Oddness-based classification: A new way of exploiting neighbors

Myriam Bouhnas, Henri Prade, Gilles Richard

Oddness-based classification: A new way of exploiting neighbors

Myriam Bounhas[1,2] **Henri Prade**[3,4] **Gilles Richard**[3,5]

[1] LARODEC Laboratory, ISG de Tunis, Tunis, Tunisia

[2] Emirates College of Technology, Abu Dhabi, UAE

[3] IRIT, Université Paul Sabatier, Toulouse, France

[4] QCIS, University of Technology, Sydney, Australia

[5] British Institute of Technology and E-commerce, London, UK

Correspondence
Myriam Bounhas, Emirates College of Technology, P.O. Box 41009, Abu Dhabi, United Arab Emirates.
Email: myriam_bounhas@yahoo.fr

Abstract

The classification of a new item may be viewed as a matter of associating it with the class where it is the least at odds w.r.t. the elements already in the class. An oddness measure of an item with respect to a multiset, applicable to Boolean features as well as to numerical ones, has been recently proposed. It has been shown that cumulating this measure over pairs or triples (rather than larger subsets) of elements in a class could provide an accurate estimate of the global oddness of an item with respect to a class. This idea is confirmed and refined in the present paper. Rather than considering all the pairs in a class, one can only deal with the pairs whose an element is one of the nearest neighbors of the item, in the target class. The oddness evaluation computed on this basis still leads to good results in terms of accuracy. One can take a step further and choose the second element in the pair also as another nearest neighbor in the class. Although the method relies on the notion of neighbors, the resulting algorithm is far from being a variant of the classical k-nearest neighbors approach. The oddness with respect to a class computed only on the basis of pairs made of two nearest neighbors leads to a low complexity algorithm. Experiments on a set of UCI benchmarks show that the classifier obtained can compete with other well-known approaches.

1 INTRODUCTION

A decade ago, a new method of classification based on analogical proportions (ie, statements of the form "a is to b as c is to d"), has been proposed, showing its effectiveness on Boolean classical UCI benchmarks.[1] The method had a cubic complexity due to the search for suitable triples of examples for building analogical proportions with the item to be classified. A computational improvement of the previous analogical classifier has been recently made.[2,3] It has been shown that choosing one element of the triple as a nearest neighbor of the item to be classified reduces the complexity without harming the accuracy.

Besides, analogical proportions have been recognized as being members of a larger family, the so-called "logical proportions."[4,5] Logical proportions are propositional logic expressions that link four Boolean variables through equivalences between similarity or dissimilarity indicators pertaining to pairs of these variables. Apart from the analogical proportion, another remarkable type of logical proportions, named heterogeneous proportions, expresses that there is an element at odds (an intruder) among four Boolean values, which is not in some prescribed position (eg, "there is an intruder value among a, b, c, d which is not a). If there is an intruder in the multiset $\{a, b, c\} \cup \{d\}$, which is neither a, nor b, nor c, this means that d is at odds with the triple $\{a, b, c\}$. The truth value of such a heterogeneous proportion-based expression can then be considered as an oddness index telling if d is at odds or not w.r.t. $\{a, b, c\}$.

Several classification methods rely on the intuitive idea that a new item should be classified in the class with respect to which it appears to be the least at odds. Heterogeneous proportions provide a natural basis to build a global oddness measure of an item x with respect to a set, by cumulating the oddness index with regard to different triples of elements in the set for the different features. This has suggested a simple procedure for classifying an item x, which puts it into the class $\mathcal{C}$ that minimizes this global oddness measure. This has been experimented with promising results for Boolean features.[6] But this method has the drawback of dealing with a huge number of triples. Moreover, it has also been noticed that the oddness measure of an item with respect to a triple can be generalized to a subset of any size, as well as to numerical features.[7,8] In this paper, we reconsider the above ideas, but we focus on the interest of using pairs rather than triples when building of the oddness measure. This has an obvious complexity advantage, but may also lead to slightly better results in general, as we shall see.

First, in this paper, we further investigate the comparison between the use of pairs and the use of triples in the oddness measure. Then, we only use pairs and triples in a given class whose one element is a nearest neighbor of the item, *in the class*. Finally, we more particularly investigate the use of oddness measures based on *selected* pairs, since our aim here is to show that we can keep on with the same idea of oddness. Doing so, we preserve the accuracy while drastically reducing the complexity by constraining the set of considered pairs. The idea is to constrain the choice of the *two* elements in the pairs. We first consider the option of building pairs with one element as a nearest neighbor and the other member of the pair as a most remote element of the item among the examples (thus providing a view of the diversity in the class). Then, we investigate the use of pairs made of two nearest neighbors only. However, the way we use these pairs of neighbors is very different from the k-NN method as it will be explained below.

The paper is a substantially expanded version of a short conference paper,[9] with more methodological developments and a new experimental part. The paper is structured as follows. Section 2 restates the logical definition of an oddness index, for Boolean, and then for numerical features (rescaled in the unit interval, as a graded truth value), first with respect to one feature and then to a set of features. In Section 3, we explain why it is of interest to use a pair- or a triple-based evaluation of global oddness. We then build a global oddness measure of an item with respect to a class by cumulating oddness indices w.r.t. subsets of the given class. In Section 4, our experimentations proceed in two steps: first, we implement and test oddness-based classifiers with singletons, pairs, and triples. Second, we select one element of each pair, or triple, as a nearest neighbor in the candidate class, thus reducing the complexity. It appears that the accuracy becomes even slightly better, especially in the case of pairs. In Section 5, we investigate the possibility of considering more constrained pairs where the two elements of the pairs are selected as nearest neighbors. It preserves the good accuracy results while reducing again the complexity. The concluding remarks in Section 7 stress the novelty of the approach.

2 ODDNESS INDEX OF AN ITEM W.R.T. A MULTISET OF VALUES

We first provide a definition of the oddness of an item with respect to a multiset of any size, for one feature, and we relate this elementary oddness index to heterogeneous logical proportions in the case of triples. Then, we extend the oddness index to numerical features, and to a set of features.

2.1 The Boolean case

What does it mean to be an outsider or an intruder with respect to a multiset of n Boolean values, logically speaking? This can be formally stated as follows: Given a multiset of Boolean values $\{a_i \mid i \in [1, n]\}$, we look for a logical formula $F(a_1, \ldots, a_n, x)$ such that:

$$F(a_1, \ldots, a_n, x) = 1 \text{ if } a_1 = \cdots = a_n = 0, x = 1$$

$$\text{or if } a_1 = \cdots = a_n = 1, x = 0$$

and

$$F(a_1, \ldots, a_n, x) = 0 \text{ otherwise}$$

This formula should obviously be independent of the order of the a_i's and should satisfy the following properties:

1. In the case $x = 1$, $F(a_1, \ldots, a_n, x) = 1$ implies that all the a_i's should be 0, which is equivalent to $\vee_{i\in[1,n]} a_i = 0$. So the formula we look for implies:

$$\neg\left(\bigvee_{i\in[1,n]} a_i \equiv x\right)$$

since x and the disjunction should have different truth values.

TABLE 1 $odd(\{a_1, a_2\}, x)$ truth values

a_1	a_2	x	odd
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

2. In the case $x = 0$, $F(a_1, \ldots, a_n, x) = 1$ implies all the a_i's should be 1, which is equivalent to $\wedge_{i\in[1,n]} a_i = 1$. So the formula we look for also implies:

$$\neg\left(\bigwedge_{i\in[1,n]} a_i\right) \equiv x$$

It means that we can summarize the oddness of a given Boolean value x with respect to a given multiset of Boolean values $\{a_i \mid i \in [1, n]\}$ as follows:

$$odd(\{a_i | i \in [1, n]\}, x) =_{def} \neg (\vee_{i\in[1,n]} a_i \equiv x) \wedge \neg (\wedge_{i\in[1,n]} a_i \equiv x) \quad (1)$$

Due to the commutativity of logical operators $\vee$ and $\wedge$, the ordering of the a_i's is meaningless and we can simply keep a multiset notation. In the particular case of three Boolean variables ($n = 2$), this formula leads to the truth table in Table 1, in the case of four Boolean variables ($n = 3$), to the truth table in Table 2.

It is clear that *odd* holds true only when the value of x is seen as being *at odds* among the other values: roughly speaking, x is the intruder in the multiset of values. In the case $n = 2$, $odd(\{a_1, a_2\}, x)$ is 0 if and if the value of x is among the majority value in the set $\{a_1, a_2, x\}$. When $n = 3$, $odd(\{a_1, a_2, a_3\}, x)$ does not hold true in the situation where there is a majority among values in a_1, a_2, a_3, x and x belongs to this majority (eg, $odd(\{0, 1, 0\}, 0) = 0$), or when there is no majority at all (eg, $odd(\{0, 1, 1\}, 0) = 0$).

2.2 Link with heterogeneous proportions

When $odd(\{a, b, c\}, x)$ is true, this means that among the four variables a, b, c, x in this order, there is an intruder which is not a, b, or c. A direct link between the Boolean oddness index and the heterogeneous proportions previously mentioned in the introduction, in relation with the

TABLE 2 $odd(\{a_1, a_2, a_3\}, x)$ truth values

a_1	a_2	a_3	x	odd
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

TABLE 3 H_1, H_2, H_3, H_4 Boolean truth tables

H_1				H_2				H_3				H_4			
1	1	1	0	1	1	1	0	1	1	1	0	1	1	0	1
0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0
1	1	0	1	1	1	0	1	1	0	1	1	1	0	1	1
0	0	1	0	0	0	1	0	0	1	0	0	0	1	0	0
1	0	1	1	0	1	1	1	0	1	1	1	0	1	1	1
0	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0

analogical proportion, can be established. There are four heterogeneous proportions denoted H_1, H_2, H_3, H_4[5] whose truth tables are given in Table 3 (only the lines leading to truth value 1 are indicated).

These proportions express the fact that there is an intruder among $\{a, b, c, d\}$, which is not $a(H_1)$, which is not $b(H_2)$, which is not $c(H_3)$, and which is not $d(H_4)$, respectively. It is then clear that we have:

$$odd(\{a, b, c\}, x) \equiv H_1(a, b, c, x) \wedge H_2(a, b, c, x) \wedge H_3(a, b, c, x).$$

This conjunction tells us that among a, b, c, x in this order, there is an intruder, which is not $a, b,$ or c: this means x is the intruder. Indeed, we can check that the conjunction $H_1(a, b, c, x) \wedge H_2(a, b, c, x) \wedge H_3(a, b, c, x)$ has Table 2 as truth table.

2.3 Extension to numerical data

Assuming that numerical features are renormalized between 0 and 1, an option for an extension of the oddness index to such feature values is to use the standard counterpart of the logical connectives in the [0, 1]-valued Łukasiewicz logic,[10] where:

1. min extends the conjunction $\wedge$, max the disjunction $\vee$,
2. $1 - (\cdot)$ extends the negation $\neg$,
3. $1 - |\cdot - \cdot|$ extends the equivalence $\equiv$.

Then, a graded counterpart to formula (1) is,

$$odd(\{a_i \mid i \in [1, n]\}, x) =_{def} \min(|x - \max_{i\in[1,n]} a_i|, |x - \min_{i\in[1,n]} a_i|) \tag{2}$$

When restricted to Boolean values, this formula reduces to formula (1). The use of Łukasiewicz connectives for extending oddness to numerical values, even if many other choices would be possible, is a natural option here since it leads to take into account extremal values (via min and max) and the absolute values of differences, easy to understand as distances. It can be checked in the graded case that:

1. $odd(\{u, \cdots, u\}, v) = |u - v|$. Indeed, if $u = v$, the last value is not an intruder. The larger $|u - v|$, the more v is at odds w.r.t. the n values equal to u.
2. $odd(\{u, \cdots, u, v, \cdots, v\}, v) = 0$ which is consistent with the expected semantics of *odd* since v belongs to the multiset $\{u, \cdots, u, v, \cdots, v\}$.
3. if $\min_{i\in[1,n]} a_i \leq x \leq \max_{i\in[1,n]} a_i$, formula (2) implies that $odd(\{a_i | i \in [1, n]\}, x) \leq 0.5$. This again agrees with our need since, in that case, *odd* should remain rather small as long as x is in between elements of the multiset. The oddness can be high (> 0.5) only if x is outside the convex hull of the multiset.

From these properties, we understand that the proposed definition fits with the intuition and provides high truth values when x appears at *odds* w.r.t. the set $\{a_1, \cdots, a_n\}$ and low truth values in the opposite case where x is not too different from the other values. Let us consider the particular cases when $n = 1$, $n = 2$, and $n \geq 3$.

1. When $n = 1$, in the Boolean case, the formula $odd(\{a_1\}, x)$ is just equivalent to $\neg(a_1 \equiv x)$. In that particular case, this is equivalent to $\neg a_1 \equiv x$. In the multiple-valued logic extension, the truth value of $odd(\{a_1\}, x) = |x - a_1|$. This is just the distance (in $\mathbb{R}$) between the two values.
2. When $n = 2$, formula (2) reduces to: $odd(\{a, b\}, x) = \min(|x - \max(a, b)|, |x - \min(a, b)|)$, ie, $odd(\{a, b\}, x) = \min(|x - a|, |x - b|)$, ie, the min of the distances between x and each of a and b. This is also the classical distance between x and the set $\{a, b\}$.
3. When $n \geq 3$, it is worth noticing that $odd(\{a_i \mid i \in [1, n]\}, x)$ is no longer always equal to the distance of x to the subset $\{a_i | i \in [1, n]\}$, ie, $d(\{a_i | i \in [1, n]\}, x) = \min_{i\in[1,n]} |x - a_i|$. In fact, $d(\{a_i | i \in [1, n]\}, x) \leq odd(\{a_i | i \in [1, n]\}, x)$. Indeed, as can be seen in formula (2), *odd* is the minimum of two distances to elements in the multiset $\{a_i | i \in [1, n]\}$, while

$d(\{a_i | i \in [1, n]\}, x)$ is the minimum of the distances to the n elements of the multiset. For instance, in the case $n = 3$, $odd(\{0.2, 0.5, 0.8\}, 0.6) = 0.2$ and $d(\{0.2, 0.5, 0.8\}, 0.6) = 0.1$.

This is a distinctive feature of singletons and pairs to be such that $odd(\{a_i | i \in [1, n]\}, x) = d(\{a_i | i \in [1, n]\}, x)$ for $n = 1$ and $n = 2$. In fact, for $n \geq 3$, we have that $odd(\{a_i | i \in [1, n]\}, a_i)$ is not necessarily equal to 0. Values $n \geq 3$ have to be investigated in the future.

2.4 | Extension to vectors

Generally, items are represented by a set of m features, rather than one feature. That is why we have to define an oddness measure for vectors of dimension m. When dealing with vectors $\vec{x} \in [0, 1]^m$, Boolean vectors are also covered as a particular case. The *odd* measure, defined by (1) and extended in (2) is used to estimate to what extent a value x can be considered as being at odds among a multiset S of values for one feature.

A natural option for an extension to m features is to consider a definition of *odd* for vectors as the sum componentwise of the *odd* values computed via formula (1) or (2). This leads to the general formula:

$$odd(\{\vec{a_i} | i \in [1, n]\}, \vec{x}) =_{def} \sum_{j=1}^{m} odd(\{a_i^j | i \in [1, n]\}, x^j) \tag{3}$$

where x^j is the jth component of $\vec{x}$ and the a_i^j's are the jth components of the vectors $\vec{a_i}$ respectively. Obviously,

$$odd(\{\vec{a_i} | i \in [1, n]\}, \vec{x}) \in [0, m]$$

as being the sum of m numbers belonging to $[0, 1]$.

In particular, $odd(\{\vec{a}\}, \vec{x})$ reduces to the L^1 norm in $\mathbb{R}^m$ of the vector $\vec{a} - \vec{x}$: $\Sigma_{j=1}^{m} |a^j - x^j| = ||\vec{a} - \vec{x}||_{L^1}$. In the case of $\mathbb{B}^n$, this is just the Hamming distance between $\vec{a}$ and $\vec{x}$.

The Boolean case and the numerical case are just particular cases of formula (3) where $m = 1$. Note that we implicitly assume that the features are independent in this context.

If $odd(\{\vec{a_i} | i \in [1, n]\}, \vec{x}) = 0$, it means that no feature indicates that $\vec{x}$ behaves as an intruder among the a_i's. On the contrary, high values of $odd(\{\vec{a_i} | i \in [1, n]\}, \vec{x})$ (close to m) means that, on *many* features, $\vec{x}$ appears as an intruder.

3 ODDNESS-BASED CLASSIFIERS

We now explain how the oddness index *odd* can be used for building a classifier.

3.1 Global oddness measure

Given a set $\mathscr{C} = \{\vec{a_i} | i \in [1, n]\}$ of vectors gathering the set of examples of the same class, one might think of computing $odd(\mathscr{C}, \vec{x})$ as a way of evaluating how much $\vec{x}$ is at odds with respect to $\mathscr{C}$. An immediate classification algorithm would be to compute $odd(\mathscr{C}, \vec{x})$ for every class and to allocate to $\vec{x}$ the class which minimizes this number. Nevertheless, as explained now, this number is not really meaningful when the size of $\mathscr{C}$ is large, whatever the number of features.

Indeed, we have to be careful because then $\{\vec{a_i} | i \in [1, n]\}$ is summarized by two vectors made respectively by the minimum and the maximum of the feature values among the examples of $\mathscr{C}$ (due to formulas 2 and 3). These two vectors have high chance to be *fictitious* in the sense that, usually, they are not elements of $\mathscr{C} = \{\vec{a_i} | i \in [1, n]\}$. Approximating our knowledge of the set $\{\vec{a_i} | i \in [1, n]\}$ using only the maximal ranges of the feature values over the members of the set seems very crude.

The above remark tends to indicate that $odd(\{\vec{a_i} | i \in [1, n]\}, \vec{x})$ may not be a good marker of the oddness of x w.r.t. $\{\vec{a_i} | i \in [1, n]\}$ when n is large. We have to devise a method allowing to overcome this issue.

An idea is then to consider small subsets S of the class $\mathscr{C}$, then compute $odd(S, \vec{x})$ and finally add all these atomic oddness indices to get a global measure of oddness of $\vec{x}$ w.r.t. $\mathscr{C}$. This approach leads to the following initial formula: $\Sigma_{S \subseteq \mathscr{C}, |S|=n} odd(S, \vec{x})$. But, to take into account the relative size of the different classes, it is fair to introduce a normalization factor and our final definition is:

$$Odd_n(\mathscr{C}, \vec{x}) = \frac{1}{\binom{|\mathscr{C}|}{n}} \sum_{S \subseteq \mathscr{C}, |S|=n} odd(S, \vec{x})$$

As we consider only subsets S of small size (ie, singletons, pairs, or triples), the previous formula becomes:

1. For singletons:

$$Odd_1(\mathscr{C}, \vec{x}) = \frac{1}{|\mathscr{C}|} \sum_{\vec{a} \in \mathscr{C}} odd(\{\vec{a}\}, \vec{x})$$

Since $odd(\{\vec{a}\}, \vec{x}) = ||\vec{a} - \vec{x}||_{L_1}$, Odd is just the average L_1 distance between $\vec{x}$ and all the elements of the class $\mathscr{C}$.

2. For pairs:

$$Odd_2(\mathscr{C}, \vec{x}) = \frac{1}{\binom{|\mathscr{C}|}{2}} \sum_{\vec{a}, \vec{b} \in \mathscr{C}} odd(\{\vec{a}, \vec{b}\}, \vec{x})$$

As already said, normalizing the summation aims to take into consideration the relative size of different classes. As the size of the subsets S (equal to 2 here) do not have any impact on classification in practice and as, for big classes, $\binom{|\mathscr{C}|}{2}$ is more or less equal to $|\mathscr{C}|^2$, in practice, we simply use a normalization factor equal to $\frac{1}{\mathscr{C}^2}$.

3. For triples:

$$Odd_3(\mathscr{C}, \vec{x}) = \frac{1}{\binom{|\mathscr{C}|}{3}} \sum_{\vec{a}, \vec{b}, \vec{c} \in \mathscr{C}} odd(\{\vec{a}, \vec{b}, \vec{c}\}, \vec{x})$$

In practice, we use $\frac{1}{|\mathscr{C}|^3}$ as normalization factor.

In the above evaluation, we consider *all* the subsets of $\mathscr{C}$ of a given size. We first explore this option. We might also imagine to deal only with *particular* subsets of a given size as soon as we do not harm the performance. In the following, we refer to the choice of particular subsets, by the general term of *optimization* when speaking of the proposed classifiers.

3.2 | Classification

A classification algorithm is easily deducible from this oddness measure. Let TS be a training set composed of instances $(\vec{z}, cl(\vec{z}))$, where $\vec{z} \in \mathbb{B}^m$ or $\mathbb{R}^m$ and $cl(\vec{z})$ is the label of $\vec{z}$. Given a new instance $\vec{x} \notin TS$ without label, and choosing a suitable size n, we allocate to this item the label corresponding to the class minimizing $Odd_n(\mathscr{C}, \vec{x})$.

Algorithm 1 Oddness-based algorithm

```
Input: a training set $TS$ of examples $(\vec{z}, cl(\vec{z}))$
       a new item $\vec{x}$,
       an integer $n$,
Partition $TS$ into sets $\mathscr{C}$ of examples with the same label $c$.
for each $\mathscr{C}$ do
    Compute $Odd_n(\mathscr{C}, \vec{x})$ for subsets of size $n$
end for
$cl(\vec{x}) = argmin_c(Odd_n(\mathscr{C}, \vec{x}))$
return $cl(\vec{x})$
```

In the following, we first report results obtained with this algorithm on different benchmarks. Later, we shall continue to apply the same algorithm but by slightly modifying the global oddness measure Odd_n that we use in Algorithm 1.

4 EXPERIMENTATIONS AND PRELIMINARY DISCUSSION

In this section, we provide experimental results obtained with *Oddness*-based classifiers and we compare them to other machine learning classifiers. After introducing the datasets for the benchmarks, we report the results obtained with the oddness-based classifiers, first considering *a*ll the subsets of a given size to compute the global oddness measures. Then, we progressively filter the considered subsets by only using appropriate ones.

4.1 Datasets

The experimental study is based on 19 datasets taken from the UCI machine learning repository.[11] A brief description of these data sets is given in Table 4.

Since an oddness-based classifier is able to deal with both Boolean or multivalued features in this study, the first part in this Table 4 includes eight datasets with categorical or Boolean attribute values and the second part includes 10 datasets with only numerical attributes.

In terms of classes, we deal with a maximum number of 26 classes. To apply the Boolean and multiple-valued semantics framework, all categorical attributes are binarized and all numerical

TABLE 4 Description of datasets

Datasets	Instances	Nominal Att.	Binary Att.	Numerical Att.	Classes
Balance	625	4	20	–	3
Car	743	7	21	–	4
Monk1	432	6	15	–	2
Monk2	432	6	15	–	2
Monk3	432	6	15	–	2
Spect	267	–	22	–	2
Voting	435	–	16	–	2
Hayes-Roth	132	5	15	–	3
Diabetes	768	–	–	8	2
W. B. Cancer	699	–	–	9	2
Heart	270	–	–	13	2
Magic	1074	–	–	11	2
Ionosphere	351	–	–	35	2
Iris	150	–	–	4	3
Wine	178	–	–	13	3
Satellite Image	1090	–	–	36	6
Segment	1500	–	–	19	7
Glass	214	–	–	9	7
Letter	1076	–	–	16	26

attributes are rescaled. More precisely, we just replace a real value r with $\frac{r-r_{\min}}{r_{\max}-r_{\min}}$, where $r_{\min}$ and $r_{\max}$ respectively represent the minimal and the maximal values for this attribute on this data set. A real value is thus changed into a number that may be understood as a truth value. The minimal and the maximal values for each attribute are computed is a preprocessing step using the training set only. These values are then used to scale both training and testing data. If a value in the rescaled testing set is outside the range $[0, 1]$, we remove the corresponding item. This appears in only some of the tested datasets with maximum one or two items in the testing set to be removed.

For nominal attributes, we experiment over the following eight datasets:

1. Balance, Car, and Hayes-Roth are multiple classes datasets.
2. Monk1, Monk2, Monk3, Spect, and Voting datasets are binary class problems. Monk3 has noise added (in the sample set only). Spect and Voting data sets contain only binary attributes. Voting has missing attribute values.

For numerical datasets, we experiment over the following 11 datasets:

1. Iris, Wine, Sat.Image, Glass, Segment, and Letter datasets are multiple class problems.
2. Diabetes, W.B. Cancer, Heart, Magic, and Ionosphere are binary class datasets.

4.2 Testing protocol

In terms of protocol, we apply a standard 10-fold cross-validation technique. As usual, the final accuracy is obtained by averaging the 10 different accuracies for each fold. However, classical classifiers (with which we compare our approach) as well as some variants of *Oddness*-classifiers (that will be introduced in Section 4.4) have parameters to be tuned before performing this cross-validation. To do that, we randomly choose a fold (as recommended by[12]), we keep only the corresponding training set (ie, which represents 90% of the full data set). On this training set, we again perform a 10-fold cross-validation with diverse values of the parameters. We then select the parameter values providing the best accuracy. These tuned parameters are then used to perform the initial cross-validation. As expected, these tuned parameters change with the target data set. To be sure that our results are stable enough, we run each algorithm (with the previous procedure) five times so we have five different parameter optimizations. Each displayed parameter in Tables 5 and 6 (that we denote β) is the average value over the five different values (one for each run). The results shown in Tables 5 and 6 are the average values obtained from five rounds of this complete process. In case there is no parameter to be tuned (this is the case of Odd_1, Odd_2, and Odd_3 in next section), we simply apply a standard 10-fold cross-validation still repeated five times to get stable results.

4.3 Results for the classifiers Odd_1, Odd_2, and Odd_3

In this section, we test the efficiency of the basic oddness classifiers. Table 7 provides mean accuracies and standard deviations obtained with our implementations of Odd_1, Odd_2, and Odd_3.

We observe that:

1. Odd_1 is globally less accurate than other classifiers for all datasets, having an average accuracy of 73% when Odd_2 has 78% and Odd_3 has 79%.
2. For 10 datasets, the accuracy increases when we use subsets of triples instead of pairs (this is also true when we use subsets of pairs instead of singletons; see for example “Car” and “Spect.” datasets).
3. However, these results remain not satisfactory if compared to the well-known algorithms such as SVM or IBK, not only in terms of accuracy, but also in terms of complexity. In the next section, we apply different optimizations, focusing on Odd_2 classifier.

Note that the results reported here for Odd_1 are quite similar to the ones obtained in Bounhas et al.[8] But since we follow a slightly more strict experimental protocol in this paper, we have repeated the experiments with the new protocol for having clear points of comparison for the new experiments reported in the following. While results for Odd_2 and Odd_3 are reported for the first time in this paper.

4.4 Use of an improved oddness measure

The rather poor performances of our oddness classifiers may be due to the huge number of subsets considered in each class, having equal importance, while a lot of them blur the accuracy of oddness measure through the summation. We may think of privileging subsets including elements of particular interest such as nearest neighbors in the class.

On top of that, such a strategy will reduce the computational complexity of the algorithms. An obvious option is to consider only subsets which contain one of the k nearest neighbors of x in the class. In that case, we have to adjust the normalization factor which becomes $\frac{1}{k\times|\mathscr{C}^{n-1}|}$ in the Odd_n formula.

TABLE 5 Classification accuracies given as mean and standard deviation with improved $Odd_1(NN)$, $Odd_2(NN, Std)$, $Odd_2(NMRE)$, $Odd_2(NN, NN)$, and $Odd_3(NN, Std, Std)$ obtained with the best parameter β

	$Odd_1(NN)$		$Odd_2(NN, Std)$		$Odd_2(NMRE)$		$Odd_2(NN, NN)$		$Odd_3(NN, Std, Std)$	
Datasets	**Acc.**	β	**Acc.**	β	**Acc.**	β	**Acc.**	β	**Acc.**	β
Balance	82.79 ± 4.29	9	87.4 ± 4.08	21	82.62 ± 3.06	(16, 13)	83.9 ± 5.02(−)	17	88.62 ± 3.4	2
Car	91.95 ± 3.31	8	92.04 ± 4.04	7	91.19 ± 2.68	(6, 20)	92.45 ± 3.88(.)	7	90.93 ± 4.03	5
Monk1	99.91 ± 0.09	6	99.77 ± 0.23	4	99.43 ± 0.79	(7, 7)	99.81 ± 0.67(.)	6	99.31 ± 3.39	1
Monk2	66.54 ± 2.1	20	64.45 ± 3.1	19	65.42 ± 3.27	(19, 2)	67.46 ± 3.25(+)	16	60.93 ± 4.16	17
Monk3	99.95 ± 0.05	3	99.95 ± 0.72	1	99.37 ± 1.4	(5, 12)	100(.)	5	99.95 ± 0.05	2
Spect	82.74 ± 6.49	13	83.95 ± 4.72	9	83.14 ± 8.15	(14, 16)	84.55 ± 4.42(+)	16	84.1 ± 4.58	5
Voting	93.04 ± 3.2	9	94.23 ± 3.95	10	93.45 ± 4.99	(5, 15)	95.33 ± 2.84(+)	17	93.81 ± 2.86	11
Hayes-Roth	63.17 ± 12.41	14	78.35 ± 11.94	3	74.59 ± 8.93	(4, 19)	77.73 ± 9.62(−)	7	79.37 ± 9.74	7
Diabetes	75.06 ± 3.25	17	76.28 ± 3.83	17	75.5 ± 4.53	(16, 11)	75.28 ± 4.26(−)	18	75.91 ± 4.58	15
Cancer	97.07 ± 2.19	3	97.27 ± 1.34	5	97.15 ± 2.61	(4, 9)	97.24 ± 1.63(.)	12	97.04 ± 2.24	5
Heart	82.33 ± 5.15	13	82.52 ± 7.87	13	83.41 ± 7.04	(10, 5)	82.54 ± 5.06(.)	19	82.2 ± 4.01	13
Magic	78.78 ± 1.58	13	79.05 ± 3.13	16	78.06 ± 3.48	(14, 18)	79.23 ± 2.41(.)	18	74.53 ± 3.02	17
Ionosphere	90.61 ± 3.84	1	92.09 ± 3.32	8	91.18 ± 4.51	(1, 18)	91.78 ± 4.95(−)	15	90.55 ± 4.05	14
Iris	94.56 ± 4.11	9	94.97 ± 4.41	9	94.94 ± 4.33	(11, 9)	94.57 ± 4.11(−)	14	94.64 ± 5.32	7
Wine	97.55 ± 3.07	11	98.47 ± 2.52	5	97.2 ± 3.19	(3, 15)	98.37 ± 2.77(.)	9	97.16 ± 4.76	5
Sat. Image	94.79 ± 2.78	3	95.03 ± 3.08	1	94.49 ± 2.08	(2, 12)	95.38 ± 2.59(+)	5	93.43 ± 2.38	1
Segment	96.76 ± 1.3	2	96.67 ± 1.44	1	96.67 ± 1.25	(2, 9)	96.79 ± 1.17(+)	4	95.31 ± 2.14	3
Glass	72.87 ± 8.1	3	75.84 ± 9.8	3	74.94 ± 8.58	(3, 5)	77.93 ± 7.27(+)	3	72.22 ± 8.19	3
Letter	75.66 ± 3.34	2	75.86 ± 4.57	2	75.1 ± 2.77	(3, 17)	78.04 ± 3.76(+)	5	73.8 ± 2.81	2
Average	86.11		87.59		86.73		87.81		86.52	

TABLE 6 Results for other machine learning classifiers obtained with the best parameter β

Datasets	IBk		C4.5		JRIP		SVM (RBFKernel)		SVM (PolyKernel)	
	Accuracy	β	*Accuracy*	β	*Accuracy*	β	*Accuracy*	β	*Accuracy*	β
Balance	90.15 ± 2.6	17	77.5 ± 5.12	0.3	76.84 ± 2.28	7	99.17 ± 0.34	(32 768, 0.00195)	98.53 ± 0.26	(128, 1)
Car	91.84 ± 3.51	2	95.53 ± 2.02	0.2	91.54 ± 3.21	6	99.37 ± 0.12	(32 768, 0.03125)	99.19 ± 0.19	(32 768, 2)
Monk1	99.95 ± 0.05	3	97.77 ± 2.95	0.1	94.63 ± 9.29	3	100	(32 768, 0.5)	100	(32 768, 6)
Monk2	67.04 ± 1.19	13	95.32 ± 2.22	0.4	79.68 ± 6.14	10	100	(32 768, 0.03125)	100	(32 768, 2)
Monk3	99.95 ± 0.05	1	100.0 ± 0.0	0.1	99.91 ± 1.44	2	100	(32 768, 0.5)	100	(32 768, 7)
Spect	80.91 ± 8.46	6	82.21 ± 6.38	0.3	82.75 ± 5.7	5	83.59 ± 0.55	(5, 8)	83.14 ± 1.03	(0.5, 1)
Voting	92.58 ± 3.21	2	95.1 ± 3.58	0.3	95.42 ± 2.68	2	96.37 ± 0.10	(32, 0.03125)	95.72 ± 0.21	(0.03125, 2)
Hayes-Roth	63.62 ± 9.38	5	82.57 ± 5.18	0.1	83.8 ± 6.64	5	79.70 ± 1.55	(32 768, 0.0078)	79.85 ± 2.05	(32, 1)
Diabetes	75.08 ± 3.53	20	74.73 ± 4.14	0.2	74.63 ± 5.22	5	77.37 ± 0.31	(8192, 3.051E − 5)	77.34 ± 0.30	(0.5, 1)
W. B. Cancer	96.66 ± 2.97	4	94.79 ± 3.19	0.2	95.87 ± 2.9	4	96.74 ± 0.12	(2, 2)	96.92 ± 0.23	(2, 1)
Heart	82.06 ± 8.82	10	78.34 ± 7.05	0.2	78.52 ± 7.32	4	79.98 ± 0.73	(32, 0.125)	83.77 ± 0.55	(0.5, 1)
Magic	78.4 ± 2.53	13	75.73 ± 2.55	0.3	76.69 ± 3.44	5	82.06 ± 0.23	(512, 0.125)	81.89 ± 0.45	(32, 3)
Ionosphere	90.83 ± 3.83	1	89.56 ± 5.62	0.1	89.01 ± 4.75	5	94.70 ± 0.32	(2, 2)	89.28 ± 0.43	(0.03125, 2)
Iris	94.99 ± 3.89	6	94.28 ± 5.19	0.2	93.65 ± 5.24	6	94.13 ± 1.28	(32 768, 0.5)	96.13 ± 0.99	(512, 1)
Wine	98.06 ± 2.81	8	94.23 ± 5.54	0.1	94.99 ± 3.49	8	98.20 ± 0.47	(32 768, 2)	98.53 ± 0.75	(2, 1)
Sat. Image	94.9 ± 2.04	1	92.71 ± 2.73	0.1	92.77 ± 3.48	3	96.01 ± 0.24	(8, 2)	95.11 ± 0.18	(0.5, 4)
Segment	96.46 ± 1.44	2	95.77 ± 1.77	0.2	94.55 ± 1.96	6	96.98 ± 0.25	(2048, 0.125)	97.14 ± 0.17	(8, 4)
Glass	72.87 ± 5.38	1	69.92 ± 7.4	0.2	69.06 ± 6.28	5	68.50 ± 1.57	(2, 8)	$73{,}01 \pm 1.50$	(2048, 2)
Letter	75.79 ± 3.32	1	63.38 ± 4.04	0.2	62.6 ± 5.42	9	83.59 ± 0.55	(32 768, 0.5)	82.93 ± 0.54	(0.5, 3)
Average	86.43		86.81		85.63		90.87		90.97	

TABLE 7 Classification accuracies given as mean and standard deviation with Odd_1, Odd_2, and Odd_3

Datasets	Odd_1	Odd_2	Odd_3
Balance	83.67 ± 3.82	58.19 ± 6.04	61.42 ± 7.08
Car	57.89 ± 7.73	77.81 ± 7.74	84.35 ± 3.87
Monk1	75.01 ± 6.53	74.92 ± 6.11	74.91 ± 4.71
Monk2	50.74 ± 9.11	50.92 ± 8.97	51.87 ± 7.92
Monk3	97.23 ± 1.78	97.23 ± 2.02	97.22 ± 2.67
Spect	44.02 ± 6.63	72.99 ± 9.34	84.32 ± 4.77
Voting	89.13 ± 5.34	89.42 ± 4.79	88.78 ± 4.7
Hayes-Roth	66.47 ± 9.64	76.87 ± 10.39	80.46 ± 8.03
Diabetes	75.05 ± 3.96	73.59 ± 3.13	72.55 ± 4.25
W. B. Cancer	94.17 ± 3.8	96.16 ± 2.89	97.01 ± 1.74
Heart	83.17 ± 6.77	82.53 ± 7.64	81.8 ± 6.33
Magic	61.48 ± 2.41	73.22 ± 2.96	71.79 ± 3.33
Ionosphere	69.38 ± 3.87	87.83 ± 4.28	86.6 ± 6.47
Iris	94.53 ± 7.28	94.98 ± 6.1	95.08 ± 4.51
Wine	94.95 ± 5.4	95.49 ± 5.34	95.92 ± 4.86
Sat. Image	86.89 ± 2.51	87.63 ± 2.93	88.6 ± 2.92
Segment	78.74 ± 3.55	85.44 ± 4.12	85.17 ± 2.37
Glass	37.29 ± 11.31	47.27 ± 12.23	48.48 ± 7.43
Letter	49.72 ± 3.72	59.2 ± 3.37	60.47 ± 5.59
Average	73.13	77.98	79.31

In Table 5, we provide mean accuracies and standard deviations for improved Odd_1, Odd_2, and Odd_3 in case of Boolean or numerical data. For each classifier, the displayed results correspond to the optimal value of the parameter β (for all *oddness*-classifiers, β being the number of nearest neighbors) obtained as a result of the optimization step in the inner cross-validation. We use an obvious notation for each classifier, $Odd_2(NN, Std)$ for instance means that we use a nearest neighbor and any other element to build a pair. In Table 5, we also save results for two supplementary tested *oddness*-classifiers named $Odd_2(NMRE)$ and $Odd_2(NN, NN)$. A full description of these two latter classifiers will be provided in Section 5.

As we can see in this Table 5, considering the average accuracy on all datasets, Odd_2 is the best performer for most datasets if compared to other options using singletons or triples, even if Odd_1, and Odd_3 remain quite close.

For this reason, we now investigate Odd_2 to get a better understanding of its behavior and then we compare it smoothly to other classifiers. We also propose other options for building pairs in the next section. In the following, we only focus on $Odd_2(NN, Std)$ classifier and we study the impact of parameter k on this classifier.

Study of impact of k on $Odd_2(NN, Std)$ performances

Looking at the results of $Odd_2(NN, Std)$ in Table 5, we can draw the following conclusions:

1. It is clear that this optimized classifier is significantly more efficient than the basic classifier Odd_2 for most datasets. The best accuracy for this option is noted for datasets: "Balance," "Car,"

"Spect," "Sat.Image," "Wine," and "Glass" having large number of attributes and/or classes. The average accuracy over all datasets is 87% for the $Odd_2(NN, Std)$ and 78% for the basic Odd_2.

2. Regarding the optimized parameter k, we can see that the best results are obtained with large values of k for some datasets such as: "Balance," "Monk2," and "Diabetes." For other datasets with large dimension such as "Sat.Image," "Segment," and "Letter," even very small values of k provide the best accuracies ($k = 1$ or 2). Since subsets of pairs are generally less informative than subsets of triples, it is better to consider, for this option, large values of k to take advantage of a larger variety of data.[7,8] It remains to investigate what would be a suitable value for k leading to the best accuracy for any data set.
3. It is quite clear that the proposed classifier, especially $Odd_2(NN, Std)$, performs well to classify numerical as well as Boolean data sets. These results highlight that the proposed multivalued oddness measure correctly extends the Boolean case.

4.5 Comparison with other classifiers

To evaluate the efficiency of the *oddness*-classifiers, we compare their accuracy to existing classification approaches:

- IBk: a k-NN classifier, we use the Manhattan distance and we tune the classifier on different values of the parameter $k = 1, 2, \ldots, 21$.
- C4.5: generating a pruned or unpruned C4.5 decision tree. We tune the classifier with different confidence factor used for pruning $C = 0.1, 0.2, \ldots, 0.5$.
- JRip: propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER), optimized version of IREP. We tune the classifier for different number of optimization runs $O = 2, 4, \ldots, 10$ and we apply pruning.
- SVM: a sequential minimal optimization algorithm for training a support vector classifier. We use two types of kernels: the RBF-Kernel and we tune its parameter $\gamma\,(\gamma = 2^{-15}, 2^{-13}, \ldots, 2^{3})$ and the Poly-Kernel and we tune its *degree* d ($d = 1, 2, \ldots, 10$; also called *exponent*). We also tune the complexity parameter $C = 2^{-5}, 2^{-3}, \ldots, 2^{15}$ with each type of kernel as recommended by Hsu et al.[12]

Accuracy results for IBk, C4.5, JRIP, and SVMs are obtained by using the free implementation of Weka software to the datasets described in Table 4. To run IBk, C4.5, and JRIP, we first optimize the corresponding parameter for each classifier, using the meta CVParameterSelection class provided by Weka. For SVM, since we have to tune both the complexity C (independent of the type of the kernel) as well as the kernel parameter γ for RBF-Kernel and the *degree* d for the Poly-Kernel, we use the GridSearch package available with Weka suitable for tuning two parameters. Both CVParameterSelection and GridSearch classes allow to perform parameter optimization by cross-validation applied to the training set only. This enables to select the best value of the parameter for each data set, then we train and test the classifier using this selected value of this parameter. Classification results for C4.5, JRIP, and SVMs, displayed in Table 6, correspond to the best/optimized value of each tuned parameter (denoted β in this table, for SVM β correspond to the best pair (C,γ) for RBF-kernel or (C,d) for Poly-kernel).

4.5.1 Evaluation of $Odd_2(NN, Std)$ classifier with respect to other classifiers

If we compare the results of $Odd_2(NN, Std)$ classifier to those of machine learning algorithms in Table 6, we note that:

1. The $Odd_2(NN, Std)$ classifier performs more or less in the same way as the best known algorithms. Especially, this classifier outperforms all other classifiers Except SVM for 13 out of 19 datasets. In particular, $Odd_2(NN, Std)$ is significantly better than *IBk* and SVM-based Poly-Kernel for datasets "Ionosphere," "Spect," and "Glass" and performs similar to SVM-based Poly-Kernel for datasets "Monk1," "Monk3," "W.B. Cancer," "Wine," and "Sat. Image."
2. If compared to *IBk* classifier, we can observe that $Odd_2(NN, Std)$ significantly outperforms *IBk* on datasets "Spect," "Voting," "Hayes-Roth," "Diabetes," "Ionosphere," "Magic," and "Glass" and has similar results for "Monk1," "Monk3," "Wine," "Sat.Image," "Segment," and "Letter."
3. The computed average accuracy over 19 datasets for each classifier, confirms our observations and shows that the $Odd_2(NN, Std)$ is ranked the second just after SVM classifier.
4. $Odd_2(NN, Std)$ shows high efficiency to classify datasets "Balance," "Car," "Sat.Image," "Segment," and "Glass" (which have multiple classes) which demonstrates its ability to deal with multiple class data sets.
5. $Odd_2(NN, Std)$ seems to be also efficient when classifying data sets with a large number of attributes as in the case of "Car" and "Ionosphere" and large number of instances as in the case of "Magic," "Sat.Image," and "Segment" for instance.
6. $Odd_2(NN, Std)$ has close classification results to those of analogy-based classifier in the numerical case[3,13] for most datasets. In the Boolean case, both *oddness*-based and *analogy*-based classifiers[2,3] achieve good results for "Balance," "Car," "Monk1," and "Monk3." For "Monk2" data set, Analogy-based classifier significantly outperforms $Odd_2(NN, Std)$ while for "Spect" and "Voting" the converse is observed. However, even if there is a path (through logical proportions, in the Boolean case) relating the respective building blocks on which analogy-based classifiers and the classifiers studied here are based, the two types of classifiers seem to rely on different ideas: the control of the dissimilarity via the oddness measure, and the fact of privileging linearity in the other case.[14]

The above results suggest to focus more on the Odd_2. In the next section, we investigate further options to select suitable pairs in a class.

5 EXPERIMENTING WITH MORE CONSTRAINED PAIRS

We have seen that selecting one element of a pair as a nearest neighbor in the target class of the item to be classified leads to good accuracy rates. So, why not to also carefully select the second element of the pair? This is what we do in the following sections, first by choosing a second element very far from the item to be classified, then by choosing the second element as another nearest neighbor in the class.

5.1 Odd2 using a remote element

A drastic option is to consider the second element of the pair as being among the *Most Remote Elements* (MRE) to the item $\vec{x}$. Indeed, we may think that pairs including an MRE are more informative since they allow to sample a larger variety of data, which may be of interest especially when the function underlying the classification is complex. The intuition behind taking $\vec{b}$ as an *MRE* might be justified by the fact that, in this case, the interval of values $[a_i, b_i]$, $i = 1, \ldots, m$ remains sufficiently large since vectors $\vec{a}$ and $\vec{b}$ are very different (*min* and *max* are more informative here). This will guaranty to get sufficiently *high* values of atomic $odd(x_i, \{a_i, b_i\})$. Cumulating these elementary *odd* values through pairs may contribute to quickly converge to the appropriate class. In that case, $Odd(\vec{x}, \mathscr{C})$ is the sum of $k \times k^{'}$ atomic oddness values and then belong to the interval $[0, k \times k^{'}]$. In the following, we denote *NMRE*, the classifier using this option.

The algorithm *NMRE* has two main parameters to be tuned: k the number of nearest neighbors and $k^{'}$ the number of most remote elements which are taken into account to compute the oddness measure. In the following, we investigate the behavior of *NMRE* algorithm on all the experimented datasets. In Table 5, we provide the accuracy results of *NMRE*-classifier (denoted: $Odd_2(NMRE)$). These results correspond to the optimal values of k and $k^{'}$ obtained as a result of the optimization step in the inner cross-validation.

From Table 5, we can draw the following comments:

1. *NMRE* classifier provides good results for large values of k or $k^{'}$ on most datasets. This suggests that, for small values of k, there is not enough information allowing to properly classify.
2. The average value of parameter $k^{'}$ over all datasets is higher than that of parameter k.
3. *NMRE* is efficient to classify both Binary or numerical datasets as in the case of Odd_2 classifier.
4. If compared to $Odd_2(NN, Std)$, the *NMRE* seems less efficient on almost all datasets except for Monk2 for which it provides better results.

These primary observations lead us to test another option.

5.2 Odd2 using two nearest neighbors

The previous experiments have shown that choosing one element in the pair contributes to improve the performance w.r.t. the case where any subsets were considered. Nevertheless, choosing the second element of the pair as the most remote element does not really help to improve the accuracy. So we are led to the idea of still choosing this second element as one of the k nearest neighbors in the class, keeping it distinct from the first one. And the normalization factor is chosen accordingly as $\frac{1}{\binom{k}{2}}$. This is also clearly beneficial from a complexity viewpoint.

In Table 5, we also show classification accuracy of Odd_2 classifier, in which each pair element is among the k nearest neighbors in the class. We denote this classifier $Odd_2(NN, NN)$.

Results of $Odd_2(NN, NN)$ in Table 5 shows that:

1. If we compare results of the classifier using pairs with two nearest neighbors to those of the basic Odd_2 classifier in Table 7, it is clear that this third optimized option also performs largely better than the basic Odd_2 classifier.
2. To compare the $Odd_2(NN, NN)$ to the $Odd_2(NN, Std)$ classifier, in the $Odd_2(NN, NN)$ column of Table 5 we assign a positive "+" mark if the $Odd_2(NN, NN)$ is better than the $Odd_2(NN, Std)$, a negative mark "−" in the opposite case and a neutral mark "." if they have equivalent accuracies. This comparison shows that the two classifiers have close efficiency for many datasets. Especially, the $Odd_2(NN, NN)$ is slightly better for seven datasets and worst for five datasets.

5.3 | Statistical evaluation of *Oddness*-classifiers

The comparative studies with other classifiers are first carried out through the Friedman test.[15] It is a nonparametric test used to detect differences in n treatments groups with equal sample sizes across multiple test attempts. The null hypothesis, $H0$: $F(1) = F(2) = \cdots = F(n)$ states that there is no significant difference between groups of algorithms against the alternative hypothesis: at least one group is significantly different. In case the Friedman test indicates significance, a post-hoc test after Conover[16] is applied to calculate the corresponding levels of significance. The output of this test is simply the computed P-values corresponding to each pair of compared groups saved in the lower triangle of the P-values matrix. For this test, we also apply a Bonferroni-type adjustment of P-values.

In a preliminary step, we aim to compare between *Oddness* classifiers using singletons, pairs, or triples to check our first observations in previous sections. For this reason, we compare between Odd_1, Odd_2, Odd_3, $Odd_1(NN)$, $Odd_2(NN, Std)$, and $Odd_3(NN, Std, Std)$. This comparison confirms that $Odd_2(NN, Std)$ is significantly better than *all* other compared classifiers using singletons or triples with a P-value at least equal to 0.00644 as a result of the post-hoc test after Conover.

Since *Oddness*-classifiers using pairs are the best among other family of *Oddness* classifiers, in the following we restrict our evaluation to the efficiency of $Odd_2(NN, Std)$, $Odd_2(NMRE)$, and $Odd_2(NN, NN)$ and we compare their accuracy to that of other machine learning classifiers. Since the Friedman test, provided a significant P-value$= 1.295\mathrm{e} - 9$, we then apply a post-hoc test after Conover and we use Bonferroni-type adjustment of the P-values.

In Table 8, we provide the results of the computed P-values for each pair of compared classifiers. Significant P-values (<0.05) are given in bold.

The computed P-values are consistent with the following observations:

1. The SVM using (RBF or Poly-Kernel) significantly outperforms the IBK, the C4.5, and the JRIP classifiers. It also outperforms $Odd_2(NMRE)$ classifier. While SVM using Poly-Kernel seems better than $Odd_2(NN, Std)$.
2. It is clear that $Odd_2(NN, Std)$ significantly outperforms the IBK, the C4.5, the JRIP classifiers and also $Odd_2(NMRE)$.
3. This demonstrates that the proposed first optimization not only reduces the complexity which becomes linear, but also considerably improves the classification accuracy if compared to the basic classifier without any optimization or to IBK classifier.
4. The $Odd_2(NN, NN)$ classifier is largely better than IBK, the C4.5, and the JRIP.

TABLE 8 Results for the Wilcoxon Matched-Pairs Signed-Ranks Test, The * (resp. $_*$) means that the classifier in the row (resp. in the column) is statistically better than the classifier on the column (resp. on the raw)

	IBK	C4.5	JRIP	SVM(RBFKernel)	SVM(PolyKernel)	$Odd_2(NN, Std)$	$Odd_2(NMRE)$
C4.5	1	–	–	–	–	–	–
JRIP	0.09433	1	–	–	–	–	–
SVM(RBFKernel)	$9.5e-08$*	$1.8e-11$*	$1.2e-14$*	–	–	–	–
SVM(PolyKernel)	$2.8e-09$*	$4.0e-13$*	$2.3e-16$*	1	–	–	–
$Odd_2(NN, Std)$	0.01172*	$1.5e-05$*	$2.7e-08$*	0.20023	0.02257$_*$	–	–
$Odd_2(NMRE)$	1	1	0.07757	$1.3e-07_*$	$3.9e-09_*$	0.01462$_*$	–
$Odd_2(NN, NN)$	0.00011*	$5.1e-08$*	$5.1e-11$*	1	0.77887	1	0.00014*

TABLE 9 Comparative study between oddness classifiers and IBk in terms of procedure and complexity

	Procedure	**Complexity**
Odd_3	- Compute the average distance to *allpossibletriples* of items in each class.	$O(\lvert C\rvert^3)$
	- Assign to $\vec{x}$ the class with the shortest distance.	
$Odd_3(NN, Std, Std)$	- Select the k-NN of $\vec{x}$ in each class.	$O(k^*\lvert C\rvert^2)$
	- Compute the average distance to *all possible item triples* in each class where *only* one of the items is among the k-NN.	
	- Assign to $\vec{x}$ the class with the shortest distance.	
Odd_2	- Compute the average distance to *all possible pairs* of items in each class.	$O(\lvert C\rvert^2)$
	- Assign to $\vec{x}$ the class with the shortest distance.	
$Odd_2(NN, Std)$	- Select the k-nearest neighbors of $\vec{x}$ in each class.	$O((k^*\lvert C\rvert))$
	- Compute the average distance to *all possible pairs* of items in each class where *only* one of the items is among the k-nearest neighbors.	
	- Assign to $\vec{x}$ the class with the shortest distance.	
$Odd_2(NN, NN)$	- Select the k-nearest neighbors of $\vec{x}$ in each class.	$O(k^*(k-1)/2)$
	- Compute the average distance to *all possible pairs of nearest neighbors* in each class.	
	- Assign to $\vec{x}$ the class with the shortest distance.	
Odd_1	- Compute the average distance to *all* items in each class.	$O(\lvert C\rvert)$
	- Assign to $\vec{x}$ the class with the shortest distance.	
$Odd_1(NN)$	- Select the k-nearest neighbors of $\vec{x}$ in each class.	$O(k)$
	- Compute the average distance to *each nearest neighbor* in each class.	
	- Assign to $\vec{x}$ the class with the shortest distance.	
IBk	- Select the overall k-nearest neighbors of $\vec{x}$ in TS regardless of the class.	$O(k)$
	- Assign to $\vec{x}$ the most frequent class among its k-nearest neighbors.	

5. As in the case of the $Odd_2(NN, Std)$ classifier, the $Odd_2(NN, NN)$ is also statistically better than the $Odd_2(NMRE)$ classifier.
6. Since the computed P-value is not significant, no clear conclusion can be stated regarding $Odd_2(NN, Std)$ and $Odd_2(NN, NN)$. This is obvious from Table 5 since they have close average accuracies over the 19 datasets.

The last results suggest that using subsets made of pairs of two nearest neighbors is enough to achieve good results and there is no need to cross all the training set to construct pairs as in the $Odd_2(NN, Std)$ classifier for example. In view of the accuracy results, these pairs could be considered as representative of the training data for the considered class.

6 NEAREST NEIGHBORS IN ODDNESS CLASSIFIERS AND COMPARISON WITH THE STANDARD k-NN

Lastly, we study the difference between all proposed oddness classifiers and also the standard k-NN classifier in terms of procedure and complexity.

It is worth noticing that although we use k nearest neighbors, this leads to a method that differs from the standard k-NN classifier at list in two sides. First, we use the nearest neighbors in a given class, and we do it for each class. In fact, we consider local nearest neighbors instead of global ones as in k-NN. It means that we are considering the same number of nearest neighbors for each class, while the k-NN are not, in general, uniformly distributed over the classes.

Second, oddness classifiers benefits from averaging the distance of $\vec{x}$ to its k nearest neighbors in each class when selecting the best class, while standard k-NN method applies directly a vote on the k nearest neighbors labels without computing the distance to these nearest neighbors. It's known that, the efficiency of the "majority voting" classifiers can be significantly decreased in case there is a most frequent that may dominate the prediction of the classified instances, since this class tends to be common among the k nearest neighbors.[17]

The experimental results given in the previous sections (see for example first column in Table 5) confirm this difference since we observe that $Odd_1(NN)$ and $Odd_2(NN, NN)$ perform quite differently than k-NN on some benchmarks even they have close complexity.

In Table 9, we provide a comparative study by summarizing the basic logic behind each proposed classifier and a detailed complexity evaluation.

It is clear that the complexity is significantly reduced when we use $Odd_3(NN, Std, Std)$ instead of Odd_3 and $Odd_2(NN, Std)$ instead of Odd_2. This drop in complexity is more achieved with $Odd_2(NN, NN)$ classifier if compared to $Odd_2(NN, Std)$ mainly for data sets with large number of examples. Evidently, this has a considerable impact on the run time. If we consider the case of Monk1 when C is equal to 90% of the whole data set in each class, the total number of pairs that can be built from the used part of the data set for each class is:

1. more than 18 000 for Odd_2,
2. Almost 2900 for the $Odd_2(NN,Std)$ with $k = 15$,
3. Only about 100 for $Odd_2(NN,NN)$ with $k = 15$.

7 | CONCLUSION

Classifying an item by choosing the class where it is the less at odds may be considered as a basic idea in machine learning. In this paper, we have studied an original way to estimate oddness. We have started from a logical interpretation of oddness to define a logical oddness index, indicating if a given item is at odds or not w.r.t. a multiset of n values, both in the Boolean case and in the real case. Then, by adding these atomic oddness indices, we get a global oddness measure of this item w.r.t. a class. With diverse choices of n, we implement this notion to build classifiers. It appears that using pairs ($n = 2$) provides good accuracy results while giving a lower complexity. We investigate deeper this option by filtering the candidate pairs for a given class. To do so, we select the elements of the pair in various ways. First, we fix one element of the pair as a nearest neighbor in the class of the item to be classified. Then, we fix the second one as a most remote element: this leads to a degraded accuracy. Finally, we consider the second element as a second nearest neighbor in the class. Our experiments on UCI benchmarks show that we are still competitive with regard to state of the art classifiers (k-NN, SVM) while having drastically decreased the complexity. Indeed, as datasets become bigger and bigger, the scalability of oddness-based classifiers is paramount: this has to be further investigated in future works. While using the classical notion of nearest neighbor, our handling of them is quite different from the one in k-NN methods. It is also worth noticing that the idea of no longer estimating similarity on a pure one-to-one basis, but rather in considering triples or quadruplets (including the new item) can be encountered elsewhere in machine learning,[18] in relation with the idea of comparative proportions, which is not far from the notion of logical proportions to which our view of oddness is related.

REFERENCES

1. Miclet L, Bayoudh S, Delhay A. Analogical dissimilarity: definition, algorithms and two experiments in machine learning. *JAIR*. 2008;32:793-824.
2. Bounhas M, Prade H, and Richard G. Analogical classification: A new way to deal with examples. ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August, Prague, 135-140.
3. Bounhas M, Prade H, Richard G. Analogy-based classifiers for nominal or numerical data. *Int J Approx Reasoning*. 2017;91:36-55.
4. Prade H, Richard G. Reasoning with logical proportions. In: Lin FZ, Sattler U, Truszczynski M, eds. *Proceedings of 12th International Conference on Principles of Knowledge Representation and Reasoning, KR 2010, Toronto, May 9-13, 2010*. AAAI Press; 2010:pp. 545-555.
5. Prade H, Richard G. Homogenous and heterogeneous logical proportions. *IfCoLog J Logics Appl*. 2014;1(1):1-51.
6. Bounhas M, Prade H, Richard G. Evenness-based reasoning with logical proportions applied to classification. *Proceedings of Scalable Uncertainty Management -9th International Conference, SUM*. Springer; 2015:139-154. volume 9310 of LNCS.
7. Bounhas M, Prade H, and Richard G.Oddness-based classifiers for Boolean or numerical data.: Advances in Artificial Intelligence - 38th Annual German Conference on AI Dresden, Germany, September 21-25, 2015, 32-44, 2015.
8. Bounhas M, Prade H, Richard G. Oddness/eve nness-based classifiers for Boolean or numerical data. *Int J Approx Reasoning*. 2017;82:81-100.
9. Bounhas M, Prade H, Richard G. Not being at odds with a class: A new way of exploiting neighbors for classification. In: Kaminka GA, Fox M, Bouquet P, Hüllermeier E, Dignum V, Dignum F, vanHarmelen F, eds. *22nd European Conference on Artificial Intelligence, 29 August - 2 September 2016, The Hague*. IOS Press; 2016:pp. 1662-1663. volume 285 of Frontiers in Artificial Intelligence and Applications.
10. Rescher N. *Many-valued logic*. New York: McGraw-Hill Inc; 1969.
11. Mertz J and Murphy PM. Uci repository of machine learning databases: ftp://ftp.ics.uci.edu/pub/machine-learning-databases, 2000.
12. Hsu CW, Chang CC, Lin CJ. A practical guide to support vector classification. 2010.
13. Bounhas M, Prade H, Richard G. Analogical classification: Handling numerical data. In: Straccia U, Cali` A, eds. *Proceedings of the 8th International Conference on Scalable Uncertainty Management, SUM*. Springer; 2014:pp. 66-79. volume 8720 of LNCS.
14. Couceiro M, Hug N, Prade H, et al. Analogy-preserving functions: A way to extend Boolean samples, Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence Main track, IJCAI 2017, Melbourne, August 19-25.1575-1581, 2017.
15. Friedman M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *J Am Stat Assoc*. 1937;32(200):675-701.
16. Conover WJ. *Practical Nonparametric Statistics*. New York: John Wiley; 1971.
17. Coomans D and Massart DL. Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-nearest neighbour classification by using alternative voting rules, Anal Chim Acta, 15-27, 1982.

18. Law MT, Thome N, and Cord M.Quadruplet-wise image similarity learning. 2017 IEEE International Conference on Computer Vision (ICCV), 2013.

https://doi.org/10.1002/int.22035