International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 17, No. 8 (2003) 1–14

World Scientific
www.worldscientific.com

A SIMPLE LOCALLY ADAPTIVE NEAREST NEIGHBOR RULE WITH APPLICATION TO POLLUTION FORECASTING

RICHARD NOCK*

Grimaag-Département Scientifique Interfacultaire, Université des Antilles-Guyane, Campus de Schoelcher, BP 7209, 97275 Schoelcher, France
rnock@martinique.univ-ag.fr

MARC SEBBAN†

Eurise-Département d'Informatique, Université Jean Monnet, 23, Rue du Docteur Paul Michelon, 42023 Saint-Etienne Cedex 2, France
Marc.Sebban@univ-st-etienne.fr

DIDIER BERNARD

Laboratoire de Physique de l'Atmosphère Tropicale, Université des Antilles-Guyane, Campus de Fouillole, 97159 Pointe-À-Pitre, France
dbernard@univ-ag.fr

In this paper, we propose a thorough investigation of a nearest neighbor rule which we call the "Symmetric Nearest Neighbor (sNN) rule". Basically, it symmetrises the classical nearest neighbor relationship from which are computed the points voting for some instances. Experiments on 29 datasets, most of which are readily available, show that the method significantly outperforms the traditional Nearest Neighbors methods. Experiments on a domain of interest related to tropical pollution normalization also show the greater potential of this method. We finally discuss the reasons for the rule's efficiency, provide methods for speeding-up the classification time, and derive from the sNN rule a reliable and fast algorithm to fix the parameter k in the k-NN rule, a long-standing problem in this field.

Keywords: Machine learning; case-based reasoning; neighborhood; pollution forecasting.

1. Introduction

The Nearest Neighbor (NN) rule is one of the simplest and oldest classification techniques related to Case-Based Reasoning (CBR). It uses a set of observations, also called reference sample, $\mathbf{S}$, from a n-dimensional metric space $\mathbf{X}$ to classify members of $\mathbf{X}$ in one of c classes based on the classification of the elements of

*Author for correspondence.
†This work was done while the author was with the Université des Antilles-Guyane.

S. For each $x \in \mathbf{X}$, we choose the element $x' \in \mathbf{S}$ that is nearest to x (with respect to a given metric), and assign to x the same class as x'. In case of ties, a tie breaking rule chooses randomly for x' one of the nearest neighbors. This rule is obviously encompassed by the following one: the nearest neighbor of x can be replaced by a voting set of k of the nearest instances of x, the k-nearest neighbors (k-NN), a majority vote giving the class of x. Numerous experiments record that simple nearest neighbor methods are competitive with very sophisticated classifiers on challenging problems.[8] Since the first appearance of a k-NN-like rule,[6] domains such as memory-based, instance-based, local, or lazy learning have contributed to its widespread use, and much work was done in and around the NN paradigm to improve the rule. In particular, recent publications[13] have focused in locally adapted neighborhoods, transforming the spherical neighborhoods centered on some instance x into ellipsoidal neighborhoods, elongated to the directions of constant posterior probability, and narrowed in the directions of greatest variations. Their objective was to reduce the curse of dimensionality which states that, as dimensionality increases, the average or median neighborhood size on the uniform distribution shrinks for the *nearest neighbor* only as $|\mathbf{S}|^{1/n}$, thereby leading to large neighborhoods and possibly less reliable classification. We are going to show that a very simple algorithm of low complexity, also locally adapting neighborhoods, improves the classification of the nearest neighbor rule on most of 29 benchmark problems.

In the "Symmetric Nearest Neighbor rule", k-sNN, we vote for some x the points in **S** which could belong to the k nearest neighbors of x, *and* the points in **S** for which x could belong to the k nearest neighbors. For example, if some x could vote for some y in the k-NN rule, then y *votes* for x in the k-sNN rule, even if y does not vote for x in the k-NN rule.

Figure 1 shows a simple case with 6 points in the plane and the neighborhoods corresponding to the 1-NN and 1-sNN rules for two points x_1 and x_2. Arcs (x, y) in the first part indicate that x can vote for y in the 1-NN rule. In the 1-sNN rule, arcs are replaced by edges, and the neighborhood of x_1 contains three points instead of

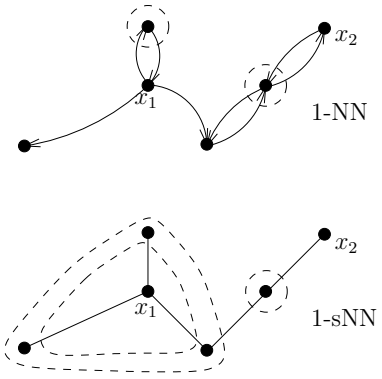


Fig. 1. A simple example showing neighborhoods of the 1-NN and 1-sNN rules for x_1 and x_2.

one in the 1-NN rule. Our motivation to address this approach and compare it to the traditional k-NN rule is simple: in the set $\mathbf{S} \cup \{x\}$, the k-sNN rule increases the number of voting neighbors compared to the k-NN rule, but *without* actually creating "new" votes: for any y voting for x in the k-sNN rule, either y could vote for x or x could vote for y in the k-NN rule. This is a way to increase the number of voters while still working on the neighborhood graph of the k-NN rule, thus keeping the same maximal neighborhood size, with possibly more reliable classification at least for reasonable values of k. The motivation of this paper is precisely to test this assertion.

These additional neighbors, which we also call *adjuncts*, were previously used in Machine Learning (ML) to remove irrelevant instances from $\mathbf{S}$, but not in subsequent classification phases.[19] Also, the graph of *symmetric neighborhoods* was previously used in the field of image processing, but in that case for filter design, and without link to ML purposes.[12]

Compared to Ref. 13, we think of our approach as providing a conceptually simple solution at low computational cost, for which experiments tend to show that it does not trade accuracy for simplicity. To demonstrate this fact, the remainder of the paper is mostly experimental. First, we present results on 29 datasets, most of which come from the UCI repository of ML database.[2] These are complemented by tests on a problem related to pollution normalization in Tropical Physics for which ML represents a solution of great potentiality (yet no real investigation in official texts has been proposed to date). In the last part, we extend the discussion of the sNN rule's reliability, we present methods to speed-up the classification of new instances, and provide tests on a particular longstanding problem to which the sNN neighborhood graph brings an accurate solution at affordable complexity: the choice of k for the k-NN rule.

2. Experimental Results on the sNN Rule

2.1. *The 1-sNN rule beats NN rules*

In this section, we compare the 1-NN rule with its equivalent tNN and sNN rules (the k-tNN rule where "t" stands for total, is the k-NN rule with the slight difference that when there are ties, *all* points at distance not exceeding that of the kth nearest neighbor vote). Table 1 presents the results that were obtained. Four accuracies are given for each dataset, computed using Leave-One Out Cross Validation (LOOCV). The first third are the accuracies of the 1-NN, 1-tNN and 1-sNN rules respectively, and the fourth is the accuracy of the k-NN classifier with k equal to the average neighborhood size (without the instance to be classified) of the 1-sNN classifier. This rule is denoted $\bar{k}_{sNN}$-NN; it is used to compare the 1-sNN with a conventional NN rule with a similar number of neighbors (on average). The last columns display the average neighborhood sizes for 1-tNN and 1-sNN over all runs and all instances (remark that this represents the average degree of the neighborhood graphs over $\mathbf{S}$). Two major factors can improve the performances of the 1-sNN rule compared to

Table 1. Accuracies on LOOCV for five rules. Average resulting neighborhood's sizes are indicated for 1-tNN and 1-sNN. When accuracies of either 1-NN, 1-tNN or 1-sNN are greater than or equal to each of the *four* accuracies, the results are bold faced (these represent very good results on corresponding problems).

	Accuracy				$\bar{k}$	
Dataset	1-NN	1-tNN	1-sNN	$\bar{k}_{sNN}$-NN	1-tNN	1-sNN
Audiology	78.30	78.30	**78.80**	74.80	1.83	2.85
Australian	77.70	77.50	78.00	80.10	1.77	2.62
Balance	79.20	80.30	80.50	81.00	6.40	6.80
Bigpole	65.00	64.40	**65.40**	62.80	3.75	4.64
Breast-W	95.60	96.30	96.40	96.90	1.66	7.42
Car	89.20	88.40	**99.30**	97.70	2.00	7.28
Crx	79.00	79.00	**80.40**	78.40	1.94	2.93
Dermatology	90.00	89.30	**90.40**	90.00	1.12	1.69
Echo	54.20	54.70	**63.40**	58.80	2.40	3.42
German	68.10	71.70	**72.50**	72.30	3.23	4.94
Glass2	66.30	66.90	**74.20**	73.60	6.03	7.37
Hard	**58.20**	53.10	48.80	50.40	2.82	3.76
Heart	75.00	75.00	75.10	76.66	1.86	2.73
Hepatitis	83.20	84.50	**86.50**	81.30	1.84	2.90
Horse Colic	71.20	**71.70**	70.90	70.90	1.18	1.95
Ionosphere	84.30	83.20	**84.90**	78.90	4.16	6.41
LED Even	87.40	87.20	87.60	88.80	18.77	19.73
LED Even+17	65.50	68.50	72.00	75.50	1.86	2.86
Lung Cancer	53.10	50.00	**62.50**	53.10	1.09	1.81
Monks 1	66.40	98.60	**99.80**	97.00	7.50	7.80
Pima	67.10	68.90	69.80	70.40	9.32	10.73
Tic tac toe	75.00	76.00	**84.00**	82.00	1.81	6.32
Vehicle	65.00	71.30	**72.00**	71.00	6.35	7.53
Waves	**76.80**	75.60	75.20	76.80	1.44	2.34
WhiteHouse	92.40	92.00	**92.90**	91.30	3.23	4.92
WhiteH w/o PP	88.74	88.05	88.05	89.20	3.30	5.02
XD6	75.70	81.50	80.70	81.80	3.99	5.36
Wines	85.96	85.96	**87.64**	85.96	1.21	1.75
Zoo	**98.02**	97.03	**98.02**	91.09	3.24	4.02
Average	76.26	77.76	**79.85**	78.57	3.69	5.15

the 1-NN rule's. The first is the increasing number of voting instances, the second is the better "construction" (or choice) of the neighbors. The fourth rule was run in order to reduce the former effect, and evaluate the latter, that seems much more interesting to us.

The results of the 1-tNN rule displayed in Table 1 show that the simple increase of the number of neighbors compared to the 1-NN rule tends indeed to favor greater accuracies, yet the difference is not significant: a sign test cannot separate the two first columns even at risk $\alpha \leq 30\%$. Moreover, when removing the atypical Monks 1 problem for which the 1-NN rule performs poorly, 1-tNN beats the 1-NN rule only by 0.4% on average. The results of the 1-sNN rule are much more striking. 1-sNN beats almost systematically the 1-NN and 1-tNN rules together. One feature

appears more interesting by looking at the bold-faced results of Table 1. The entries were chosen to represent excellent results of either the 1-NN, 1-tNN or 1-sNN rules, in the cases where they achieve results at least as good as all other algorithms, including the NN rule ran with the average ks of the 1-sNN rule. Almost all bold-faced results are in the column 1-sNN, and many of the 1-sNN results are actually bold-faced. If the four algorithms were performing evenly, each would obtain the best result on average $29/4 \approx 7$ times. A sign test (third column versus all the others) reveals a threshold probability of minute order (10^{-27}), i.e. an infinitesimal proportion of configurations appear to be at least as in favor of the 1-sNN as the one presented by Table 1.

All these observations exhibit a real performance of the 1-sNN algorithm. The problem for which the increase is the most dramatic is Lung Cancer. In this case, the use of the 1-sNN rule buys an increase of more than 9% in the accuracy compared to each of the three other rules. Since the 1-sNN builds neighborhoods of larger size compared to 1-NN and 1-tNN, we might conclude that this represent the main reason for the accuracy's increase. Further remarks show that this is a loose explanation. The head-to-head comparison of the 1-tNN and 1-sNN shows that, if we except the atypical Monks 1 dataset, the choice of the 1-tNN instead of the 1-NN buys on average an increase of 0.4% in accuracy by using 2.69 more neighbors, whereas the choice of the 1-sNN instead of the 1-NN buys on average an increase of more than 2.5% in accuracy by using only 4.15 more neighbors. Therefore, the difference of less than 1.5 neighbors between 1-tNN and 1-sNN accounts for more than 2% increase in the accuracy, i.e. much more than the comparison 1-tNN versus 1-NN. Finally, the $\bar{k}_{sNN}$-NN rule, at equivalent neighborhood size, is beaten by $\approx 1.3\%$ by the 1-sNN rule. This phenomenon can be explained by the fact that the distance between one point voting for another in the 1-sNN rule is not greater than the largest one in the 1-NN, whereas the largest such distance in the $\bar{k}_{sNN}$-NN rule, which uses on average the same number of neighbors as the 1-sNN rule, can be much larger than in the 1-NN, and therefore can provide less reliable votes as distances increase. Altogether, these observations exhibit the more accurate choice of the additional neighbors in the 1-sNN rule compared to the classical NN rules.

2.2. *Average comparisons for many k*

In this section, we extend the results of the preceding section to the cases where the value of k fluctuates to large values. To compare reliably the three rules k-NN, k-tNN and k-sNN, we choose the following experimental set-up. On each dataset of Table 2, we ran each rule with the values of k set from 1 to large values $90 \leq k \leq 100$. For the datasets with too few examples to reach such large values, the algorithms were ran up to the maximum possible value for k. The accuracies on each dataset were computed by LOOCV for each possible value for k.

The first three columns of Table 2 display the respective results of the corresponding NN, tNN and sNN rules. Each result is given as a form ($\text{Mean}_{\text{Standard dev.}}$,

Table 2. Accuracies on LOOCV for 29 datasets. The average accuracy of sNN is bold-faced because a paired t-test reveals a threshold risk of order $< 1/10\%$ when comparing it w.r.t. NN and tNN (see text).

	Accuracy$_\sigma$		
Dataset	NN	tNN	sNN
Audiology	$65.45_{4.63}$	$65.96_{4.48}$	$68.79_{2.48}$
Australian	$77.89_{0.88}$	$77.88_{0.83}$	$77.68_{1.27}$
Balance	$85.35_{1.11}$	$86.48_{1.53}$	$86.59_{1.48}$
Bigpole	$64.73_{1.23}$	$64.75_{1.34}$	$65.43_{1.19}$
Breast-W	$95.89_{0.38}$	$96.03_{0.45}$	$96.38_{0.48}$
Car	$92.71_{1.46}$	$94.86_{1.82}$	$94.90_{1.82}$
Crx	$78.69_{1.29}$	$78.80_{1.13}$	$80.67_{0.89}$
Dermatology	$54.21_{7.88}$	$54.16_{7.31}$	$56.51_{6.98}$
Echo	$66.51_{1.55}$	$67.08_{1.43}$	$66.79_{1.83}$
German	$72.66_{0.96}$	$72.46_{0.08}$	$72.57_{1.13}$
Glass2	$66.71_{4.95}$	$67.91_{2.59}$	$69.37_{2.41}$
Hard	$49.31_{4.16}$	$48.82_{4.78}$	$48.97_{5.44}$
Heart	$75.32_{1.04}$	$75.14_{1.07}$	$75.36_{0.89}$
Hepatitis	$79.15_{0.64}$	$79.26_{5.63}$	$79.26_{0.89}$
Horse Colic	$68.77_{2.40}$	$68.72_{2.25}$	$71.16_{2.61}$
Ionosphere	$68.50_{6.49}$	$68.38_{6.81}$	$72.59_{3.21}$
LED Even	$89.59_{0.43}$	$89.12_{0.27}$	$89.18_{0.60}$
LED Even+17	$78.58_{1.22}$	$79.51_{1.40}$	$79.48_{1.38}$
Lung Cancer	$25.18_{13.1}$	$26.32_{14.4}$	$34.07_{16.3}$
Monks 1	$79.11_{8.01}$	$78.86_{10.0}$	$78.86_{8.40}$
Pima	$71.24_{0.72}$	$70.44_{0.90}$	$71.20_{0.64}$
Tic tac toe	$72.67_{4.95}$	$71.49_{5.78}$	$72.77_{8.31}$
Vehicle	$72.53_{0.64}$	$73.31_{0.56}$	$72.85_{0.83}$
Waves	$82.10_{1.21}$	$82.21_{1.17}$	$82.88_{1.57}$
WhiteHouse	$89.67_{1.02}$	$89.84_{1.10}$	$90.86_{1.24}$
WhiteH w/o PP	$86.94_{0.74}$	$86.96_{0.83}$	$88.23_{0.79}$
XD6	$80.71_{1.17}$	$81.33_{1.49}$	$80.87_{2.61}$
Wine	$55.46_{13.4}$	$55.41_{13.6}$	$59.02_{13.1}$
Zoo	$55.43_{16.5}$	$56.54_{17.9}$	$60.75_{18.9}$
Average	72.45	72.69	**73.93**

i.e. σ is put in index) computed over all runs. These results are useful to compare the relative behaviors of the three rules together, but they are obviously not suited to make performance comparisons with other algorithms: the values of k ranged towards values far larger than one would fix to maximize the accuracy.

A paired sign test was carried out between each of the three accuracy's columns. For the comparison k-NN versus k-tNN, it reveals that they cannot be separated up to risk $\alpha \leq 20\%$. This, along with the fact that their average accuracies over the

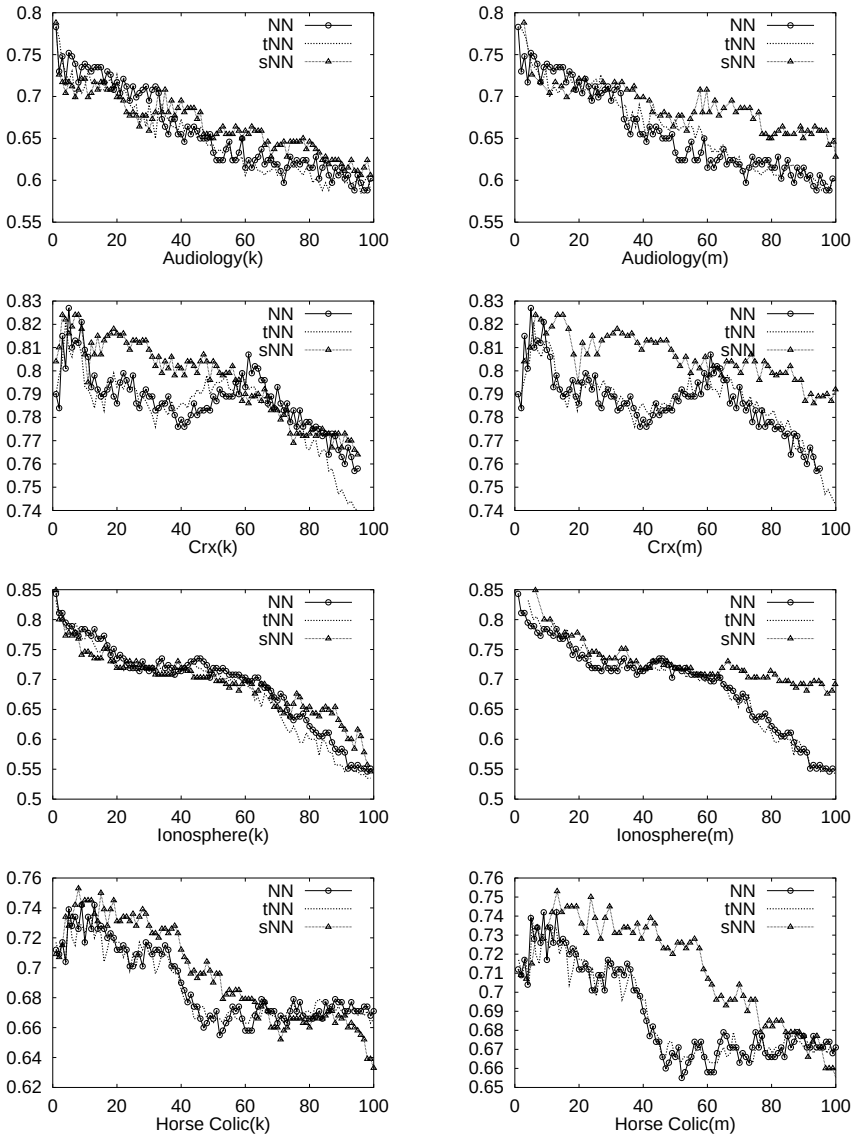


Fig. 2. Four representative examples of plots using the two scales on k and $\mu(k)$, respectively noted (k) and (m) for each dataset (vertical axes are the accuracies).

29 datasets are almost identical, shows that the two rules are equivalent. The four datasets shown in Fig. 2 bring particularly good visual evidence of this fact: the curves of these two rules are quite similar even on the two scales k and $\mu(k)$.

It is a bit different for the k-sNN rule. A sign test now gives threshold risks of $\approx 0.0012\%$ and 0.00005% to separate the k-sNN rule from respectively the k-NN and the k-tNN rule. As shown in Table 2, the primary reason for the test to be significant

is of course that the average accuracy is significantly in favor of the k-sNN rule over the 29 datasets. To explain these results, three reasons can be pointed out. The first two reasons are obviously the same as for the preceding section. The justification for the better choice of the additional neighbors in the k-sNN algorithm is now visually evident from Fig. 2, when looking at the $\mu(k)$ curves. For the Horse Colic dataset, the curve of the sNN rule is clearly located over the two other curves. For particular points of $45 \leq \mu(k) \leq 60$, the accuracies of the NN and tNN rules are similar, but they are beaten by the sNN rule by more than 5% on average, at equivalent neighborhood sizes. The third reason explaining the better results of the k-sNN rule, which were observed separately for a large majority of the datasets, is its resistance against the accuracy's degradation when the average neighborhood size increases. In Ionosphere (see Fig. 2), when $\mu(k)$ tends to large values, the performances of the sNN rule degrades about 8% compared to the small values for $\mu(k) \leq 20$, whereas for the two other rules, the degradation is greater than 20%. This phenomenon is less pronounced but still visible on the Audiology dataset in Fig. 2. It is all the more important for a NN rule to be robust to the variations of k, as there is no formal rule to fix k. To conclude this part, we have carried out a paired t-test in Table 2 to compare each algorithm versus each of the other two over all runs for k. The limit risk α for the difference tNN versus NN is only 3.6% (therefore, with a reasonable risk of few percents, we would *keep* the identity hypothesis between the two algorithms), whereas the two other risks for sNN versus NN and sNN versus tNN are respectively 0.029% and 0.057%; by means of words, we can *reject* the identity of the performances of sNN and the two algorithms while taking a very small risk $< 1/10\%$.

2.3. *sNN to fix pollution normalization*

It is well known that coastal zones are economical areas sensitive to pollution linked to human activities.[1] The administrators of these coastal structures are confronted with a crucial problem: what are the toxic metal thresholds required to immerse dredging products in the sea?[1] Experts and research organizations have set up physico-chemical criteria, called grids, which are based on heavy metal concentration levels.[1] These grids used in decision making over long-term storage cannot be applied to various environments since they are basically tailor-made for local applications only. As an example, in the Caribbean basin, comparisons with existing standards (French, English, Canadian, Dutch) establish that all tested areas are safe. Heavy metal (Lead and Zinc) concentrations observed in polluted areas in this region are frequently inferior up to several orders of magnitude with respect to the grids healthy reliable limit. One solution at an affordable price to build provisions with reasonable generalization can be to use ML on local pollution measurements. In this line of work, NN-like rules are a first step in addressing the problem, because of their performances and simplicity. Our data contains 121 instances, each described over 11 attributes (As, Cd, Cr, Cu, Fe, Hg, Mn, Ni, Rb, Sn, Zn),

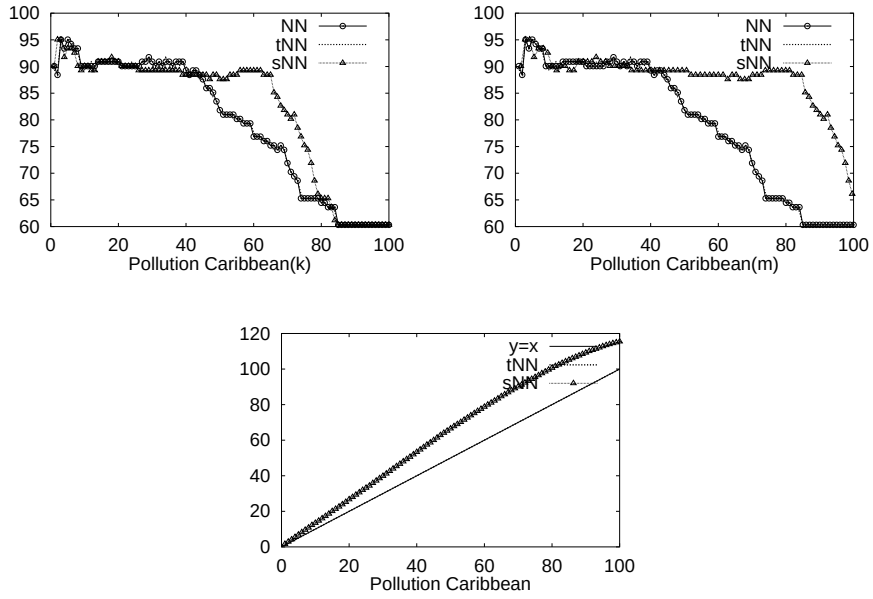


Fig. 3. NN, tNN and sNN performances on Caribbean basin pollution data (notations follow Fig. 2). Right plot displays curves of $\mu(k)$ as a function of k.

plus a class (Polluted/Non Polluted). The data were collected during the period 1992–1999 in the "Cul-de-Sac Marins" areas (about 300 km^2) of Guadeloupe which contains in particular a major harbor of the lesser Antilles: that of Pointe-à-Pitre.[1] The results of the LOOCV's comparisons between the three NN rules are displayed in Fig. 3. Note that the accuracy of C4.5's tree (default settings) is 91.7%, i.e. less than the best values of the NN classifiers. This real-valued dataset is also very interesting to compare our NN rules since the neighborhoods of the NN and tNN rules are the same. The NN and tNN curves are therefore confounded and only the sNN rule can possibly obtain better results. Figure 3 shows that it is indeed the case, as well as show the phenomenon observed in Fig. 2: the resistance against accuracy's degradation for the sNN rule when k or $\mu(k)$ fluctuate.

3. Discussion

3.1. *About the efficiency of the sNN rule*

It was recently argued[8] that the curse of dimensionality theoretically rules out nearest neighbors classifiers as reliable classifiers, a thing which is obviously not true from practice. One possible reason is that instances might be grouped in manifolds whose complexity, and not n, would guide the curse of dimensionality. The extent that its effects are dimmed for this reason is however unclear,[8] but this gives in turn a reason for the success of the 1-sNN rule. Figure 4 plots four examples of $\mu(k)$ for the tNN and sNN algorithms. Remark that on all examples the curves for

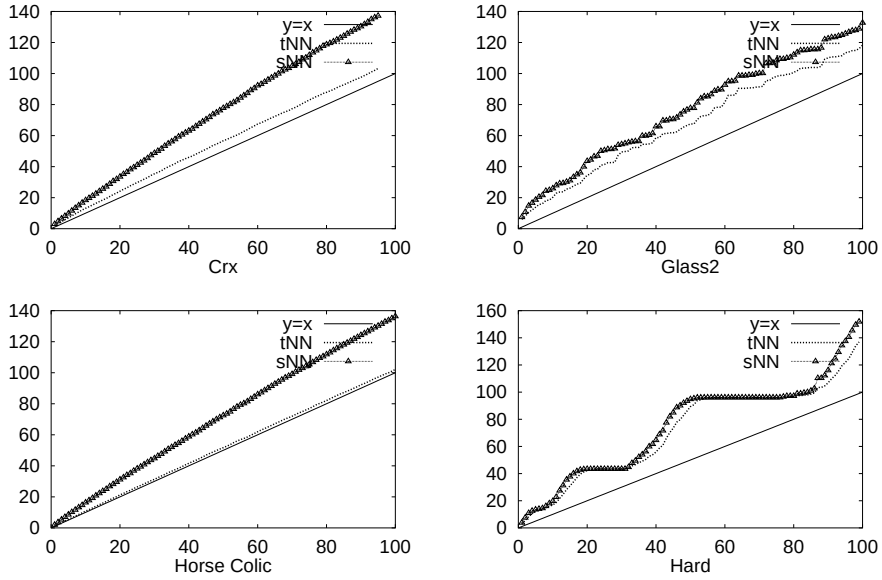


Fig. 4. Four examples of plots of the average neighborhood size of the tNN and sNN algorithms as a function of k (NN's is confounded with $y = x$).

sNN are highly distinct from $y = x$, whereas the curves for the tNN and $y = x$ are sometimes very similar, e.g. for the Horse Colic and Crx datasets.

Provided k is not too large, the maximum radius over all neighborhoods, that are the *same* for each rule, remains reasonable enough for the vote of the corresponding instances to be reliable. Since the neighborhoods of the 1-sNN rule contain the greatest number of instances, and because there are more reliable instances to vote, the overall vote will get even better. On the other hand, the Horse Colic dataset (Fig. 2) shows that when $k \geq 90$, given the greater number ($> 33\%$) of neighbors for the k-sNN rule getting further and more unreliable, the k-sNN rule begins to perform worse than the other rules. Another possible explanation for the good results of the sNN rule could be some smoothness or variation properties over the class conditional densities, stating that on the vicinity of the instances these functions do not fluctuate significantly, or at least they incur the same bias so that predicting Bayes class does not suffer a significant loss (this second hypothesis is also stated in Ref. 8). It is interesting to note that such properties, or something akin to them, were necessary[18] to obtain good theoretical bounds on the error of nearest neighbor rules.[4,5,8,14,17,18] Obviously, it is not to say that our datasets satisfied such assumptions, even if it might be the case for reasonable hypotheses in real-world domains, such as for our pollution measurements data. However, for example, on simulated datasets for which such reasonable smoothness hypotheses *cannot* be satisfied, the sNN rule did not perform as nicely as usual. This is the case for the XD6,[3] Hard and LED Even[2] domains (see Table 2). Hard was partially

generated to theoretically satisfy no such constraint: the target concept is a 5-bits XOR function with 5 additional irrelevant attributes, and attribute noise. In the case of XD6, very weak constraints can be satisfied everywhere in $\mathbf{X}$ (we refer the reader to Ref. 3 for a complete description of the problem).

3.2. *Speeding-up the classification of the sNN rule*

The NN rule has two main complexity-dependent shortcomings. Since it stores all instances in memory, it has a large, $\mathbf{O}(|\mathbf{S}|)$ storing complexity. Since it must search through all stored instances to classify a new instance, its time complexity is also large, $\mathbf{O}(|\mathbf{S}|)$. The sNN algorithm faces the same problems, along with the additional one to find adjuncts in $\mathbf{S}$. While instance selection methods[10,11,19] can significantly reduce the storing and classification complexities of the NN and sNN rules, sophisticated data structures were created for the k-NN rule to reduce the classification complexity of orders, at the expense of a pre-processing stage to organize the reference sample.

The *kd*-trees[7,9] are useful tools for fast searching nearest neighbors (in this notation, the "k" denotes the dimension of the search space, and *not* the number of neighbors). A multidimensional binary search tree is built, by recursively partitioning the reference sample into subsample of equivalent sizes, according to one current variable, until each leaf contains *buckets* of instances of fixed size b. This tree is processed by an oriented depth-first search. The search of the k nearest neighbors of some point x has two principal components. First, a priority queue maintains the k nearest instances found so far; second, pruning rules are used to remove parts of the search space to which the true nearest neighbors cannot belong (we refer the reader to Refs. 7, 9 and 20 for the detailed algorithm).

As argued by Ref. 20, this algorithm is efficient for moderate dimensions. As dimension increases, the repetitive exploration stages in the tree soon visit nearly every bucket, while the tree size consumes more and more space. Two methods of interest for our purposes were proposed to correct this drawback. The first one speeds-up the *kd*-trees processing stage,[16] by giving to some buckets a class label if their centroid obtains a *confident* classification by the k-NN rule. Whenever a labeled leaf is reached by an instance to be classified, the class label is immediately given to the instance without any further processing of the tree to search the nearest neighbors. The second one replaces *kd*-trees by trees with a different internal node structure[20]: each test is a membership test to an hyperball by defining a *vantage point* (reference point) and specifying a corresponding radius. These trees are called *vp*-trees.

The adaptation of the algorithms to the k-sNN rule can be made by completing the previous tree structures to find the adjuncts during the pre-processing stage of $\mathbf{S}$. After having built these trees for the k-NN algorithm, for each $x \in \mathbf{S}$, we compute the radius of its kth nearest neighbor, r_x. We call (x, r_x) the neighborhood ball of x. Note that $\forall\, y \in \mathbf{S}$, if $d(x, y) \leq r_x$, then x is one adjunct of y. After the construction

of the chosen tree for the k-NN rule, we process each of its leaves to make it point to each neighborhood ball intersecting the multidimensional leaf bucket. While it depends on the metric used and on the problem, locally, this number shall remain tractable even while $\mathbf{S}$ increases, because balls intersecting some bucket will get closer and closer to it. When processing this tree to find the sNN neighbors of some $x \in X$, as soon as we reach the first leaf of the tree, which means that the instance belongs to the corresponding bucket, we test the pointed neighborhood balls to see if the corresponding examples are adjuncts of x, and then search for the nearest neighbors of x by processing the tree as explained in Refs. 7, 9 and 20.

3.3. *Optimizing the k-NN rule using the sNN graph*

The principal parameter of the NN rule which can be controlled to optimize the accuracy is k. Its optimal fixation remains one important theoretical issue in the NN field,[15] but two experimental rules have a widespread use. Rule 1 states that we should fix small constant values for k, such as between 1 and 15. Rule 2 states that we should fix negligible values compared to $|\mathbf{S}|$, such as $k = \sqrt{|\mathbf{S}|}$. It appears experimentally on a majority of datasets that for sufficiently small k, the greater the average degree of the sNN graph, the more accurate the k-sNN rule compared to the k-NN rule. Table 1 also shows that the fixation of k to be the average neighborhood size of the 1-sNN rule brings a clear advantage to this $\bar{k}_{sNN}$-NN rule compared to the 1-NN rule (in LOOCV, this represents the average neighborhood size of $\mathbf{S}$ to which the tested instance is removed). We refer to this rule as Rule 3. The comparison Rules 1–3 was carried out on the 29 datasets in Table 1, namely, for each dataset, we have computed the number of values for $k \in \{1, 2, \ldots, 15\}$ for which the accuracy on LOOCV for the corresponding k-NN rule is greater than that of the $\bar{k}_{sNN}$-NN. The average number of values $k \in \{1, 2, \ldots, 15\}$ bringing better accuracies over all problems is 4,67. The median value is 3, which means that the distribution is highly concentrated on the *small* values, or similarly that Rule 3 beats Rule 1 on most cases. To make a simple comparison, replacing the $\bar{k}_{sNN}$-NN with the conventional 1-NN leads to much worse results: average 7.5, median 10, i.e. a distribution concentrated on the *large* values. The comparison Rules 2–3 shows that the $\sqrt{|\mathbf{S}|}$-NN rule performs surprisingly poorly on the 29 datasets, achieving only 76.05% accuracy, a result equivalent to that of the 1-NN rule (see Table 1), but much worse compared to the $\bar{k}_{sNN}$-NN rule, whose accuracy is higher by more than 2.5%.

4. Conclusion

This paper has further explored the possibility to automatically adapt the local neighborhoods in a NN rule type classification. Experiments tend to show the potential of a simple procedure, which increases the number of voters without facing the curse of dimensionality as in traditional nearest neighbor learning. The major issue regarding this approach is to quantify the risk of this symmetric nearest

neighbor learning rule, this time from a theoretical point of view, and compare it with corresponding results for the traditional NN rule for finite reference samples.[14]

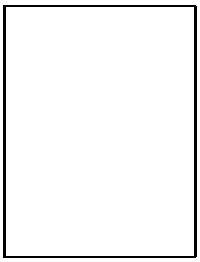
Acknowledgments

The authors gratefully acknowledge matching funds provided by the French Ministry for Ultraperipheric Regions (SEDETOM).

References

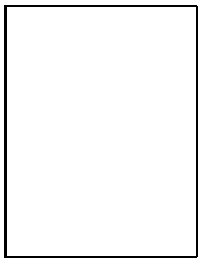
1. D. Bernard, "Metals in sediments from two lagoons off guadeloupe, West Indies," *Marine Pollution Bull.* **30** (1995) 619–621.
2. C. L. Blake, E. Keogh and C.J. Merz, "UCI repository of machine learning databases," 1998, `http://www.ics.uci.edu/~mlearn/MLRepository.html`.
3. W. Buntine and T. Niblett, "A further comparison of splitting rules for decision-tree induction," *Mach. Learn.* **8** (1992) 75–85.
4. T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inform. Th.* **13** (1967) 21–27.
5. J. A. Drakopoulos, "Bounds on the classification error of the nearest neighbor rule," *Proc. 12th Int. Conf. Machine Learning*, 1995, pp. 203–208.
6. E. Fix and J. L. Hodges, "Discrimatory analysis, nonparametric discrimination," Technical Report TR-21-49-004, Report 4, USAF School of Aviation Medicine, Randolph Field, TX, 1951.
7. J. H. Freidman, J. L. Bentley and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Software* **3** (1997) 209–226.
8. J. H. Friedman, "Flexible metric nearest neighbor classification," Technical Report, Department of Statistics, Stanford University, 1994.
9. J. H. Friedman, J. L. Baskett and L. J. Shustek, "An algorithm for finding nearest neighbors," *IEEE Trans. Comput.* **24** (1975) 1000–1006.
10. G. W. Gates, "The reduced nearest neighbor rule," *IEEE Trans. Inform. Th.* **18** (1972) 431–433.
11. P. E. Hart, "The condensed nearest neighbor rule," *IEEE Trans. Inform. Th.* **14** (1968) 515–516.
12. D. Harwood, S. Subbarao, H. Hakalahti and L. S. Davis, "A new class of edge-preserving smoothing filters," *Patt. Recogn. Lett.* **6** (1987) 155–162.
13. T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor classification," *IEEE Trans. Patt. Anal. Mach. Intell.* **18** (1996) 607–615.
14. R. Nock and M. Sebban, "An improved bound on the finite sample risk of the nearest neighbor rule," *Patt. Recogn. Lett.* **22** (2001) 413–419.
15. S. Okamoto and N. Yugami, "Theoretical analysis of the nearest neighbor classifier in noisy domains," *Proc. 13th Int. Conf. Machine Learning*, 1996, pp. 355–363.
16. A. M. Palau and R. Snapp, "The labeled cell classifier: a fast approximation to k nearest neighbors," *Proc. 14th Int. Conf. Pattern Recognition*, Vol. 1, 1998.
17. R. R. Snapp and S. S. Venkatesh, "Asymptotic derivation of the finite-sample risk of the k nearest neighbor classifier," Technical Report UVM-CS-1998-0101, University of Vermont, Burlington, 1998.
18. S. S. Venkatesh, R. R. Snapp and D. Psaltis, "BELLMAN STRIKES AGAIN! the growth rate of sample complexity with dimension for the nearest neighbor classifier," *Proc. 5th Int. Conf. Computational Learning Theory*, 1992, pp. 93–102.

19. D. Wilson and T. Martinez, "Instance pruning techniques," *Proc. 14th Int. Conf. Machine Learning*, 1997, pp. 404–411.
20. P. N. Yanilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," *Proc. 5th ACM-SIAM Symp. Discrete Algorithms*, 1993, pp. 311–321.



Richard Nock received his Ph.D. in computer science from the University of Montpellier II, France, in 1998. He also holds an agronomical engineering degree from the Ecole Nationale Super ieure Agronomique de Montpellier (ENSA.M), France. He is currently an Associate Professor of computer science at the "Université Antilles-Guyane", located in the Martinique campus of the University.

His current research interests include: machine learning, data mining, computational complexity and image processing.



Marc Sebban received his Ph.D. in computer science from the University of Lyon 2, France, in 1996. Since then, he has been an Assistant Professor at the "Université Antilles-Guyane", Associate Professor and then Full Professor of computer science at the Universit Jean Monnet, Saint-Etienne, France.

His current research interests include: machine learning, grammatical inference, knowledge discovery and computational geometry.