Mathware & Soft Computing 14 (2007) 5-22

Algorithms for Classification based on k-NN

Manuel Laguía[1] and Juan L. Castro[2]
[1]Dept. Computer Languages and Systems, Esc. Sup. Ingeniería
University of Cádiz. 11002 - Cádiz (Spain)
[2]Dept. Computer Science and A.I., E.T.S. Ingeniería Informática
University of Granada. 18071 - Granada (Spain)
manuel.laguia@uca.es, castro@decsai.ugr.es

Abstract

In this paper we focus on methods that solve classification tasks based on distances, and we introduce some variants of the basic k–NN method adding up to 3 characteristics. The experiments reveal a relationship between the accuracy of 1–NN (distances) and the accuracy of the methods based on those distances. We propose a heuristics according to this observation and test its correctness.

We study the usefulness of the proposed methods ε–ball, ε–ball^{k-NN} and ε–ball^{1-NN}, and make an exhaustive comparison using six different distance functions and 68 data sets, including UCI–Repository and artificial data sets. The proposed methods are useful and significantly outperform k–NN frequently. We have also found some evidence about the weakness of k–NN when the optimal value of k varies in different regions along the space.

1 Introduction

We are mainly interested in studying methods of classification based on k–NN and distances. We wish to make comparisons between them, establish their weak and strong points, and extract recommendations about the usefulness of these methods under different circumstances.

The simpler method based on distances is the nearest neighbor (1–NN) [17, 13, 15]. When we try to classify a new case, we directly assign the class of the case (or point) which is at a lower distance according to a function or measure of distance. Usually this measure of distance is the Euclidean metric, but it can be any function which provides a measure of dissimilarity (low values indicate that cases are similar) [30, 31]. Ideally, this function should provide a measure of distance between classes. In practice, we do not know the classification of the new case (this is exactly what we want to know), and we use in some sense a principle of locality: if the attributes of two cases are similar, probably their classes will be similar. But beyond theoretical considerations, the fact is that experimentally we know that this method works and makes right classifications.

1–NN methods suffer from some problems. Firstly, if we could find the optimal function of distance we will obtain an optimal classifier, but finding this function is equivalent to the original problem (being either not trivial or impossible). However, there have been attempts to find an optimal distance on some assumptions [20, 26, 19]. Secondly, once we accept that, in general, we cannot know the optimal distance, we must assume that often the class of the nearest neighbor will not be the right classification, i.e., depending on the base, some points violate the locality principle (1–NN has strong difficulties with outliers and noisy data).

Some variants of 1–NN methods have been studied in the literature, including the IBx series [4, 2] to reduce the storage requirements and increase noise tolerance; the nested generalized exemplar (NGE) theory [32] where hyperrectangles are used instead of points; the value difference metric (VDM) [33], the modified value difference metric (MVDM) [12], the heterogeneous value difference metric (HVDM) [39] and the simplified value difference metric (SVDM) [16] that statistically derive the distances for nominal attributes based on the overall similarity of classification of all instances for each possible value of each feature. Another interesting distance is the local asymmetrically weighted similarity metric (LASM) [29] which defines a local distance that varies along the space and is asymmetric.

k nearest neighbor (k–NN) methods make the classification considering the k nearest points to the new case, instead of using only one point [35, 38]. Here the locality principle can be more flexible in the sense that if the class of the nearest neighbor were not the right one, perhaps it would be the class of some of the k nearest neighbors. Thus, it is supposed that the measure of distance is less critical in k–NN. But two new questions arise with k–NN: choosing the right value of k (how many cases must we consider?), and combining the information of those k nearest neighbor. Regarding the combination of the information of the k nearest neighbors, the most straightforward way consists in assigning the class of the majority. Many combinations can be made, but experimental evidence strongly recommends using the "weighted vote k–NN" [35] (we have also verified this fact in preliminary experiments): consider the distance of the k points, and inversely average the relevance or weight of each case by their distance to the new case.

The optimal value of k depends on the case base, the measure of distance, and the way of combining the information of the k neighbors. k may be estimated by cross–validation [34, 35, 38]. It is possible, but unusual, to compute a local value of k for each new point instead of using one value of k in the whole space [35, 37].

k–NN can suffer problems related with the value of k. If k is too big or the new case is in a sparsely populated region, k–NN will consider distant points that are probably not very relevant. If k is too little or the new case is in a densely populated region, k–NN will ignore some close points that are probably relevant. Weighted vote k–NN can reduce the undesirable consequences of a bad choice of k, but it can still be in trouble if the optimal value of k varies along the space.

Some variants of k–NN methods have been studied in the literature, including the k surrounding neighbors (k–SN) [40] that selects k instances that are close to and "well distributed" around the new point. Other efforts have been made in reducing the number of features that must be considered and taking into account only the relevant ones (feature selection), mainly by global dimension reduction

Table 1: Proposed variants of the basic k–NN method.

Code	Characteristics	Classification method
A		k–NN
B	C1	ε–ball
C	C3	k–NN heur
D	C1 & C2	ε–ball^{k-NN}
E	C1 & C3	ε–ball heur
F	C1 & C2 & C3	ε–ball^{k-NN} heur
D1	C1 & C2′	ε–ball^{1-NN}
F1	C1 & C2′ & C3	ε–ball^{1-NN} heur

[20, 9, 5], or even by local feature selection [16].

Other papers merge k–NN and neural networks [10], k–NN and fuzzy sets [11], or focus on similarity notion instead of using distances, like Plaza et al. [28] that uses an interesting entropy–based assessment, but there is a relationship between the notion of similarity and distance [30, 31, 25, 24]. Another interesting idea is the racing algorithm [27] which discards the less promising models and can find relevant features.

In Section 2 we introduce some variants of k–NN, and propose a heuristics that in our experiments achieves a good performance. In Section 3 we describe the experiments, methods of classification and data sets that we have tested. In Section 4 we present the results of the empirical evaluation and discuss the strengths and weaknesses of the different methods. Finally, in Section 5, we present the main conclusions of the work and future research directions.

2 The Proposed k–NN Variants

We propose some variants of k–NN [35, 38] that differ in three basic characteristics:

C1: Impose a distance threshold ε on the set K of k–nearest neighbors.

C2: Whenever K is empty, use k–NN.

C3: Select its distance measure, from among a set of previously considered distances, by selecting the measure that performed best for 1–NN on the same training set.

Now we can handle these three characteristics as independent, to make comparisons and to study the usefulness of each one. Combining C1, C2 and C3 we obtain the first 6 variants shown in Table 1. C2′ denotes that whenever K is empty we use 1–NN, instead of k–NN. As we will discuss later, it is useful to distinguish this special case. But, let us now introduce more formally each characteristic.

To overcome the difficulties of the k–NN classifiers we propose the opposite point of view: when a new case must be classified we should establish a ball around that point and take into account the points inside that ball, instead of selecting a fixed

number of nearest points. In this way we always guarantee considering only close points independently of the point density of the case base: when e is in a sparse area of the space we take into account fewer points than when it is in a dense area.

We choose a fixed real value ε as the radius of the ball, thus we control the size of the ball. We define the *ε–ball* of an instance e as $K = \{e\prime \in S$ such that $d(e\prime, e) \leq \varepsilon\}$ and the *ε–ball of the class C_i* of an instance e as $K_i = \{e\prime \in S$ such that $class(e\prime) = C_i$ and $d(e\prime, e) \leq \varepsilon\}$, where S is the data set. In the influence measure of the ε–ball of a class it is important to consider the distance and number of instances of the ε–ball that belong to that class. We define the *measure of influence* as:

$$|K_i|_d = \sum_{e\prime \in K_i} \exp(-\alpha d(e, e\prime)) \qquad \text{where } \alpha = \frac{4}{\varepsilon^2}$$

To classify a new case e, we calculate the ε–ball influence measure of each class, and assign to e the class with the greatest influence.

In previous works we have observed that it is desirable the soft behavior of the exponential function from nearer (and probably relevant) points to more distant (and probably less important) points. This soft behavior has been also studied in works of other areas, as cognitive psychology, statistics or neural networks.

If the value of ε is too little it can occur that K is empty. One way to partially respond this question consists in choosing the class of the nearest neighbors whenever K is empty, i.e., when the ε–ball "fails" then we should classify with k–NN, which always provides nearest neighbors and a class. We have also studied using 1–NN when K is empty.

As expected, if there are irrelevant features (led24, waveform–40, and bands), the best behavior is obtained using feature weighting that takes into account information about the importance or utility of the attributes (correlation with the class). But other methods may be used such as mutual information [35, 38], or information gain weighting based on entropy [14]. See [36] for a useful survey of feature weighting methods. Locally weighted techniques have also been explored [20, 6, 7, 18], including class weighting algorithms [21].

We have noticed that there is a certain relationship between the accuracy obtained by 1–NN methods with different distances and the accuracy of other methods based in those distances, for instance, Figure 1 shows the results of each distance in the letter recognition data set. This idea is also supported by some data sets (see Online Appendix A), where one distance obtains extremely poor results (compared with the other distances), and the same happens to the methods that use these distances. This leads us to think that 1–NN methods can be used to extract useful characteristics about the data sets and the distribution of points along the space. For these reasons, and considering that 1–NN methods are relatively fast, we propose the following heuristics: *you should train and test 1–NN methods with a set of distances and use the desired method based on distances (k–NN, ε–ball, ε–ball^{k-NN} ...) with the distance which offers the best accuracy with 1–NN.*

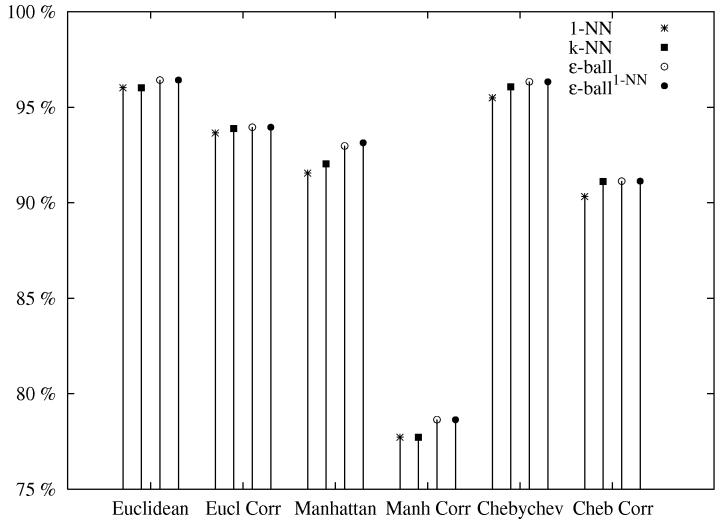


Figure 1: Accuracy of the 1–NN, k–NN, ε–ball and ε–ball^{1-NN} methods with each function of distance in the letter recognition data set. The relationship between the accuracy of 1–NN and the accuracy of the other methods is clear.

3 The experiments

We have used a large number of data sets to study the behavior of the different classifiers, and we have tested them using 10–fold cross–validation [34]. We have tested all the classifiers with exactly the same examples, and we have performed a paired two–tailored t–Student test with a significance level of 95% in order to compare the results of the classifiers.

3.1 Classifiers

To study the importance of the distance function in this kind of methods we consider six different measures: the three (normalized) basic distances of the geometry (Euclidean, Manhattan and Chebychev) and their "correlated" variants, where the importance of each feature is weighted by the correlation of the feature with the class. Basically we consider the first six classifiers shown in Table 1 above. The classifiers that do not use the heuristics (A,B,D) employ the Euclidean distance.

The (weighted vote) k–NN, ε–ball and ε–ball^{k-NN} classifiers have one or two parameters that control their behavior. To estimate the value of the parameters we have performed a separate 10–CV test with each training set. In k-NN classifiers we have considered the odd values 1, 3, 5,..., 49 for k; in ε–ball and ε–ball^{k-NN} methods we have considered 30 values in the [0,2] interval for ε. In ε–ball^{k-NN} methods we have considered odd values 1, 3, 5,..., 19 for k; and we have distinguished the special case ε–ball^{1-NN} where $k = 1$.

Table 2: Databases from the UCI–Repository used in the experiments.

					No. of attributes	
Index	Code	Domain	Size	Classes	Numeric	Symbolic
1	IR	iris plant	150	3	4	0
2	WI	wine recognition	178	3	13	0
3	PI	PIMA diabetes	768	2	8	0
4	GL	glass identification	214	6	9	0
5	CL	Cleveland	303	5	5	8
6	GD	Granada digits	1000	10	256	0
7	SN	sonar	208	2	60	0
8	LD	liver disorder	345	2	6	0
9	ZO	zoo	101	7	1	15
10	TT	tic–tac–toe	958	2	0	9
11	L7	led 7	5000	10	0	7
12	L24	led 24	5000	10	0	24
13	W21	waveform–21	5000	3	21	0
14	W40	waveform–40	5000	3	40	0
15	F1	solar flare 1	1066	8	0	10
16	F2	solar flare 2	1066	6	0	10
17	F3	solar flare 3	1066	3	0	10
18	SO	soybean	47	4	35	0
19	LR	letter recognition	20000	26	16	0

In the heuristics, to select the distance measure, we have also performed a separate 10–CV test with each training set.

In short, we have tested a great amount of combinations and spent a lot of computational time to obtain these results.

3.2 Data Sets

We have used 68 data sets: 18 well known data sets from the UCI–Repository [8], a reduced version of 1,000 instances from the Granada handwritten digits[1] (Table 2), and 49 synthetic bases. The bases from the UCI–Repository are frequently used in scientific literature, facilitating comparisons with experimental results obtained by other classifiers introduced in other papers, and synthetic bases are useful to study the classifiers in a controlled environment.

The synthetic data sets are constructed *ad hoc* over the $[0,1]\times[0,1]$ square and have 500 instances. We want to study the influence of the distribution of classes in the case base, so we consider (fig. 2):

[1]The Granada handwritten digits data set has 11,000 instances, each one with 256 numeric attributes (the normalized values of a 16×16 grid), and 10 classes (the 0,...,9 digits). Each instance corresponds to a handwritten digit. This database is private and has been yielded up by IPSA (Investigación y Programas S.A.).

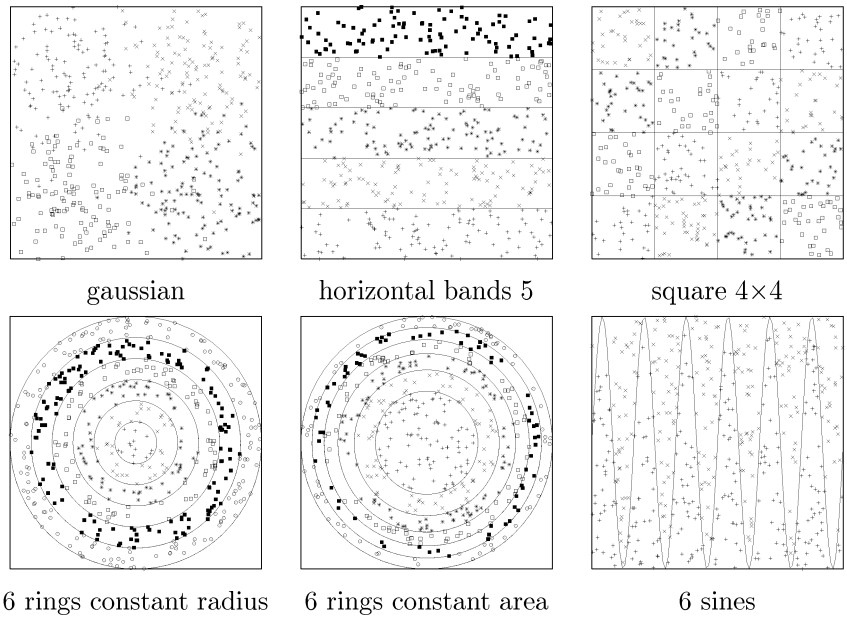


Figure 2: Some synthetic data sets. Different symbols indicate different classes and lines represent the decision boundaries.

bands (5,10,20): point class is assigned according to 5, 10 or 20 horizontal bands.

Gaussian: point class is assigned according to four Gaussian distributions with variance 0.025.

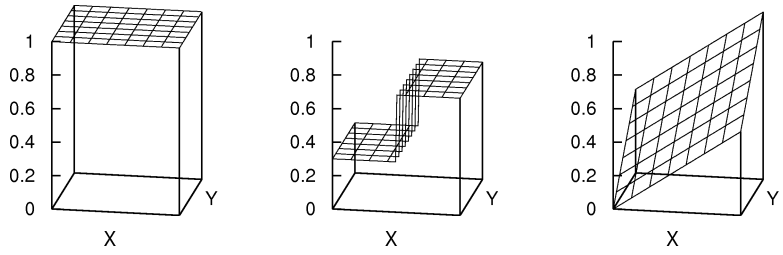
rings with constant area (3,6,9): the space is divided into 3, 6 or 9 nested rings with equal area, one class per ring, so they should have different radii. The total area of the regions has no influence and we can study the influence of the shape and the number of classes.

rings with constant radius (3,6,9): the space is divided into 3, 6 or 9 nested rings with equal radii, one class per ring.

sines (3,6,9): they have two classes, and the decision boundary is a sine curve with 3, 6 or 9 $[0,2\pi]$–intervals fitted in $[0,1]\times[0,1]$.

squares (2,4,6,8): the space is divided into a 2×2, 4×4, 6×6, or 8×8 grid. All the variants have 4 classes with the same amount of space, therefore the total area of the classes has no influence.

It seems reasonable that k–NN could have difficulties if the optimal k value is not constant along the whole space, i.e., if in some regions k must be greater than in some others. In such conditions it may be feasible for ε–ball and ε–ball^{k-NN} methods to exhibit a better behavior because in some regions k–NN will take into account too many or too few points, but ε–balls will consider only the relevant points (the points that are near enough but not the remote ones).



a. Uniform b. Half c. Progressive

Figure 3: Distribution of points in the three variants of synthetic data sets.

We have generated three variants for all the synthetic data sets (except for the Gaussian data set) to test if k–NN methods are in trouble when the optimal k value varies along the space. Densely populated regions have more points and the optimal value of k will be higher. In these three variants, the points are distributed with different probabilities across the space (fig. 3), and we get a space with different point density (and a varying optimal k value).

In the *uniform* variant the points are uniformly distributed along the space. In the *half* variant 30% of the points are in the left half of the space ($x < 0.5$) and the remaining 70% in the right half ($x \geq 0.5$): the points are distributed in two clearly different regions. In the *progressive* variant the probability of a point's acceptance is proportional to the addition of its coordinates ($x + y$): the point density progressively increases from the bottom–left corner to the top–right one.

4 The Results

We only report a summary of the results relevant in the following analysis. Full details, on each method and data set, may be obtained from Online Appendix A.

We use different ways to compare the behavior of the algorithms to avoid misleading results from a particular method. On the one hand, we have calculated for each classifier the mean accuracy, and the improvement of mean accuracy vs basic k–NN (Table 3). To facilitate the comparison between classifiers we have also provided columns with the relative position of each classifier according to each one of these measures.

On the other hand, we have applied a pairwise comparison between classifiers using a two tailored t–Student test to construct a 95% confidence interval for the difference in the accuracy rates of the algorithms (Table 4). Whenever we speak on statistically significant differences, or simply significant differences, we mean that the difference is statistically significant according to this t–Student test.

These tables show that the different ways to compare the behavior of the algorithms are very much alike, and all the proposed variants of k–NN frequently outperform the basic k–NN method.

Table 3: Accuracy of the classifiers in the experiments with all the data sets: mean accuracy, and improvement vs k–NN. "Pos." indicates the relative position of each classifier according to each measure. F attains the best position.

		Accuracy		vs k–NN	
Code	Classifier	Mean	Pos.	Δ Accur.	Pos.
A	k–NN	85.65%	8	0.00%	8
B	ε–ball	85.75%	7	0.12%	7
C	k–NN heur	87.30%	3	2.12%	3
D	ε–ball^{k-NN}	85.88%	5	0.28%	5
E	ε–ball heur	87.29%	4	2.09%	4
F	ε–ball^{k-NN} heur	87.39%	1	2.23%	1
D1	ε–ball^{1-NN}	85.81%	6	0.20%	6
F1	ε–ball^{1-NN} heur	87.37%	2	2.18%	2

Table 4: Pairwise comparison of statistically significant differences between classifiers. Each cell contains respectively the number of statistically significant wins, ties and losses between the method in the row and the method in the column. E, F and F1 significantly outperform the other classifiers frequently.

	F1	D1	F	E	D	C	B
A	0–53–15	0–63–5	0–53–15	0–53–15	0–63–5	1–53–14	1–62–5
B	2–55–11	0–67–1	2–55–11	2–55–11	0–67–1	6–48–14	
C	3–58–7	14–47–7	3–58–7	3–58–7	14–47–7		
D	2–55–11	0–68–0	2–55–11	2–55–11			
E	0–68–0	11–55–2	0–68–0				
F	0–68–0	11–55–2					
D1	2–55–11						

The most remarkable conclusion is the usefulness of characteristic C3: the proposed heuristics improves the accuracy in many data sets and the best classifiers are those that use it. Tables 3 and 4 show the improvement attained with C3 (compare A vs C, B vs E and D vs F). However, C1 and C2 provide a lower improvement. C outperforms A, B and D; thus, we'd rather use C3 over C1 and C2 together!

We must take into account that characteristic C2 includes C1. Adding C2 (and C1) shows a greater improvement than adding only C1 (compare B vs D, and E vs F), but the influence of C1 and C2 is similar. D and F improve B and E, and are also less sensitive to the choice of ε.

There is no statistically significant difference between D and D1, and between F and F1, but methods that use C2 only slightly outperform methods with C2$'$. However, there is a great difference between the computational effort required by C2 and C2$'$, because C2 implies using k–NN when the ε–ball is empty, i.e., estimating

Table 5: Results of the classifiers with the UCI–Repository data sets. "+"/"−" represents statistically significant improvement/degradation over k–NN.

	A	B	C	D	E	F
IR	96.00%	96.00%	96.00%	96.00%	96.00%	96.00%
WI	97.19%	97.19%	96.63%	97.19%	95.51%	96.63%
PI	75.39%	73.31%	76.30%	73.31%	75.52%	75.52%
GL	71.50%	71.03%	72.90%	71.50%	73.36%	75.23%
CL	57.10%	58.42%	58.09%	58.42%	57.43%	57.43%
GD	96.70%	95.70%	96.70%	96.70%	95.70%	96.70%
SN	87.02%	87.02%	88.94%	87.98%	90.87%	90.38%
LD	65.22%	63.48%	65.22%	65.51%	63.48%	65.51%
ZO	96.04%	96.04%	97.03%	97.03%	96.04%	97.03%
TT	84.24%	82.57%−	78.18%−	84.24%	80.90%	80.90%
L7	74.48%	74.36%	74.48%	74.36%	74.36%	74.36%
L24	72.34%	73.80%+	73.74%+	73.80%+	73.52%+	73.52%+
W21	85.42%	85.04%	85.42%	85.04%	85.04%	85.04%
W40	84.54%	84.62%	85.40%	84.62%	84.46%	84.46%
SO	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
F1	80.68%	82.93%+	80.58%	82.93%+	82.93%+	82.93%+
F2	96.34%	96.62%	96.25%	96.62%	96.62%	96.62%
F3	99.44%	99.53%	99.44%	99.53%	99.53%	99.53%
LR	96.02%	96.42%+	96.02%	96.42%+	96.42%+	96.42%+

the value of the parameter k while C2′ uses 1–NN. So, we recommend using C2′ instead of C2: F1 instead of F and D1 instead of D.

E, F and F1 significantly outperform the other classifiers frequently. There is no significant difference between them, but F tends to slightly outperform F1, and F1 slightly outperforms E. However, F requires by far a greater computational effort. So, we recommend using the method F1 (ε–ball^{1-NN} Heur), which significantly improves the basic k–NN frequently (15–53–0), improves the basic k–NN very frequently (45–5–18), and provides an overall improvement of 2.18% over k–NN. This is also true with the UCI data sets, where method F1 significantly improves k–NN frequently (3–16–0), improves k–NN very frequently (11–3–5) and provides an overall improvement of 0.47% over k–NN.

In short, all the proposed characteristics are useful. Clearly, the most relevant one is C3, followed by C1 and C2, and, without any prior information, we recommend using the method F1 (ε–ball^{1-NN} heur).

Tables 5, 6 and 6 show the results obtained by the classifiers with each data set[2]. Results are followed by "+" (or "−") if they show statistically significant

[2]The results with the Cleveland data set could seem abnormally lower, but we must take into account that these results are obtained considering 5 classes, while experiments with this data set have been concentrated on simply distinguishing presence from absence of heart disease.

Table 6: Results of the classifiers with the artificial data sets. "+"/"−" represents statistically significant improvement/degradation over k–NN.

	A	B	C	D	E	F
band 5u	96.20%	96.00%	99.40%+	96.00%	99.20%+	99.20%+
band 5h	95.40%	95.80%	98.80%+	95.80%	98.20%+	98.20%+
band 5p	95.40%	94.80%	98.20%+	94.80%	98.60%+	98.60%+
band 10u	88.20%	89.60%	97.00%+	89.60%	96.60%+	96.60%+
band 10h	89.80%	88.80%	97.20%+	88.80%	96.80%+	96.80%+
band 10p	87.80%	88.40%	95.80%+	88.40%	96.80%+	96.80%+
band 20u	72.60%	72.40%	93.80%+	72.60%	94.00%+	94.00%+
band 20h	71.60%	72.00%	93.20%+	72.00%	92.20%+	92.20%+
band 20p	71.60%	71.40%	91.80%+	71.60%	89.80%+	89.80%+
Gaussian	87.60%	88.20%	86.60%	88.20%	86.40%	86.40%
ring a 3u	90.60%	91.40%	93.00%+	91.40%	92.20%	92.20%
ring a 3h	91.40%	93.00%+	91.40%	93.00%+	93.00%+	93.00%+
ring a 3p	93.80%	93.60%	93.60%	93.60%	93.80%	93.80%
ring a 6u	80.80%	81.20%	80.80%	81.20%	81.20%	81.20%
ring a 6h	80.00%	81.20%	80.00%	81.20%	81.20%	81.20%
ring a 6p	82.80%	82.20%	82.40%	82.20%	81.80%	81.80%
ring a 9u	70.00%	71.20%	70.00%	71.20%	69.60%	69.60%
ring a 9h	70.60%	69.80%	70.20%	69.80%	69.80%	70.20%
ring a 9p	72.60%	72.60%	72.60%	72.60%	72.60%	72.60%
ring r 3u	95.40%	95.00%	96.20%	95.00%	95.60%	95.60%
ring r 3h	95.80%	95.00%	95.20%	95.00%	95.60%	95.60%
ring r 3p	94.20%	94.40%	94.60%	94.40%	94.60%	94.60%
ring r 6u	88.00%	88.20%	88.20%	88.20%	88.60%	88.60%
ring r 6h	89.80%	91.20%+	89.80%	91.20%+	91.20%+	91.20%+
ring r 6p	85.60%	85.60%	84.80%	85.60%	85.80%	85.80%
ring r 9u	82.60%	81.60%	84.40%	81.60%	82.80%	82.80%
ring r 9h	78.20%	78.60%	79.20%	78.60%	79.60%	79.60%
ring r 9p	77.00%	77.80%	78.20%	77.80%	79.20%	79.20%
sines 3u	93.40%	92.60%	93.40%	92.60%	92.60%	92.60%
sines 3h	91.40%	92.00%	90.60%	92.00%	90.80%	90.80%
sines 3p	93.00%	91.80%	94.40%	91.80%	93.80%	93.80%
sines 6u	85.80%	86.20%	86.00%	86.20%	86.60%	86.60%
sines 6h	83.60%	84.00%	84.20%	84.00%	84.80%	84.80%
sines 6p	83.80%	84.40%	84.60%	84.40%	85.00%	85.00%
sines 9u	75.80%	75.80%	75.80%	76.00%	75.80%	76.00%
sines 9h	78.40%	78.80%	78.60%	78.80%	78.60%	78.60%
sines 9p	76.60%	76.40%	79.80%+	77.20%	77.40%	77.40%

continue on next page...

Table 6: (cont.) Results of the classifiers with the artificial data sets. "+"/"−" represents statistically significant improvement/degradation over k–NN.

	A	B	C	D	E	F
square 2u	97.60%	97.20%	97.60%	97.20%	97.20%	97.20%
square 2h	96.80%	97.00%	96.80%	97.00%	97.00%	97.00%
square 2p	98.20%	98.60%	98.20%	98.60%	98.60%	98.60%
square 4u	93.80%	94.40%	94.40%	94.40%	94.20%	94.20%
square 4h	93.80%	93.80%	93.60%	93.80%	93.40%	93.60%
square 4p	93.20%	92.60%	93.60%	92.60%	92.00%	92.00%
square 6u	87.20%	87.20%	86.40%	87.20%	87.00%	87.00%
square 6h	85.80%	86.80%	87.80%	86.80%	87.80%	87.80%
square 6p	84.60%	85.20%	84.60%	85.20%	85.20%	85.20%
square 8u	75.80%	76.40%	75.80%	76.40%	76.40%	76.40%
square 8h	80.80%	81.00%	82.40%+	81.00%	82.80%+	82.80%+
square 8p	84.00%	83.80%	84.00%	83.80%	83.80%	83.80%

improvement (or degradation) over basic k–NN, according to a paired two-tailored t–test at a 95% confidence level.

We are mainly interested in comparing these methods to establish their weaknesses and strong points and make recommendations about their usefulness, and not in their concrete results or whether they are better or worse than other approaches. We are aware of the fact that he comparison with other classification methods is beyond the scope of this paper, which is left for future work because we focus only on k–NN in this paper. However, if you are interested in comparisons with other classification methods, we think that the experiments' methodology is standard enough to let you consult results from other papers and probably establish good comparisons. In any case, we wish to emphasize here that these classifiers will always obtain a 100% of accuracy with the training set because they retain all the cases they learn. In future work we wish to study forgetting mechanisms to retain only a subset of the training set. Then, studying the accuracy with the training set (and the size of the retained set) will make sense.

Traditionally, methods based on distances are supposed to be bad handling noise and irrelevant features, but the results of led7 and led24, and waveform–21 and waveform–40 are comparable, in spite of the 17 and 19 irrelevant features. The results with these noisy data sets are around their Bayes optimal classification rate (74% and 86% for the led display and waveform domains respectively).

The optimal k value relies on the data set itself and on the distance measure used. Of course, if we use the optimal k value of the uniform distribution with the half and progressive distributions then the accuracy drops. But, in the latter distributions, the optimal k value is closer to the optimal value in dense regions. As we modify the distribution of points along the space, the optimal k value changes in such a way that it tends to classify better denser regions. In other words, k–NN

Table 7: Mean accuracy of the classifiers in the uniform, half and progressive variants. "Δ" shows the variation with respect to the uniform variant.

		Unif.	Half		Prog	
Code	Classifier	Mean	Mean	Δ	Mean	Δ
A	k–NN	85.86%	85.83%	–0.03%	85.89%	+0.03%
B	ε–ball	86.03%	86.18%	+0.15%	85.85%	–0.18%
C	k–NN heur	88.26%	88.06%	–0.20%	88.20%	–0.06%
D	ε–ball^{k-NN}	86.05%	86.18%	+0.13%	85.91%	–0.14%
E	ε–ball heur	88.10%	88.30%	+0.20%	88.05%	–0.05%
F	ε–ball^{k-NN} heur	88.11%	88.34%	+0.23%	88.05%	–0.06%
D1	ε–ball^{1-NN}	86.04%	86.18%	+0.14%	85.86%	–0.18%
F1	ε–ball^{1-NN} heur	88.10%	88.34%	+0.24%	88.05%	–0.05%

prefers to get the right class in dense regions at the expense of sparse ones, but reaching a commitment between population and the optimal k value of different regions along the space. The same stands for ε–ball methods.

Another interesting question is when we should use a k–NN or an ε–ball classifier. We have found some evidence about the weakness of k–NN when the distribution of points is not constant along the space (and therefore the optimal k value varies). Table 7 shows that k–NN heur slightly outperforms ε–ball in the uniform and progressive variants, but it is worse in the half variant (0–14–2). In the half variant, k–NN methods (A and C) show certain weakness, they reduce their results with respect to the uniform variant, whereas ε–ball methods increase their accuracy. k–NN methods have more problems than ε–ball if there are big areas with a very different distribution of points, and a very different optimal k value. On the other hand, k–NN is preferable if the optimal k value is constant along the space (uniform variant), or if it varies progressively (progressive variant), but without shaping clear regions of very different values. k–NN heur is clearly preferable in the band data sets, and ε–ball methods in the ring data sets.

The results of k–NN methods are better than expected in the half and progressive variants. If we employ in these two variants the k value selected in the uniform variant, then k–NN methods reduce drastically their results. But k-NN methods tend to select k values greater than in the uniform variant. We think that the surprisingly good results of k–NN methods in half and progressive variants are due to the previously analyzed commitment between the population and the optimal k value of different regions along the space. Perhaps, in these artificial data sets the examples are still dense enough for finding a suitable k that gives a good classification performance. In this case, experiments with different artificial data sets must be done in order to clarify the circumstances under one would expect ε–ball methods to perform well.

5 Conclusions

In this paper we have analyzed some variants of the basic k–NN classification method, based on the addition of up to three characteristics. We have studied the usefulness of these characteristics and we have made an exhaustive comparison with k–NN (we have used six different distance functions, and tested the classifiers with 68 data sets). The three proposed characteristics have revealed useful, and the proposed variants tend to outperform the results of the basic k–NN.

As expected, the choice of the distance measure is important, decisively affects the classifier performance, independently of the family of classifiers, and relies on the data set himself. In data sets with irrelevant features a sensitive distance function is useful. This could be achieved by using information about the relative importance of the features.

We have noticed that there is a relationship between the accuracy obtained by the distances (1–NN) and the accuracy of other methods based in those distances. Considering that 1–NN methods are relatively fast, we proposed the following heuristics: *you should train and test 1–NN methods with a set of different distances and use the desired method based on distances (k–NN, ε–ball, ε–ball$^{k-NN}\dots$) with the distance which offers the best accuracy with 1–NN.* The heuristics, which is used by the best classifiers, increases accuracy. In order to speed up the distance selection (specially when we propose a large number of them) it is feasible to use the ideas of the racing algorithms [27]: you should test the models in parallel, quickly discard those that are clearly inferior using statistical bounds, and concentrate the computational effort on differentiating among the best models.

ε–ball^{k-NN} heur significantly outperforms k–NN (15–53–0), k–NN heur (7–58–3) and ε–ball (11–55–2) methods frequently. There is no statistically significant difference between ε–ball^{k-NN} heur and ε–ball heur (0–68–0), but ε–ball^{k-NN} heur attains slightly better results and it is also less sensitive to the choice of ε. Generally, when there is no statistically significant difference, ε–ball^{k-NN} is the method that attains better results with most of the data sets. There is no statistically significant difference between ε–ball^{k-NN} heur and ε–ball^{1-NN} heur. The former attains slightly better results, but it requires more time.

The time required by the classifiers is similar (except for ε–ball^{k-NN} and ε–ball^{k-NN} heur). So, as a general rule, and without any prior information, we recommend using the proposed ε–ball^{1-NN} heur, the method that achieves better performace with similar time consumption.

We have found some evidence about the weakness of k–NN when the distribution of points is not constant along the space (and therefore the optimal value of k varies). k–NN methods have more problems than ε–ball if there are big areas with a very different distribution of points. k–NN heur is clearly preferable in the band data sets, and ε–ball methods in the ring data sets.

For future work we are interested in the study of new distance measures that better fit the peculiarities of the data set. We plan to study mechanisms to reduce the memory usage with information–compacting techniques, retain useful cases and forget the less useful ones, in order to lead up to instance–based learning (IBL) [4, 2] and case–based reasoning (CBR) [22, 23, 1, 3].

6 Acknowledgements

We would like to acknowledge support for this work from the Dirección General de Investigación of Spain by the project grant TIN2006-03122.

Online Appendix A: Detailed Results of the Experiments

Full details, on each algorithm and data set, may be obtained as a spreadsheet from *http://bahia.ugr.es/∼laguia/Summary_acbknn.xls*.

References

References

[1] A. Aamodt and Enric Plaza. Case–based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7:39–59, 1994.

[2] David W. Aha. Tolerating noisy, irrelevant and novel attributes in instance–based learning algorithms. *International Journal of Man–Machine Studies*, 36:267–287, 1992.

[3] David W. Aha. The omnipresence of case–based reasoning in science and application. *Knowledge–Based Systems*, 11:261–273, 1998.

[4] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance–based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[5] Hussein Almuallim and Thomas G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI–91)*, volume 2, pages 547–552, Anaheim, California, 1991. AAAI Press.

[6] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997. Special Issue on "Lazy Learning".

[7] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning for control. *Artificial Intelligence Review*, 11:75–113, 1997. Special Issue on "Lazy Learning".

[8] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[9] Arvin L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1–2):245–271, 1997.

[10] Léon Bottou and Vladimir Vapnik. Local learning algorithms. *Neural Computation*, 4:888–900, 1992.

[11] Pablo Carmona, Juan Luis Castro, Castro José J., and Manuel Laguía. *Learning default fuzzy rules with general and punctual exceptions*, volume 129 of *Studies in Fuzziness and Soft Computing*, pages 302–337. Springer–Verlag, 2003.

[12] Scott Cost and Steven Salzberg. A weighted nearest algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.

[13] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:21–27, 1967.

[14] Walter Daelemans, Antal van den Bosch, and Ton Weijters. Igtree: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423, April 1997. Special Issue on "Lazy Learning".

[15] B. V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques.* IEEE Computer Society Press, 1991.

[16] Pedro Domingos. Context–sensitive feature selection for lazy learners. *Artificial Intelligence Review*, 11:227–253, 1997. Special Issue on "Lazy Learning".

[17] E. Fix and J. L. Hodges, Jr. Discriminatory analysis, nonparametric discrimination, consistency properties. Technical report, Randolph Field, TX: United States Air Force, School of Aviation Medicine, 1951. Technical Report 4.

[18] Jerome H. Friedman. Flexible metric nearest neighbor classification. Technical report, Dept. of Statistics, Stanford University, 1994. Available by anonymous FTP from playfair.stanford.edu (see /pub/friedman/README).

[19] A. D. Griffiths and D. G. Bridge. Towards a theory of optimal similarity measures. In *Third UK Workshop on Case–Based Reasoning (UKCBR–97).* Springer–Verlag, September 1997.

[20] Trevor Hastie and Robert Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:607–616, 1996.

[21] Nicholas Howe and Claire Cardie. Examining locally varying weights for nearest neighbor algorithms. In *Case–Based Reasoning Research and Development: Second International Conference on Case-Based Reasoning (ICCBR–97).* Springer–Verlag, 1997.

[22] Janet L. Kolodner. An introduction to case–based reasoning. *Artificial Intelligence Review*, 6:3–34, 1992.

[23] Janet L. Kolodner. *Case–Based Reasoning.* Morgan Kaufmann, 1993.

[24] Manuel Laguía. *Razonamiento Basado en Casos aplicado a Problemas de Clasificacin.* PhD thesis, Dept. Ciencia de la Computación e Inteligencia Artificial, Universidad de Granada, November 2003.

[25] Manuel Laguía and Juan Luis Castro. Similarity relations based on distances as fuzzy concepts. In *Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT–2001)*, 2001.

[26] Charles X. Ling and Handong Wang. Computing optimal attribute weight settings for nearest neighbor algorithms. *Artificial Intelligence Review*, 11:255–272, 1997. Special Issue on "Lazy Learning".

[27] Oded Maron and Andrew W. Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, 11:193–225, 1997. Special Issue on "Lazy Learning".

[28] E. Plaza, R. López, and E. Armengol. On the importance of similitude: An entropy-based assessment. In *Third European Workshop on Case-Based Reasoning (EWCBR–96)*. Springer–Verlag, 1996.

[29] Francesco Ricci and Paolo Avesani. Data compression and local metrics for nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):380–384, 1999.

[30] M. M. Ritcher. Classification and learning of similarity measures. In *16. Jahrestagung der Gesellschaft für Klassifikation (GFKL–92)*. Springer–Verlag, 1992.

[31] M. M. Ritcher. On the notion of similarity in case-based reasoning. In G. del Viertl, editor, *Mathematical and Statistical Methods in Artificial Intelligence*, pages 171–184. Springer–Verlag, 1995.

[32] Steven Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, 6:251–276, 1991.

[33] C. Stanfill and D. Waltz. Towards memory–based reasoning. *Communications of the ACM*, 29:1213–1228, 1986.

[34] S. M. Weiss and C. A. Kulikowski. *Computer Systems that Learn.* Morgan Kaufmann, 1991.

[35] Dietrich Wettschereck. *A Study of Distance-Based Machine Learning Algorithms.* PhD thesis, Oregon State University, 1994.

[36] Dietrich Wettschereck, David W. Aha, and Takao Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314, April 1997. Special Issue on "Lazy Learning".

[37] Dietrich Wettschereck and Thomas G. Dietterich. Locally adaptive nearest neighbor algorithms. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 184–191. Morgan Kaufmann Publishers, Inc., 1994.

[38] Dietrich Wettschereck and Thomas G. Dietterich. An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, 19:5–28, 1995.

[39] D. Randall Wilson and Tony R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Reseach (JAIR)*, 6:1–34, 1997.

[40] Jianping Zhang, Yee-Sat Yim, and Junming Yang. Intelligent selection of instances for prediction functions in lazy learning algorithms. *Artificial Intelligence Review*, 11:175–191, 1997. Special Issue on "Lazy Learning".