithm (Cover & Hart, 1967) and the following tree-based fast *k-NN* classifiers:

1. Adapted FN classifier (Fukunaga & Narendra, 1975)
2. Adapted GB classifier using GR pruning rule (Gómez-Ballester *et al.*, 2006)
3. Adapted ONC classifier (Oncina *et al.*, 2007)
4. Adapted MS classifier (Moreno-Seco *et al.*, 2003)
5. Cluster tree (Zhang & Srihari, 2004)

To compare FN, GB, ONC and MS classifiers with our proposed fast *k-MSN* classifier, we adapted these classifiers. The adaptation consisted in the use of the same tree structure (*TS*) proposed in Section 3.1 and the same function, suitable to work with mixed data, instead of a distance function. In this way, only the search algorithm of the fast *k-NN* classifiers, is compared.

Besides, since GB tree traversal search algorithm was proposed for a binary tree, in our GR adaptation the pruning rule is applied to all of the *C*-1 sibling nodes. When a leaf node is reached, as it could contain more than one prototype, a local exhaustive comparison is performed to find the *k-MSN*.

Since Cluster tree is proposed to work with any dissimilarity, we use this classifier with the same comparison functions for mixed data.

Also, the following Approximating-Eliminating-approach fast *k-NN* classifiers were compared:

1. AESA classifier (Vidal, 1986)
2. LAESA classifier (Mico *et al.*, 1994), using *m*=20% of the prototypes in the dataset
3. iAESA classifier (Figueroa *et al.*, 2006)
4. Probabilistic iAESA classifier (Figueroa *et al.*, 2006)

Finally, the following fast *k-NN* classifiers based on the hybrid-approach were also considered:

1. TLAESA (Mico *et al.*, 1996)
2. Modified TLAESA (Tokoro, 2006)

To compare Approximating-Eliminating and hybrid-approaches, the same comparison function for mixed data was used. In this work, the dissimilarity function *HVDM* (Wilson & Martínez, 2000), was used for the experiments. This comparison function was selected because it allows comparing mixed data and it is not a metric function since it does not satisfy the triangle inequality property.

For the experiments, 10 datasets from the UCI repository (Blake & Merz, 1998) were used (see Table 1).

In order to compare the different classifiers, the accuracy (*Acc*) and the percentage of comparisons between prototypes (*Comp*), were considered. The accuracy was computed as follows:

$$\frac{\qquad}{\qquad} \tag{8}$$

Where, *NoCorrectPrototypes* is the number of correctly classified prototypes in the testing set and *NoTestPrototypes* is the size of the testing set. The percentage of comparisons between prototypes was computed as follows:

(9)

Where *NoCompFastClassifier* is the number of comparisons done by the fast *k-NN* classifier, and *NoTrainingPrototypes* is the size of the training set. According to (10), the exhaustive classifier does the 100% of the comparisons.

Table 1. Datasets used in this section

Dataset	No. of prototypes	No. of numerical features	No. of non numerical features	Classes	Missing data
Hepatitis	155	6	13	2	yes
Zoo	101	1	16	7	no
Flag	194	3	25	8	no
Echocardiogram	132	9	2	2	yes
Hayes	132	0	4	3	no
Soybean-large	307	0	35	19	yes
Bridges	108	0	11	7	yes
Glass	214	9	0	7	no
Iris	150	4	0	3	no
Wine	178	13	0	3	no

Table 2. Classification accuracy (*Acc*) and percentage of comparisons (*Comp*), obtained by the exhaustive *k-NN* search and the tree-based fast *k-NN* classifiers, using *HVDM* function and *k*=1 *MSN*

Datasets	k-NN		Adapted FN classifier		Adapted GB classifier		Adapted ONC classifier		Adapted MS classifier		Cluster tree	
	Acc	*Com*	*Acc*	*Comp*	*Acc*	*Comp*	*Acc*	*Comp*	*Acc*	*Comp*	*Acc*	*Comp*
Hepatitis	81,75	100	81,13	118,99	81,13	87,50	81,13	71,89	81,08	95,86	77,88	42,01
Zoo	97,00	100	97,00	24,68	97,00	23,46	97,00	22,16	97,00	21,86	95,00	41,63
Flag	53,21	100	53,71	56,97	53,71	53,10	53,18	46,52	54,26	43,43	48,47	44,09
Echocard.	82,69	100	82,69	121,01	82,69	84,01	82,69	72,82	84,18	93,43	83,24	41,75
Hayes	84,29	100	84,29	28,31	84,29	21,14	84,29	16,59	83,57	27,45	67,31	27,45
Soybean-L	90,54	100	91,18	26,56	91,18	19,90	91,18	16,85	89,88	25,02	83,33	20,22
Bridges	63,36	100	64,27	93,65	64,27	53,34	65,18	50,65	63,27	54,25	40,27	36,82
Glass	68,18	100	68,18	35,96	68,18	27,04	68,18	20,16	67,71	34,36	60,30	34,64
Iris	94,67	100	94,67	21,12	94,67	19,87	94,67	18,21	95,33	19,37	86,67	37,41
Wine	95,46	100	95,46	43,78	95,46	32,66	95,46	27,88	94,35	34,31	93,24	45,85
Avg.	**81,12**	**100**	**81,26**	**57,10**	**81,26**	**42,20**	**81,30**	**36,37**	**81,06**	**44,93**	**73,57**	**37,18**

In all the experiments ten-fold-cross-validation was used. According to this technique, the dataset is divided in ten partitions; nine of them are used for training and the last partition is used as testing set. This process is repeated ten times, in such a way that each partition is used once as testing set.

In (Hernandez-Rodriguez et al., 2007) different experiments for choosing a value of the parameter *C* and *PercThres* were done. In our experiments, *C*=3, *NoP*=10% of the dataset, and *PercThres*=100% were used, since in (Hernandez-Rodriguez et al., 2007), the fast *k-NN* classifiers reached their best results with these values.

In table 2, the results (*Acc* and *Comp*) obtained with the different tree-based fast *k-NN* classifiers, are shown. From this table, we can notice that the adapted FN, GB and ONC become approximate methods (the classification accuracy is not the same as using the exhaustive *k-NN*) using *HVDM* function, which occurs because this comparison function does not satisfy the triangle inequality property. From table 2, we can also notice that Cluster tree, which is the only fast *k-NN* classifier in the state of the art, proposed to work with non metric functions, reduced more the number of prototype comparisons required to classify a new query (from 100% to 37.18%). However, the classification accuracy was decreased from 81.12% (obtained by *k-NN*) to 73.57%. From all the tree-based fast *k-NN* classifiers, ONC obtained the best results (i.e. classification accuracy did not decrease and the percentage of comparisons was reduced from 100% to 36.37%)

In table 3, the results (*Acc* and *Comp*) obtained with different Approximating-Eliminating and Hybrid approaches for fast *k-NN* classifiers, are shown. From this table, we can notice that AESA, LAESA, iAESA, TLAESA and modified TLAESA also become approximate methods, using *HVDM* function. From table 3, it can also be observed that probabilistic iAESA reduced more the number of prototype comparisons required to classify a new query (from 100%, done by the *k-NN* to 22.45%).

In table 4, the results obtained with the fast *k-MSN* classifiers proposed in this work: Tree k-MSN (using *DF* search to traverse the tree), AEMD, LAEMD (using the algorithm *BPNodesTS* to select base prototypes), Tree LAEMD (using the *DFSAE* tree traversal algorithm), are presented. From this table, it is possible to notice that all the proposed classifiers obtained similar classification accuracy than all the other evaluated methods (enlisted in tables 3 and 4) but achieved a biggest reduction in the number of prototype comparisons. Among the proposed classifiers, Tree LAEMD obtained the best results. Additionally, a t-student test (Dietterich, 1998) with a confidence level of 95%, was done. From this test, we noticed that the classification accuracy difference between the proposed classifiers and all other evaluated fast *k-NN* classifiers is not statistically significant, while the prototype comparison reduction (done by Tree k-MSN, AEMD, LAEMD and Tree LAEMD) is statistically significant.

Table 3. Classification accuracy (*Acc*) and percentage of comparisons (*Comp*), obtained by Approximating-Eliminating and Hybrid-approach fast *k-NN* classifiers, using *HVDM* function and *k*=1 *MSN*

Datasets	AESA		LAESA		TLAESA		Modified TLAESA		iAESA		Probabilistic iAESA	
	Acc	Com	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp
Hepatitis	81,68	51,03	80,61	60,73	81,64	84,64	81,03	68,26	81,64	49,37	80,29	37,85
Zoo	97,00	21,34	96,00	25,23	95,75	55,77	95,78	27,34	97,20	21,04	95,42	18,56
Flag	53,60	27,23	52,82	27,57	52,01	49,25	50,19	42,94	52,53	27,15	52,00	25,93
Echocard.	82,54	64,34	82,08	67,39	82,25	75,84	82,12	38,30	82,92	63,18	82,34	63,04
Hayes	83,71	21,23	80,71	21,84	80,73	49,44	80,48	25,49	82,31	21,02	81,84	20,62
Soybean-L	89,87	2,07	89,87	5,12	89,87	38,44	87,23	18,35	89,03	2,05	90,23	2,04
Bridges	63,21	24,23	60,37	26,23	59,37	48,45	59,49	38,92	60,64	24,71	60,60	24,57
Glass	68,18	13,20	68,18	24,53	68,18	49,54	68,18	22,39	68,18	11,92	67,32	11,25
Iris	94,67	8,23	94,67	10,68	94,67	42,54	94,67	13,29	94,67	8,05	94,00	8,01
Wine	95,46	14,23	95,46	14,75	95,46	36,45	95,46	13,52	95,46	13,58	95,41	12,58
Avg.	**80,99**	**24,71**	**80,08**	**28,41**	**79,99**	**53,04**	**79,46**	**30,88**	**80,46**	**24,21**	**79,95**	**22,45**

Table 4. Classification accuracy (*Acc*) and percentage of comparisons (*Comp*), obtained by the proposed fast *k-MSN* classifiers, using *HVDM* function and *k*=1 *MSN*

Datasets	Tree k-MSN		AEDM		LAEDM		Tree LAEDM	
	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp
Hepatitis	83,71	9,54	81,31	14,63	81,64	18,96	81,59	13,23
Zoo	96,00	19,68	97,10	18,61	97,00	32,85	96,00	12,79
Flag	52,21	13,20	53,63	16,23	52,00	17,44	52,47	9,73
Echocard.	79,62	16,50	82,62	17,51	82,62	21,16	81,49	13,05
Hayes	83,52	18,19	83,85	14,52	83,85	18,42	82,17	10,67
Soybean	85,26	9,72	90,54	11,52	90,54	16,25	89,37	7,83
Bridges	60,36	15,80	61,85	17,62	60,64	17,96	60,60	8,32
Glass	67,73	12,91	67,01	14,99	68,18	15,72	68,18	8,38
Iris	92,67	15,66	94,09	14,82	94,09	15,62	94,67	10,37
Wine	91,57	13,80	95,00	14,62	95,00	18,51	95,46	12,07
Avg.	**79,27**	**14,50**	**80,70**	**15,50**	**80,56**	**19,29**	**80,20**	**10,64**

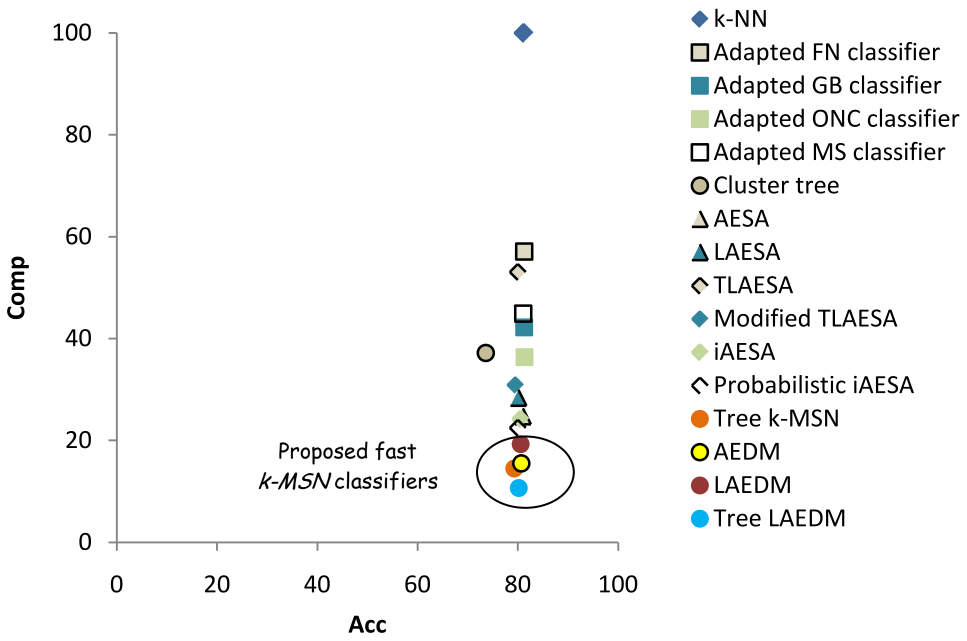

Fig. 3. Classification accuracy (*Acc*) against percentage of comparisons (*Comp*) obtained by the different fast *k-NN/k-MSN* classifiers, using *HVDM* similarity function and *k*=1 *MSN*

In Figure 3, a graph of the accuracy (Acc) against the number of prototype comparisons (Comp) is shown. From this graph, we can notice that all the classifiers obtained similar average classification accuracy, except Cluster tree which obtained the lowest classification accuracy results. However, the proposed classifiers (Tree k-MSN, AEMD, LAEMD and Tree LAEMD), did the smallest number of prototype comparisons. All the experiments were repeated, using *k*=3 and *k*=5 and the performance of the fast *k-NN* classifiers were similar.

4 Conclusions

The development of fast *k-NN* classifiers has been an active research area in the last years, but most of these classifiers rely on metric properties to reduce the number of prototype comparisons. Moreover, very few work has been focused on applications where the comparison function does not satisfy metric properties. For this reason, in this thesis some fast *k* most similar neighbor (*k-MSN*) classifiers for mixed data and non metric comparisons functions were proposed. To develop these methods, the most successful approaches from the state of the art were followed.

In order to make comparisons, other tree-based fast *k-NN* classifiers were adapted using our proposed tree structure and the same comparison function, to allow them working on mixed data, because of under these circumstances the original algorithms cannot be applied. Also, other methods based on Approximating-Eliminating and Hybrid approaches were considered for comparisons. In these cases, the original algorithms were tested using the same comparison function for mixed data.

Based on our experimental results, in comparison with the exhaustive classifier, and the other fast *k-NN* classifiers (FN, GB, ONC, MS, AESA, LAESA, iAESA, probabilistic iAESA, TLAESA, modified TLAESA and Cluster tree), the proposed classifiers (Tree k-MSN, AEMD, LAEMD and Tree LAEMD), obtained a big reduction on the number of comparisons between prototypes, which is of particular importance in applications where a fast response is required.

Among the proposed fast *k-MSN* classifiers, using Tree LAEMD the best results were obtained. However, it is important to remark that Tree LAEMD requires more storage space than Tree k-MSN. For this reason, the selection of Tree k-MSN or Tree LAEMD would depend on the size of the particular problem. The proposed classifier LAEMD is applicable when the comparison function is very expensive, because the preprocessimg stage required by LAEMD is faster than the preprocessing stage required by Tree LAEMD.

Finally, we can conclude that for large mixed datasets and non-metric prototype comparison functions, the proposed classifiers are the best option.

As future work, we plan to look for other pruning rules (elimination criteria), not based on metric properties, which would allow us to reduce even more the number of prototype comparisons for AEMD, LAEMD and Tree LAEMD.

References

Adler, M., & Heeringa, B. (2008). Search Space Reductions for Nearest-Neighbor Queries. *Theory and Applications of Models of Computation. Lecture Notes in Computer Science*, 4978, 554-567.

Arya, S., Mount, D., Netanyahu, N., Silverman, R., & Wu, A. (1998). An optimal algorithm for approximate nearest neighbor searching in high dimensions. *Journal of the ACM*, 45(6), 891-923.

Athitsos, V., Alon, J., & Sclaroff, S. (2005). Efficient Nearest Neighbour Classification Using Cascade of Approximate with Similarity Measures. *IEEE Conference on Computer Vision and Pattern Recognition 2005*, Washington, USA, 486-493.

Beckmann, N., Kriegel, H., Schneider, R., & Seeger, B. (1990). The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. *ACM SIGMOD Record 19 (2)*, New Jersey, USA, 322-331.

Blake, C., & Merz, C. (1998). UCI Repository of machine learning databases.
[http://archive.ics.uci.edu/ml/datasets.html], Department of Information and Computer Science, University of California, Irvine, CA, January 2006.

Chávez E., & Navarro G. (2005). A compact space decomposition for effective metric indexing. *Pattern Recognition Letters*, 26(9), 1363-1376.

Cheng, D., Gersho, A., Ramamurthi, B., & Shoham, Y. (1984). Fast search algorithms for vector quantization and pattern matching. *IEEE International Conference on Acoustics, Speech and Signal Processing*, California, USA, 372-375.

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1), 21-27.

Denny, M., & Franklin, M.J. (2006). Operators for Expensive Functions in Continuous Queries. *22nd International Conference on Data Engineering ICDE´06*, Georgia, USA, 147-147.

Dietterich, T. (1998). Statistical Tests for comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7), 1895-1923.

D'haes, W., Dyck, D., and Rodel, X. (2002) PCA-based branch and bound search algorithms for computing k nearest neighbors. *Pattern Recognition Letters*, 24(9-10), 1437-1451.

Figueroa, K., Chávez, E., Navarro, G., and Paredes, R. (2006). On the last cost for proximity searching in metric spaces. *Workshop on Experimental Algorithms*. *Lecture Notes in Computer Science*, 4007, 279-290.

Fredriksson K. (2007). Engineering efficient metric indexes. *Pattern Recognition Letters*, 28(1), 75-84.

Friedman J. H., Bentley J. L., & Finkel R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3), 209-226.

Fukunaga, K., & Narendra, P. (1975). A branch and bound algorithm for computing k-nearest neighbors. *IEEE Transactions on Computers*, C-24(7), 750-753.

García-Serrano, J. R., & Martínez-Trinidad, J. F. (1999). Extension to C-Means Algorithm for the use of Similarity Functions. *European Conference on Principles of Data Mining and Knowledge Discovery*. *Lectures Notes in Artificial Intelligence*, 1704, 354-359.

Goh K., Li B., & Chang E. (2002). DynDex: A Dynamic and Non-metric Space Indexer. *Proceedings of the tenth ACM international conference on Multimedia*, Juan-les-Pins, France, 466-475.

Gómez-Ballester, E., Mico, L., and Oncina, J. (2006). Some approaches to improve tree-based nearest neighbor search algorithms. *Pattern Recognition*, 39(2), 171-179.

Guttman, A. (1984). R-trees: A Dynamic Index Structure for Spatial Searching. *ACM SIGMOD International Conference on Management of Data*, New York, USA, 47-57.

Hernández-Rodríguez, S., Martínez-Trinidad, J., & Carrasco-Ochoa, A. (2007). Fast k Most Similar Neighbor Classifier for Mixed Data Based on a Tree Structure. *Iberoamerican congress on Pattern*

Recognition. Lecture Notes in Computer Science, 4756, 407-416.

Hernández-Rodríguez, S., Martínez-Trinidad, J., & Carrasco-Ochoa, A. (2007). Fast Most Similar Neighbor Classifier for Mixed Data. *The 20th Canadian Conference on Artificial Intelligence. Lecture Notes in Artificial Intelligence*, 4509, 146-158.

Hernández-Rodríguez, S., Martínez-Trinidad, J., & Carrasco-Ochoa, A. (2008). Fast k Most Similar Neighbor Classifier for Mixed Data based on Approximating and Eliminating. *Pacific-Asia Conference on Knowledge Discovery and Data Mining. Lecture Notes in Artificial Intelligence*, 5012, 697-704.

Hernández-Rodríguez, S., Martínez-Trinidad, J., & Carrasco-Ochoa, A. (2008). Fast k Most Similar Neighbor Classifier for Mixed Data based on a Tree Structure and Approximating-Eliminating. 13th *Iberoamerican congress on Pattern Recognition: Progress in Pattern Recognition, Image Analysis and Applications, Lecture Notes in Computer Science*, 5197, 364-371.

Hernández-Rodríguez, S., Martínez-Trinidad, J., & Carrasco-Ochoa, A. (2008). On the Selection of Base Prototypes for LAESA and TLAESA Classifier. 19th *International Conference on Pattern Recognition.* Florida, USA, 407-416.

Hwang W., & Wen K. (2002). Fast kNN classification algorithm based on partial distance search. *Electronics Letters*, 34(21), 2062-2063.

Kalantari, I., & McDonald, G. (1983) A data structure and an algorithm for the nearest point problem. *IEEE Transactions on Software Engineering*, 9(5), 631-634.

Katayama, N., & Satoh, S. (1997). The sr-tree: An index structure for high-dimensional nearest neighbor queries. *ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona, USA, 369-380.

McNames, J. (2001). A Fast Nearest Neighbor Algorithm Based on a Principal Axis Search Tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9), 964-976.

Micó, L., Oncina, J., and Vidal, E. (1994). A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing-time and memory requirements. *Pattern Recognition Letters*, 15(1), 9-17.

Mico, L., Oncina, J., & Carrasco, R. (1996). A fast Branch and Bound nearest neighbor classifier in metric spaces. *Pattern Recognition Letters*, 17(7), 731-739.

Moreno-Seco, F., Mico, L., & Oncina, J. (2003). Approximate Nearest Neighbor Search with the Fukunaga and Narendra Algorithm and its Application to Chromosome Classification. *Iberoamerican congress on Pattern Recognition, Lecture Notes in Computer Science* 2905, 322-328.

Nene, S. A., & Nayar, S. K. (1997). A simple algorithm for nearest neighbour search in high dimensions. *IEEE Transactions in Pattern Analysis and Machine Intelligence*, 19(9), 989-1003.

Omachi, S., & Aso, H. (2000). 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.

Oncina, J., Thollard, F., Gómez-Ballester, E. Micó, L., & Moreno-Seco, F. (2007). A Tabular Pruning Rule in Tree-Based Fast Nearest Neighbor Search Algorithms. *Iberian Conference on Pattern Recognition and Image Analysis. Lecture Notes in Computer Science*, 4478, 306-313.

Panigrahi, R. (2008), An Improved Algorithm Finding Nearest Neighbor Using Kd-trees. *8th Latin American conference on Theoretical informatics. Lecture Notes in Computer Science*, 4957, 387-398.

Ramasubramanian, V., & Paliwal, K. (2000). Fast Nearest-Neighbor Search Algorithms based on Approximation-Elimination search. *Pattern Recognition* 33(9), 1497-1510.

Tokoro, K., Yamaguchi, K., & Masuda, S. (2006). Improvements of TLAESA Nearest Neighbour Search Algorithm and Extension to Approximation Search. *29th Australasian Computer Science Conference*, Hobart, Australia, 48, 77-83.

Uhlmann, J. (1991). Metric trees. *Applied Mathematics Letters*, 4(5), 61-62.

Vidal, E. (1986). An algorithm for finding nearest neighbours in (approximately) constant average time complexity. *Pattern Recognition Letters*, 4(3), 145-157.

White, D., & Jain, R. (1996). Similarity indexing with the ss-tree. *ICDE '96: Twelfth International Conference on Data Engineering*, Washington, USA, 516-523.

Wilson, D., & Martínez, T. (2000). Reduction techniques for instance based learning algorithms. *Machine Learning*, 38, 257-286.

Yianilos, P. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. *SODA '93: Fourth annual ACM-SIAM Symposium on Discrete algorithms*, Philadelphia, USA, 311-321.

Yong-Sheng, C., Yi-Ping, H., & Chiou-Shann, F. (2007). Fast and versatile algorithm for nearest neighbor search based on lower bound tree, *Pattern Recognition Letters*, 40(2), 360-375.

Yunck T. (1976). A technique to identify nearest neighbors. *IEEE Transactions on Systems, Man and Cybernetics*, 6(10), 678-683.

Zhang B., & Srihari S. (2004). Fast k- nearest neighbour classification using cluster-based tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(4), 5

Selene Hernández Rodríguez. *Received her B. S. degree in Computer Science from the Computer Science faculty of the Autonomous University of Puebla (BUAP), Mexico in 2004; her M.Sc. degree in Computer Science from the National Institute of Astrophysics, Optics and Electronics (INAOE), Mexico, in 2006 and her Ph.D. degree in Computer Science from INAOE, Mexico, in 2009. Her research interests are Pattern Recognition, Machine Learning, Data Mining and Supervised Classification.*

José Francisco Martínez Trinidad. *Received his B.S. degree in Computer Science from Physics and Mathematics School of the Autonomous University of Puebla (BUAP), Mexico in 1995, his M.Sc. degree in Computer Science from the faculty of Computers Science of the Autonomous University of Puebla, Mexico in 1997 and his Ph.D. degree in the Center for Computing Research of the National Polytechnic Institute (CIC, IPN), Mexico in 2000. Professor Martinez-Trinidad edited/authored four books and over fifty journal and conference papers, on subjects related to Pattern Recognition.*

Jesús Ariel Carrasco Ochoa. *Received his Ph.D. degree in Computer Science from the Center for Computing Research of the National Polytechnic Institute (CIC-IPN), Mexico, in 2001. Currently, he is a full time researcher at the National Institute for Astrophysics, Optics and Electronics (INAOE) of Mexico. His current research interests include Sensitivity Analysis, Logical Combinatorial Patter Recognition, Testor Theory, Feature Selection, Prototype Selection and Clustering.*