Fast *k* Most Similar Neighbor Classifier for Mixed Data Based on Approximating and Eliminating

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

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. The *k* nearest neighbor (*k-NN*) classifier has been a widely used nonparametric technique in Pattern Recognition. In order to decide the class of a new prototype, the *k-NN* classifier performs an exhaustive comparison between the prototype to classify (query) and the prototypes in the training set *T*. However, when *T* is large, the exhaustive comparison is expensive. To avoid this problem, many fast *k-NN* algorithms have been developed. Some of these algorithms are based on Approximating-Eliminating search. In this case, the Approximating and Eliminating steps rely on the triangle inequality. However, in soft sciences, the prototypes are usually described by qualitative and quantitative features (mixed data), and sometimes the comparison function does not satisfy the triangle inequality. Therefore, in this work, a fast *k* most similar neighbour classifier for mixed data (AEMD) is presented. This classifier consists of two phases. In the first phase, a binary similarity matrix among the prototypes in *T* is stored. In the second phase, new Approximating and Eliminating steps, which are not based on the triangle inequality, are presented. The proposed classifier is compared against other fast *k-NN* algorithms, which are adapted to work with mixed data. Some experiments with real datasets are presented.

Keywords: Nearest Neighbors Rule, Fast Nearest Neighbor Search, Mixed Data, Approximating Eliminating search algorithms.

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 prototype to classify and the prototypes in the training set *T* becomes impractical. Therefore, many fast *k-NN* classifiers have been designed to avoid this problem.

Some of these fast *k-NN* algorithms can be classified as exact methods, because they find the same *NN* that would be found using the exhaustive search. Some other algorithms are approximate methods, because they do not guarantee to find the *NN* to a query prototype among the training set, but they find an approximation faster than the exact methods.

To avoid comparisons between prototypes during the search of the *NN*, different techniques have been developed: Approximating Eliminating algorithms [2-5], Tree-based algorithms [4,6-8]. In particular, in this work, the proposed algorithm is based on an Approximating Eliminating approach.

One of the first approaches that uses approximating and eliminating steps is AESA (Approximating Eliminating Search Algorithm), proposed by Vidal [2]. In a preprocessing phase, this algorithm creates a matrix of distances between the prototypes in the training set. Given a new prototype *Q* to classify; a new candidate is approximated, compared against *Q* and, supported on the triangle inequality, those prototypes that can not be closer that the current *NN* are eliminated from the set *T*. The process finishes when all prototypes in *T* have been compared or eliminated.

Using AESA, good results have been obtained. However, a drawback of AESA is its quadratic memory space requirements. For this reason, in [3] an improvement (LAESA), which requires linear memory space, is proposed (LAESA). LAESA algorithm is focused on reducing the amount of information stored, but this algorithm increases the number of comparisons between prototypes. In [5] an improvement on the Approximation step is proposed, for approximating a better candidate and, therefore reducing the number of comparisons between prototypes even more than AESA.

AESA, LAESA and iAESA are exact methods to find the *k-NN*. However, in [5] a probabilistic approach [9] to find approximate *k NN's* is also proposed. In order to reduce the amount of work, the search is stopped when certain percentage of the database has been evaluated, this method is called Probabilistic iAESA.

In [4] TLAESA algorithm is proposed. In TLAESA, a binary tree and a matrix of distances between the prototypes in *T* and a subset of *T*, are used. In [10] an improvement of TLAESA is presented.

All these methods based on Approximating and Eliminating search, were designed to work with quantitative data when the prototype comparison function satisfies the triangle inequality. However, in soft sciences as Medicine, Geology, Sociology, etc., the prototypes are described by quantitative and qualitative features (mixed data). In these cases, sometimes the comparison function for mixed data does not satisfy the triangle inequality and therefore, we can not use most of the methods proposed for quantitative prototype descriptions. Therefore, in this paper we introduce a fast approximate *k* most similar neighbor (*k-MSN*) classifier for mixed data, based on new Approximating and Eliminating steps, which are not based on the triangle inequality property of the comparison function.

This paper is organized as follows: in Section 2 the comparison function used in this work is described. In Section 3 our fast *k-MSN* classifier (AEMD) is introduced. Finally, we report experimental results (Section 4) and conclusions (Section 5).

2 Comparison Functions for Mixed Data

In this work, in order to compare prototypes described by mixed data, the function *F* [11], which does not fulfil the triangle inequality, was used. Let us consider a set of prototypes $\{P_1, P_2, \ldots, P_N\}$, each of them described by *d* attributes $\{x_1, x_2, \ldots, x_d\}$. Each feature could be quantitative or qualitative. The function *F* is defined as follows:

$$F(P_1,P_2)=1-\frac{|\{x_i \mid C_i(x_i(P_1),x_i(P_2))=1\}|}{d} \tag{1}$$

For qualitative data $C_i(x_i(P_1), x_i(P_2))$ is defined as follows:

$$C_i(x_i(P_1), x_i(P_2)) = \begin{cases} 1 & \text{If } x_i(P_1) = x_i(P_2) \text{ and neither } x_i(P_1) \text{ nor } x_i(P_2) \\ & \quad \text{is a missing value} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

For quantitative data $C_i(x_i(P_1), x_i(P_2))$ is defined as follows:

$$C_i(x_i(P_1), x_i(P_2)) = \begin{cases} 1 & \text{If } | x_i(P_1) - x_i(P_2) | < \sigma_i \text{ and neither } x_i(P_1) \text{ nor } x_i(P_2) \\ & \quad \text{is a missing value} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Where, σ_i is the standard deviation of the attribute x_i. Using the function *F*, the most similar neighbor (*MSN*) of a prototype *P*, is the one that minimizes the function.

3 Proposed Classifier

In this section, an approximate fast *k-MSN* classifier, which considers prototypes described by mixed data, is introduced. The classifier consists of two phases: preprocessing and classification.

3.1 Preprocessing Phase

In this phase, AEDM computes the following:

1. *Similarity Matrix* (*SM*). In this work, we proposed to compute and store an array of similarity per attribute among the prototypes in the training set (*T*), where $SM[P_a,P_b,x_i]=1$ if, according to certain criterion, we can conclude that the prototypes P_a and P_b are similar considering the attribute x_i and $SM[P_a,P_b,x_i]=0$, in other case; $a,b \in [1,N]$ and $i \in [1,d]$ (see figure 1). In this work, the similarity criterion described in Section 2, was used.

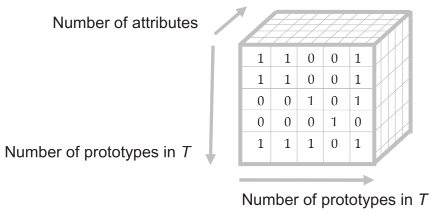

Fig. 1. *SM* matrix

The required space to store *SM* matrix is *N* x *N* x *d* but each element is a bit, therefore, the needed space is *N* x *N* words of *d* bits.

2. *A representative prototype per class* (RP_c). In order to obtain a first approximation during the classification phase, we propose to use a representative prototype per class, taking advantage of the class information. To compute RP_c, let $Class_c$ be the set of

prototypes in *T*, which belong to the class *c*. Then, for each prototype $P_a \in Class_c$, the following function is computed:

$$AvgSim(P_a) = \frac{\sum_{b=1, b\neq a}^{|Class_c|} F(P_a, P_b)}{|Class_c|} \tag{4}$$

AvgSim evaluates the average of similarity between a fixed prototype (P_a) and the rest of the prototypes that belong to the same class. Thus, the representative prototype for class *c* (RP_c) is the most similar on average (or the one that minimizes *AvgSim* function):

$$RP_c = Argmin(AvgSim(P_a)), \quad \forall a \in [1, |Class_c|] \tag{5}$$

This process is repeated for every $c \in [1, C]$, where *C* is the number of classes in the training set.

3. *Similarity threshold between prototypes* (*SimThres*). The average value of the similarity between the prototypes belonging to the same class in *T*, is used as a confidence threshold to make decisions during the classification phase. This value can be a parameter given by the user. However, in this section three options to compute the confidence threshold are proposed.

To define the similarity threshold for each class *c*, the average of similarity, among the prototypes belonging to the same class, is computed as follows:

$$AvgValueClass_c = \frac{\sum_{a=1}^{|Class_c|} \frac{\sum_{b=1, a\neq b}^{|Class_c|} F(P_a, P_b)}{|Class_c|-1}}{|Class_c|-1}, \quad \forall a, b \in [1, |Class_c|] \tag{6}$$

Finally, the similarity threshold is selected following (7), (8) and (9):

$$SimThres = SimMin = \text{Argmin}(AvgValueClass_c), \quad \forall c \in [1, C] \tag{7}$$

$$SimThres = SimAvg = \frac{\sum_{c=1}^{C} AvgValueClass_c}{C}, \quad \forall c \in [1, C] \tag{8}$$

$$SimThres = SimMax = \text{Argmax}(AvgValueClass_c), \quad \forall c \in [1, C] \tag{9}$$

3.2 Classification Phase

Given a new prototype *Q* to classify, *SM*, RP_c and *SimThres*, computed during the preprocessing phase, are used to avoid comparisons among prototypes. The classification phase of the proposed algorithm (AEMD) is based on Approximating and Eliminating steps, which are not based on the triangle inequality.

Initial approximation step. At the beginning of the algorithm, the prototype Q is compared against the class representative prototypes to obtain a first approximation to the most similar prototype *MSN* and its similarity value S_{MSN}.

$$MSN = \text{ArgMin}\,(F(Q, RP_c)),\ \forall c \in [1, C] \tag{10}$$

The current *MSN* is eliminated from the set T. If $S_{MSN} \geq SimThres$ (where *SimThres* is a confidence value of similarity between prototypes belonging to the same class in T), the prototype *MSN* is used to eliminate prototypes from T (*Eliminating step*).

Eliminating step. In this step, given a fixed prototype (*MSN*) to eliminate prototypes from T, a binary representation (*BR*) contains the similarity per attribute, between Q and *MSN* is created as follows:

$$BR_i(Q, MSN) = C_i(x_i(Q), x_i(MSN)),\ \forall i \in [1, d] \tag{11}$$

Thus, $BR_i(Q, MSN)=1$ if Q and *MSN* are similar in the attribute x_i and $BR_i(Q, MSN)=0$, in other case. Using *BR*, those prototypes in T, which are not similar to *MSN* at least, in the same attributes in which *MSN* is similar to Q, are eliminated from T (using $SM(MSN, P_a), \forall P_a \in T$).

For example, supposed that P_0, P_1, Q and *MSN*, are such that $BR(Q, MSN) = [1,1,0,1,1,1,0,0]$, $SM(MSN, P_0)=[1,1,1,1,1,1,0,1]$ and $SM(MSN, P_1)=[1,0,0,0,0,1,0,1]$. Then, according to this criterion, P_0 is not eliminated because is similar to *MSN* in the same attributes, where *MSN* is similar to Q (attributes 1, 2, 4, 5 and 6). But P_1 is eliminated, without have explicitly compared it to Q, because P_1 is not similar in the same attributes, where *MSN* is similar to Q (*MSN* is similar to Q in attribute 2, but P_1 is not similar to *MSN* in this attribute). The similarity per attribute between *MSN* and P_0 ($SM(MSN, P_0)$) and the similarity per attribute between *MSN* and P_1 ($SM(MSN, P_1)$) are known (because these similarities were computed in the preprocessing phase). The similarity between *MSN* and Q, has already been computed.

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 ($S_{MSN'}$), eliminated from T and used to update the current *MSN*. If $S_{MSN'} < SimThres$ a new *MSN'* is randomly selected (*Approximating step*). Otherwise, if $S_{MSN'} \geq SimThres$ (where *SimThres* is a confidence value of similarity between prototypes belonging to the same class in T), 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 *MSN*, its class is assigned to Q.

4 Experimental Results

In this section, the performance of the proposed classifier (AEMD) is evaluated. In order to compared AEMD; the exhaustive search [1], AESA [2], LAESA ($|BP|$=20% of the objects in the dataset) [3], iAESA [5], Probabilistic iAESA (using 70% as percentage threshold of the data set) [5], TLAESA [4] and modified TLAESA [10] algorithms were evaluated using the same comparison function (F, described in Section 2) instead of using a metric. To do the experiments, 10 datasets from the UCI repository

[12] were used (Mixed datasets: Hepatitis, Zoo, Flag and Echocardiogram. Qualitative datasets: Hayes, Bridges and Soybean-large. Quantitative: Glass, Iris and Wine).

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:

$$Acc = \frac{NoCorrectObj}{NoTestObj} \quad (12)$$

Where, *NoCorrectObj* is the number of correctly classified prototypes in the test set and *NoTestObj* is the size of the test set. The percentage of comparisons between objects was computed as follows:

$$Comp = \frac{NoCompFastClass * 100}{NoTrainingObj} \quad (13)$$

Where, *NoCompFastClass* is the number of comparisons done by the fast classifier, and *NoTrainingObj* is the size of the training set. According to (13), for the exhaustive classifier, the 100 % of the comparisons is done. In all the experiments, *k*=1 in *k-MSN*, was used.

As first experiment, the proposed algorithm (AEMD) was evaluated. To use AEMD algorithm, *SimThres*, which corresponds to a confidence value of similarity, was tested with the values *SimThres* = 40, 60 and 80 (see table 1).

Table 1. Obtained results using AEMD, according to different values of *SimThres*

Datasets	Exhaustive *k-NN* classifier		AEMD (*SimThres*=40%)		AEMD (*SimThres*=60%)		AEMD (*SimThres*=80%)	
	Acc	Comp	Acc	Comp	Acc	Comp	Acc	Comp
Hepatitis	81,66	100	74,65	7,45	80,65	17,37	81,63	23,17
Zoo	96,00	100	78,42	8,83	94,01	9,63	95,30	28,80
Flag	54,67	100	45,15	4,11	53,85	7,62	52,08	13,11
Echocardiogram	82,44	100	80,15	7,44	81,06	9,04	81,70	25,23
Hayes	81,24	100	78,23	8,19	80,21	12,52	81,05	18,57
Soybean-large	85,40	100	65,74	7,32	84,12	8,65	83,07	8,11
Bridges	57,85	100	38,52	3,55	53,54	9,20	56,78	11,19
Glass	68,26	100	62,58	9,11	67,35	13,66	67,90	16,45
Iris	93,30	100	45,52	8,50	91,01	10,29	93,30	12,99
Wine	90,90	100	58,42	7,17	90,90	13,58	89,63	13,18
General average	**79,17**	100	**62,74**	7,17	**77,67**	11,16	**78,24**	17,08

As we can see from table 1, the bigger the value of *SimThres*, the higher the obtained accuracy. However, the percentage of comparisons is also increased. Besides, for some datasets (Echocardiogram and Hayes), good results were obtained using *SimThres*=40, while for other datasets (Flag and Soybean-large) good results are obtained using *SimThres*=60. From these results, we can not conclude an optimal value for *SimThres*. For this reason, the criteria, described in section 3.1, to establish a value for *SimThres*, were used. Thus, AEMD algorithm was evaluated with *SimThres*= *SimMin*, *SimAvg* and *SimMax* (see table 2). From table 2, we can observe that using *SimThres*=*SimAvg*, good results are obtained, for all the datasets.

Table 2. Obtained results using AEMD, according to different values of *SimThres*

Dataset	Exhaustive *k-NN* search		AEMD (*SimThres*=*SimMin*)		AEMD (*SimThres*=*SimAvg*)		AEMD (*SimThres*=*SimMax*)	
	Acc	*Comp*	*Acc*	*Comp*	*Acc*	*Comp*	*Acc*	*Comp*
Hepatitis	81,66	100	80,03	11,95	81,63	13,52	81,66	82,46
Zoo	96,00	100	94,10	9,31	94,00	24,76	96,00	91,00
Flag	54,67	100	52,60	16,87	52,05	17,41	54,67	56,32
Echocardiogram	82,44	100	81,05	13,56	81,70	18,50	82,44	85,60
Hayes	81,24	100	81,16	13,80	81,05	12,97	81,24	79,40
Soybean-large	85,40	100	84,21	15,70	83,07	23,46	85,40	57,55
Bridges	57,85	100	56,12	15,21	57,00	12,54	57,85	66,75
Glass	68,26	100	66,74	12,50	67,92	11,17	68,26	95,33
Iris	93,30	100	93,30	15,04	93,30	12,52	93,30	93,20
Wine	90,90	100	89,63	13,77	89,63	16,32	90,90	78,62
General average	**79,17**	100	**77,89**	13,77	**78,14**	16,32	**79,17**	78,62

Table 3. Obtained results using different classifiers

Dataset	Exhaustive *k-NN* search		AESA		LAESA		i AESA	
	Acc	*Comp*	*Acc*	*Comp*	*Acc*	*Comp*	*Acc*	*Comp*
Hepatitis	81,66	100	80,57	52,96	80,54	61,86	81,03	52,78
Zoo	96,00	100	96,00	23,50	96,00	15,26	96,00	19,36
Flag	54,67	100	51,45	28,02	51,45	25,73	51,36	27,65
Echocardiogram	82,44	100	81,77	62,04	81,77	68,23	81,05	63,62
Hayes	81,24	100	81,24	24,82	81,24	23,32	80,77	17,62
Soybean-large	85,40	100	85,40	2,51	85,40	4,49	85,40	1,96
Bridges	57,85	100	57,85	25,62	57,85	36,10	57,85	25,07
Glass	68,26	100	66,45	14,02	67,92	20,83	66,34	12,62
Iris	93,30	100	93,30	9,22	93,30	6,86	93,30	7,54
Wine	90,90	100	89,01	15,46	90,90	25,26	90,01	10,62
General average	**79,17**	100	**78,30**	25,82	**78,64**	28,79	**78,31**	23,88

Table 4. Obtained results using different classifiers

Dataset	Probabilistic i AESA		TLAESA		Modified TLAESA		PROPOSED CLASSIFIER AEMD (*SimThres*= *SimA*)	
	Acc	*Comp*	*Acc*	*Comp*	*Acc*	*Comp*	*Acc*	*Comp*
Hepatitis	80,64	32,44	81,33	87,54	81,66	72,65	81,63	13,52
Zoo	94,00	17,51	96,00	42,74	96,00	23,95	94,00	24,76
Flag	49,62	26,41	52,84	48,41	52,09	32,95	52,05	17,41
Echocardiogram	80,06	63,08	81,77	71,58	82,44	44,62	81,70	18,50
Hayes	80,07	16,74	80,54	46,42	81,06	24,05	81,05	12,97
Soybean-large	82,15	2,04	85,40	47,51	85,40	16,85	83,07	23,46
Bridges	56,95	25,064	56,74	46,75	57,23	38,74	57,00	12,54
Glass	66,21	12,06	67,92	62,47	67,72	22,85	67,92	11,17
Iris	93,30	8,01	93,30	41,51	93,30	11,65	93,30	12,52
Wine	90,90	10,54	90,90	39,75	90,90	12,64	89,63	16,32
General average	**77,39**	21,39	**78,67**	53,47	**78,78**	30,10	**78,14**	16,32

In order to compare the classifiers proposed in this work, different classifiers, based on Approximating and Eliminating, were considered. In table 3 and 4, the obtained results are shown.

Form table 3 and 4, we can observe that when the comparison function does not satisfy the triangle inequality, AESA, LAESA, iAESA, TLAESA and modified TLAESA algorithms are inexact methods (the obtained results are not the same as using the exhaustive search). However, the percentage of comparisons is, on average, reduced from 100%, done by the exhaustive search, to 25.82 %, 28.79 %, 23.88%, 53.47 % and 30.10%, respectively.

5 Conclusions

In this work, a fast approximated *k-MSN* classifier for mixed data, based on Approximating and Eliminating approach, applicable when the comparison function does not satisfy metric properties was proposed. In order to compare our method, AESA, LAESA, iAESA, probabilistic iAESA, TLAESA, modified TLAESA algorithms were implemented using the same prototype comparison function for mixed data. Based on our experimental results, it is possible to conclude that, our classifier (AEMD) obtained competitive accuracy, but with a smaller number of comparisons between prototypes.

As future work, we are going to look for an strategy to reduce the memory space required to store the similarity matrix (*SM*).

References

1. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. Trans. Information Theory 13, 21–27 (1967)
2. Vidal, E.: An algorithm for finding nearest neighbours in (approximately) constant average time complexity. Pattern Recognition Letters 4, 145–157 (1986)
3. Micó, L., Oncina, J., Vidal, E.: A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing-time and memory requirements. Pattern Recognition Letters 15, 9–17 (1994)
4. Mico, L., Oncina, J., Carrasco, R.: A fast Branch and Bound nearest neighbor classifier in metric spaces. Pattern Recognition Letters 17, 731–739 (1996)
5. Figueroa, K., Chávez, E., Navarro, G., Paredes, R.: On the last cost for proximity searching in metric spaces. In: Àlvarez, C., Serna, M.J. (eds.) WEA 2006. LNCS, vol. 4007, pp. 279–290. Springer, Heidelberg (2006)
6. Fukunaga, K., Narendra, P.: A branch and bound algorithm for computing k-nearest neighbors. IEEE Trans. Comput. 24, 743–750 (1975)
7. Gómez-Ballester, E., Mico, L., Oncina, J.: Some approaches to improve tree-based nearest neighbor search algorithms. Pattern Recognition Letters 39, 171–179 (2006)
8. 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 40(2), 360–375 (2007)
9. Bustos, B., Navarro, G.: Probabilistic proximity search algorithms based on compact partitions. Journal of Discrete Algorithms (JDA) 2(1), 115–134 (2003)
10. Tokoro, K., Yamaguchi, K., Masuda, S.: Improvements of TLAESA nearest neighbor search and extension to approximation search. In: ACSC 2006: Proceedings of the 29th Australian Computer Science Conference, pp. 77–83 (2006)
11. García-Serrano, J.R., Martínez-Trinidad, J.F.: Extension to C-Means Algorithm for the use of Similarity Functions. In: Żytkow, J.M., Rauch, J. (eds.) PKDD 1999. LNCS (LNAI), vol. 1704, pp. 354–359. Springer, Heidelberg (1999)
12. Blake, C., Merz, C.: UCI Repository of machine learning databases, Department of Information and Computer Science, University of California, Irvine, CA (1998), `http://www.uci.edu/mlearn/databases/`