SNN: A Supervised Clustering Algorithm

Jesús S. Aguilar, Roberto Ruiz, José C. Riquelme, and Raúl Giráldez

Department of Computer Science. University of Sevilla
Avda. Reina Mercedes s/n. 41011 Sevilla. Spain.
aguilar@lsi.us.es

Abstract. In this paper, we present a new algorithm based on the nearest neighbours method, for discovering groups and identifying interesting distributions in the underlying data in the labelled databases. We introduces the theory of nearest neighbours sets in order to base the algorithm S-NN (Similar Nearest Neighbours). Traditional clustering algorithms are very sensitive to the user-defined parameters and an expert knowledge is required to choose the values. Frequently, these algorithms are fragile in the presence of outliers and any adjust well to spherical shapes. Experiments have shown that S-NN is accurate discovering arbitrary shapes and density clusters, since it takes into account the internal features of each cluster, and it does not depend on a user-supplied static model. S-NN achieve this by collecting the nearest neighbours with the same label until the enemy is found (it has not the same label). The determinism and the results offered to the researcher turn it into a valuable tool for the representation of the inherent knowledge to the labelled databases.

Keywords: clustering, supervised learning, nearest neighbours.

1. Introduction

In the area of the supervised learning there are several techniques to classify a new example from the labelled database from which the inherent knowledge has been obtained. The form in which it estates the knowledge is dependent on the technique (decision rules, decision trees, association rules, etc.); however, some methods do not provide that knowledge limiting themselves to carry out the classification (neuronal networks, Bayesian model, nearest neighbours, etc.).

From the works of [3], [5], [9], [6], [7], [4], [10], [11], or more recently [12], [8], and [1] the research has been mainly focused on the convergence of the method, the search of prototypes or surfaces of separation, the techniques of editing and condensing and in the acceleration of algorithm. However, there has not been any interest on providing to the technique of the nearest neighbours a form to represent the inherent knowledge to the information.

Clustering, in Data Mining, is a useful technique for grouping data points such that points in a single cluster have similar characteristics (or are close to each other). Traditional clustering algorithms are applied in the area of the learning non-supervised.

S-NN employs a novel hierarchical clustering algorithm based on the nearest neighbour techniques. S-NN stars with each input as a separate cluster and at each successive step merges the clusters with identical neighbours. We collect all the neighbours that their distances are shorter than the first enemy, that is to say, with not the same label.

The remainder of the paper in organised as follows. In section 2 and 3, we survey basis contents of the theory of the nearest neighbours' sets. These hard definitions allow us apply the supervised clustering algorithm. The step involved en clustering using S-NN are described in Section 4. In Section 5, we present the results of our experiments. Section 6 concludes and presents our ideas for future work.

2. Basic Concepts

Before beginning to describe the near set theory, we have to mention the concepts of the classic theory of sets that are necessary for the development of that theory. We will use the operations known on sets: $\in$, $\cup$, $\cap$ and # (cardinal of a set). Also we will use the logic operations on the set *{F, T}* (false and true): $\wedge$, $\vee$, and $\neg$; and the following generalisations: $\forall$ (universal quantifier) and $\exists$ (existential quantifier), where

$$z = \forall i : D(\overline{x}).E \equiv E(x_1) \wedge E(x_2) \wedge ... E(x_n)\ ) \qquad \textbf{(1)}$$
$$z = \exists i : D(\overline{x}) \cdot E \equiv E(x_1) \vee E(x_2) \vee \ldots \vee E(x_n)$$

and z is T if all the expressions (if some of the expressions) $E(x_i)$ are T in the domain D for $\overline{x} = (x_1,...,x_n)$, if we consider the universal quantifier (existential quantifier).

Definition 1 (Sequence): a sequence is a finite or infinite collection of elements with an inherent order of access (sequential). It is always begun by first and to accede to any element *i*, it will be necessary to pass through *i-1* previous. Since its definition is inherited of sets, it also inherits the operations associated to these $\in$, $\cup$, $\cap$, - and #.

Likewise, we defined the following operations for sequence *S* of elements of *T* type: $<>: \rightarrow S$ (empty sequence); $\_+\_ : S \times T \rightarrow S$ (insertion of an element in the end of the sequence); $[\_]: S \times N \rightarrow T$ (access to *i*th element of the sequence, with $i \in \{1... \#S\}$); and $+ i : D \cdot E$ (generalised concatenation of sequences), where

$$+ i : \{1..k\} \cdot s(i) = \mathrm{s}(1) + \mathrm{s}(2) + ... + \mathrm{s}(\mathrm{k}) \qquad \textbf{(2)}$$

with *s(i)* sequences. By convenience, it will be written as $\overset{k}{\underset{i=1}{+}} s(i)$.

Definition 2 (Ordered Sequence): a sequence *s*, of size *#s* is ordered if it satisfies:

$$\forall i : \{1..(\#s) - 1\} \cdot s[i] \leq_T s[i+1] \qquad \textbf{(3)}$$

where $\leq_T$ is an established relation of total order between the elements of *T* type of the sequence.

3. Definitions

Definition 3 (Attribute): attribute *A* is defined by a set of values. The attribute can be continuous or discrete. If the attribute is continuous, the set of values will be limited by the extreme values of an interval, forming therefore the rank of values for the attribute. If the attribute is discrete, the set of values of this one will appear like an enumeration of the possible values of the attribute. We will name *C* the set of values that can adopt the label.

Definition 4 (Example): an example E is one row formed by the Cartesian product of the attributes of condition and decision. Likewise, we defined the following operations to get to the attributes of condition or their label.

$$\mathrm{atr}: \mathrm{E} \times \mathrm{N} \to \mathrm{A} \quad \mathrm{etiq}: \mathrm{E} \to \mathrm{C} \tag{4}$$

Definition 5 (Universe): the universe *U* is a sequence of examples. We will say that a database with *n* examples, each one of them with *m* attributes (the last one is denominated label), will form the particular universe from this moment. Then *U=<u[1]...,u[n]>.*

Since we will model the database with a sequence, the access for an example of the database will be made by means of the access to the sequence, that is to say, the sequence is *s*, then *s[i]* represents the example *i*th of the database. To accede to *j*th attribute of the example, since we have modelled to this one with one row, one will become *atr(s[i],j),* and to know its label, *etiq(s[i]).*

Definition 6 (Distance): the distance between two examples is a function that fulfils the properties of a metric space, that is to say,

$$d: E \times E \to \Re^{+} \cup \{0\} \tag{5}$$

with the following properties: reflective, defined nonnegative, symmetrical and transitive.

Since the examples belong to a sequence, we can redefine the distance basing on the position that these examples occupy in the sequence, therefore, compute the range between two examples e_i and e_j, we will do *d(i,j).*

Definition 7 (Sequence of Distances): sequence *SD(i)* formed by the distances of an example *i* to all the others, and is defined by

$$SD(i) = \mathop{+}_{j=1}^{\#s} (j, d(i,j)) \tag{6}$$

In the universe *U*, the sequence associated for the first example will be: *SD(1)=<(1,d(1,1)),(2,d(1,2)),(3,d(1,3))>* and each element of the sequence are a pair formed by the position of the example for which it is wanted to compute the range and the value of the distance. Hence, in the expression *(j,d(i,j))* the first coordinate is the index of an example and the second coordinate is the distance of an example *i* for the example whose index is indicated in the first coordinate. In order to access to each one of the two coordinates easily we defined two operations on the pair:

$$\mathrm{ind}: \mathrm{N} \times \Re \to \mathrm{N} \quad \mathrm{dist}: \mathrm{N} \times \Re \to \Re \tag{7}$$

Definition 8 (Ordered Sequence of Distances): as of a sequence *P* of pairs (index, distance), we can obtain a sequence *Q* ordered by the distance if this one fulfils the following property:

$$\forall j:\{1..\#Q-1\}\cdot dist(Q[j]) \le dist(Q[j+1]) \wedge \forall j:\{1..\#Q\}\exists k:\{1..\#P\}\cdot ind(P[k]) = ind(Q[j]) \qquad \textbf{(8)}$$

that we can be resumed like *ordered(Q,P)*. A ordered sequence *OSD(i)* of distances fulfils *ordered(OSD(i), SD(i))*.

Definition 9 (Relation of Neighbourhood): two examples whose positions in the ordered sequence are *i* and *j* are neighbours to a third *k* if in the ordered sequence of distances of example *k*, *OSD(k)*, any example between *i* and *j* does not exist (or between *j* and *i*) whose label is different from which they have *i* and *j*. Therefore, if examples *i* and *j* do not have the same label, are not neighbours. The relation can be defined of the following way:

$$R_k=\{(u[ind(OSD(k)[i])],u[ind(OSD(k)[j])])\in U^2 | \forall h:\{min(i,j)+1..max(i,j)-1\}) \cdot(etiq(u[ind(OSD(k)[i])])=etiq(u[ind\ (OSD(k)[j])])=etiq(u[ind(OSD(k)[h])])\} \qquad \textbf{(9)}$$

For example, it is *OSD(1)=<(1,0),(4,1),(5,2),(34,3),(3,4)>*, we will say that examples 4 and 3 are neighbours of 1 if examples 4 and 3 have the same label and all the examples that are between these (5 and 34) have the same label that those too.

Definition 10 (Class of Neighbours of an Example i respect to another Example k): class of neighbours of an example *i* is defined respect to another *k* like all those that in the ordered sequence of distances of *k* can be grouped around *i* in a region of examples of the same class. The class is defined from the neighbourhood relation as it follows: $[\,i]_k=\{j\in N \mid u[i\,]\ R_k\ u[j]\}$

This class of neighbours can also be understood like a sequence, because intrinsically an order relative to the distance exists [2].

Definition 11 (Ordered Subsequence of Order k respect to an Example i): given to the relation of neighbourhood and the definition of class of neighbours, we can construct the ordered sequence of distances of an example *i* from the concatenation of ordered subsequences, that is to say,

$$OSD(i)=OSD(i)_1+OSD(i)_2+...\ +OSD(i)_k+...\ +OSD(i)_z \qquad \textbf{(10)}$$

where every subsequence $OSD(i)_k$ is constructed from the classes of separates examples in relation to *i*. This way, each $OSD(i)_k$ is a class of examples whose attribute of decision is the same one, but that differs from the decision attribute of the classes $OSD(i)_{k-1}$ and $OSD(i)_{k+1}$. Therefore, we could represent the database (together with the information relative to the distances) of the following way:

$$\begin{array}{l} OSD(1)=OSD(1)_1+OSD(1)_2+...\ +OSD(1)_{k1} \\ OSD(2)=OSD(2)_1+OSD(2)_2+...\ +OSD(2)_{k2} \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ OSD(n)=OSD(n)_1+OSD(n)_2+...\ +OSD(n)_{kn} \end{array} \qquad \textbf{(11)}$$

where $K_i \in \{1..n\}$; however, if some K_i were 1, all the examples would belong to the same class, and the more it approaches *n*, in principle, the more homogeneously distributed will be the examples of the same class in the database.

Since the sequence *OSD* has got associated with each element an example and the distance to it, we are going to do without the distance now to associate for each example of the database a sequence of classes, where the concatenation of all of them will group the total of examples of the database. Hence we will have associated for each example (preceding to the symbol ≈) an indefinite number of classes in a specific order that implicitly contains the information about the "proximity" of these for the example of the beginning. Then,

$$\begin{aligned}[1] &\approx [1]_1+[1]_2+\ldots+[1]_{k1} \\ [2] &\approx [2]_1+[2]_2+\ldots+[2]_{k2} \\ &\ldots\ldots\ldots\ldots \\ [n] &\approx [n]_1+[n]_2+\ldots+[n]_{kn}\end{aligned} \qquad \textbf{(12)}$$

it indicates that example *k* has a class of neighbouring examples $[k]_1$ whose labels are the same ones that the one of *k*. Afterwards, there is another class of neighbouring examples $[k]_2$ whose labels are different that those of the previous class $[k]_1$ and the later class $[k]_3$. And so on. To these classes, $[i]_j$, we will denominate classes *j*-neighbours of an example *i*.

From a mathematical point of view, we have obtained the joint quotient according to the relation of neighbourhood *R* for each example of the universe:

$$\forall i:\{1..\#U\}\cdot[i]=OSD(i)/R \qquad \textbf{(13)}$$

For example, if *OSD(1)=<(1,0),(4,1),(5,2),(34,3),(3,4)> + <(73,5),(2,6),(31,7)>+ ...* we are indicating that examples 1, 4, 5, 34 and 3 have the same label (values 0, 1, 2, 3 and 4 would be the distance of each example for example 1), that in addition differs from the one to examples 73, 2 and 31. From this we can here construct the class [1] of the following form: *[1]≈[1,4, 5, 34,3]+[73, 2, 31]+….*The class 1-neighbour of 1 is [1, 4, 5, 34,3]; the class 2-neighbour of 1 is [73, 2, 31]; and so on.

Definition 12 (Class of Order k j-Neighbour of an Example i): the class of order 0 is defined 1-neighbour of an example *i*, $[i]_1^0$, like the class of neighbours of an example *i* respect to itself. That is to say, since in the ordered sequence of distances of an example *i*, *OSD(i)*, the first example always will be the own *i*, since the distance to itself is 0, the class of order 0 1-neighbour of example *i* will be the set of neighbouring examples of this one whose label is the same one, or of another form, they will be those that belong to $OSD(i)_1$.

On the other hand, the class of order 0 *j*-neighbour of an example *i* will be $[i]_j^0$. We are specially interested on the classes of order *k* 1-neighbours. We define then the order class 1 1-neighbour like:

$$[i]_1^1=[i]_1^0\cup\left\langle j \,\middle|\, j\in\bigcup_{k\in[i]_1^0}[k]_1^0\right\rangle \qquad \textbf{(14)}$$

The interpretation of this expression is the following one: since *i* contains all the examples with the same label than *i* that is nearer to it (including it) until finding another example of different label, the class $[i]_1^1$ has to those contained in $[i]_1^0$ plus the 1-neighbours of them. For example, in Fig. 1 we have $[i]_1^0$ =*<i,1,2,3,4,5>* (the 6 does not belong to it, it has another label).

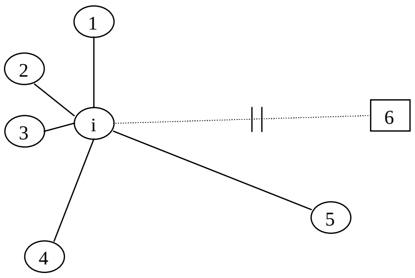

Fig. 1.

In general, the class of order *k* 1-neighbour of an example *i* is defined as:

$$[i]_1^k = \bigcup_{0 \le j < k} [i]_1^j \qquad \textbf{(15)}$$

And, therefore, the class of order *k* *j*-neighbour of an example *i* is defined as:

$$[i]_j^k = \bigcup_{0 \le h < k} [i]_j^h \qquad \textbf{(16)}$$

By convenience, we will speak of *k*-class instead of class of order *k*, and therefore, *k*-class *j*-neighbour, instead of class of order *k* *j*-neighbour. In particular, we are interested on the *k*-classes 1-neighbours, and when we will speak about them we omit subscript 1, that is to say, instead of $[i]_1^k$ we will write $[i]^k$. Only when the neighbourhood order is different from 1 we will express this order.

Definition 13 (Equality of Classes): two classes are equal when both contain exactly the same examples, although in different order. Formally,

$$[i] = [j] \Leftrightarrow (\forall e \in [i] \Rightarrow e \in [j] \wedge \forall e \in [j] \Rightarrow e \in [i]) \qquad \textbf{(17)}$$

Definition 14 (Set of k-Classes 1-Neighbours): we define the set of k-classes 1-neighbour like the set formed by the k-classes 1-neighbour for each example. Also, by convenience, k-set of neighbouring classes will be named, instead of set of k-classes 1-neighbour, and it will written like SN^k where

$$SN^k = \{[1]^k, [2]^k, \ldots, [n]^k\} \qquad \textbf{(18)}$$

Definition 15 (Reduced k-Set of Neighbouring Classes): we define k-set reduced of neighbouring classes as the set of k-classes 1-neighbours where there are not two equal classes. Formally,

$$RSN^k = \{[i]^k \in SN^k \mid \forall [j]^k \in SN^K \cdot i \neq j \Rightarrow [i]^k \neq [j]^k\} \qquad \textbf{(19)}$$

We will identify the reduced sets of neighbouring classes with the examples that have been reduced so that we do not lose information, that is to say, if the class [i] and the class [j] are equal, since their neighbours are the same [w_1...,w_k], then [i, j] has as neighbours to [w_1...,w_k].

4. Algorithm ‘‘Similar Nearest Neighbour" (S-NN)

Once seen all the necessary definitions that support the theory that we present in this work, we describe the details of our algorithm in figure 2.

```
S-NN (U: Database) ret (RSN: Set of Classes)
          $i \leftarrow 0$
          $SN_1^i \leftarrow \{\ \}$
          For each example j de U
            $SN_1^i \leftarrow SN_1^i \cup \{[j]_1^i\}$
          $RSN_1^{i-1} \leftarrow \{\ \}$ (by convenience $RSN_1^{-1}$)
          While $SN_1^i \neq RSN_1^{i-1}$
            $RSN_1^i \leftarrow reduction(SN_1^i)$
            $SN_1^{i+1} \leftarrow \{\ \}$
            For each $[j]_1^i \in RSN_1^i$
              For each $k \in [j]_1^i$
                $[j]_1^{i+1} \leftarrow [j]_1^i \cup [k]_1^i$
              $SN_1^{i+1} \leftarrow SN_1^{i+1} \cup \{[j]_1^{i+1}\}$
            $i \leftarrow i+1$
```

Fig. 2. Algorithm

The Input parameters are the U database, containing n examples with m attributes. As we mentioned earlier, starting with the individual points as individual clusters, at each successive step the clusters with identical neighbours are merged. The process is repeated until we can not simplify the set of clusters.

S-NN treats each input point as a separate cluster, in each iteration of the while-loop, until we can not simplify the set of clusters, we compute the neighbours of each cluster member.

```
reduction (C:Set of Classes) ret (RSN:Set of Classes)
          $RSN \leftarrow C$
          For each (x,y) con x, y $\in$ C
            If [x]=[y]       ([x]=[x]$\cup$[y]=[x]$\cap$[y])
              $RSN \leftarrow RSN - \{y\}$
              $x \leftarrow x \cup y$
```

Fig. 3. Reduction

The expression $\mathrm{RSN}_1^i \leftarrow \mathrm{reduction}(\mathrm{SN}_1^i)$ invokes to the following algorithm shown in figure 3, whose assignment is to simplify the set of classes by means of the elimination of those classes that have exactly the same neighbours. If there are two classes *x* and *y* and they have the same neighbours, then the examples of *y* are added to those of *x*, which both, will have exactly the same neighbours.

5. Results

5.1. Iris

We have used the database Iris to illustrate the complete results of the method because they are possible to be included in the article. However, we regret not to be able to include, by lack of space, the intermediate results (set SN and RSN for each order of iterations).

The next table contains two types of rows:

- odd rows: [class, examples of the class, neighbours of the examples of the class]. The first value refers to the class or labels; the second indicates how many examples belong to the class that has the mentioned label; and the third value corresponds with the number of neighbours that have got that class.
- even rows: the example of the class are placed on the left column, whose cardinal corresponds with the second number of the previous row; and in the right column the neighbours of the examples of the class are placed, of the left column, and has got as cardinal the third value of the previous row.

[A,50,50]	
1, 6, 10, 18, 26, 31, 36, 37, 40, 42, 44, 47, 50, 51, 53, 54, 55, 58, 59, 60, 63, 64, 67, 68, 71, 72, 78, 79, 87, 88, 91, 95, 96, 100, 101, 106, 107, 112, 115, 124, 125, 134, 135, 138, 139, 143, 144, 145, 149, 136	1, 95, 106, 55, 36, 64, 125, 88, 107, 112,145, 134, 72, 67, 63, 37, 100, 31, 54, 135, 50, 47, 68, 6, 18, 101, 144, 78, 42, 143, 149, 139, 53, 51, 26, 115, 40, 10, 44, 96, 60, 124, 91, 58, 79, 138, 87, 59, 136, 71
[C, 47, 48]	
2, 4, 17, 21, 23, 24, 39, 41, 45, 73, 80, 89, 102, 110, 126, 7, 13, 15, 20, 35, 27, 49, 104, 56, 57, 74, 148, 77, 81, 83, 111, 122, 123, 127, 131, 132, 146, 16, 75, 32, 34, 46, 52, 62, 82, 108, 137	2, 57, 122, 83, 131, 4, 15, 132, 13, 35, 81, 27, 41, 111, 123, 74, 17, 102, 23, 20, 127, 148, 34, 7, 80, 146, 46, 32, 75, 16, 5, 56, 49, 77, 126, 52, 104, 110, 73, 24, 62, 45, 108, 137, 82, 39, 21, 89
[B, 47, 49]	
3, 28, 113, 8, 11, 14, 76, 85, 86, 116, 109, 121, 129, 19, 29, 30, 43, 66, 70, 99, 33, 98, 48, 133, 38, 61, 119, 65, 69, 84, 150, 93, 92, 94, 97, 103, 105, 114, 141, 118, 120, 140, 128, 130, 142, 22, 147	3, 92, 141, 113, 142, 119, 61, 29, 128, 11, 117, 103, 130, 28, 118, 147, 22, 69, 30, 105, 114, 76, 86, 66, 94, 19, 121, 43, 99, 116, 85, 65, 8, 109, 14, 33, 140, 98, 133, 70, 48, 129, 93, 38, 9, 150, 97, 84, 120
[C, 1, 1]	
5	5
[B, 1, 1]	
9	9
[B, 1, 1]	
12	12
[C, 1, 1]	
25	25
[C, 1, 1]	
90	90
[B, 1, 1]	
117	117

For the database Iris 4 iterations have been needed, in each one of which the cardinal one of set RSN has been: 98, 62, 19 and last the 9 that is in the table.

The method offers a very valuable information because it provides:

- The number of regions: 9.
- If the examples of the class agree with the neighbours (fact that happens for the class A) then the region is clearly separable of the rest.
- Which are the examples that make difficult the classification (5, 9, 12, 25, 90 and 117), therefore, we could extract them from the database for a later classification.
- An estimation of the error rate on the training file (let us take into account that if we keep the three first regions we would be around 96%, that is approximately what they provide other good sort keys).

5.2. Breast Cancer

For this database (with the 683 examples without noise), and needing 5 iterations we have obtained 44 regions. The regions calculated in each iteration are: 440, 308, 142, 45 and 44. The two first are stops A (427 examples) and for B (210 examples), which means that only with these two we would be around the 93,26% of the information.

Regarding the computational cost of the algorithm, the executions have been made in a PC Pentium 550 MHz and for the database Iris it uses less than 1 second; for the breast cancer database it uses 2 minutes.

6. Conclusions

The definitions presented in this article base the theory that it supports on algorithm S-NN. The algorithm, besides does not need parameters is determinist. As for the information that it provides, in the example Iris it is demonstrated that it is able to obtain: a geometric idea of the distribution of examples of the database; an estimation of the number of regions (possible rules); an estimation of the difficulty of classification of the database; and which are the examples that make difficult the learning of the database, with a view to eliminate them in the phase of learning.

On the other hand, algorithm S-NN allows some interesting directions, which we are studying, as far as the reduction criterion is concerned (it see point 2,1 of the reduction algorithm). Three criteria of reduction exist: a restrictive criterion (the one that at the moment is applied, that is to say, they will be reduction if the two classes are exactly equal); a moderate criterion (there will be reduction if one of the classes is included in the other); and, finally, a relaxed criterion (there will be reduction if the intersection of the classes is not empty). These criteria provide different solutions, as much more numerous as for regions, as restrictive is the reduction criterion. The characteristics of the contributed solutions as well as their differences, both analytical and geometrically, will be object of next works. In the same way, another interesting line is the use of the set of classes of neighbours like sort key.

Acknowledgements. This work has been supported by the Spanish Research Agency CICYT under grant TIC99-0351.

References

1. Aha, D. W. A (1990). Study of Instance-Based Algorithms for Supervised Learning Tasks: Mathematical, Empirical, and psychological Evaluations. Ph. D. Dissertation. UCI.
2. Codrington, C. W. y Brodley, C. E. (1997). On the Qualitative Behavior of Impurity-Based Splitting Rules I: The Minima-Free Property. Technical Report, Purdue University.
3. Cover, T. M. y Hart, P. E. (1967). Nearest Neighbor Pattern Classification. NN-Pattern Classification Techniques. IEEE.
4. Chang, C. L. (1974). Finding Prototypes for Nearest Neighbor Classifiers. IEEE Transactions on Computers.
5. Hart, P.E. (1968). The condensed nearest neighbor rule. IEEE Transactions on Information Theory, IT-14.
6. Hellman, M. E. (1970). The Nearest Neighbor Classification Rule with a Reject Option. NN-Pattern Classification Techniques. IEEE.
7. Jarvis, R. A. y Patrick, E. A. (1973). Clustering using a Similarity Measure Based on Shared Near Neighbors. IEEE Transactions on Computers.
8. Joussellin, A. y Dubuisson, B. (1987). A Link Between k-Nearest Neighbor Rules and Knowledge Based Systems by Sequence Analysis. Pattern Recognition Letters.
9. Patrick, E. A. y Fischer, F. P. (1970). A Generalized k-Nearest Neighbor Rule. NN-Pattern Classification Techniques. IEEE.
10. Ritter, G. L., Woodruff, H.B., Lowry, S.R. y Isenhour, T.L. (1975). An algorithm for a Selective Nearest Neighbor Decision Rule. IEEE Transactions on Information Theory, 21.
11. Tomek, I. (1976). An Experiment with the Edited Nearest-Neighbor Rule. IEEE Transactions on Systems, Man, an Cybernetics SMC-6.
12. Wilson, D. (1972). Asymptotic Properties of Nearest Neighbor Rules using Edited Data. IEEE Transactions on Systems, Man and Cybernetics 2.
13. D. Fisher. (1995). Optimization and Simplification of Hierarchical Clusters. Proceedings of the International Conference on Knowledge Discovery and Data Mining.
14. S. Guha, (1998). CURE: An Efficient Clustering Algorithm for Large Databases. Proceedings of the 1998 ACM SIGMOD Conference.