Logical Proportions for Classification and Preference Learning

Myriam Bounhas

Université de Tunis
Institut Supérieur de Gestion
Ecole Doctorale Sciences de Gestion

Logical Proportions for Classification and Preference Learning

HABILITATION UNIVERSITAIRE en Gestion

Présentée et soutenue publiquement par:

Dr. Myriam BOUNHAS

Le 29 Juin 2021

Membres du Jury:

♦ *M. Salah BENABDALLAH*	*: Professeur à l'Université de Tunis*	*: Président*
♦ *M. Lamjed BEN SAID*	*: Professeur à l'Université de Tunis*	*: Rapporteur*
♦ *M. Slim BECHIKH*	*: Maître de Conférences à l'Université de Carthage*	*: Rapporteur*
♦ *Mme Nahla BEN AMOR*	*: Professeur à l'Université de Tunis*	*: Membre*
♦ *M. Zied ELOUEDI*	*: Professeur à l'Université de Tunis*	*: Membre*
♦ *M. Henri PRADE*	*: Dir. de Recherche CNRS Émérite à l'IRIT, Toulouse (France)*	*: Invité*

Année Universitaire: 2020-2021

In memory of my father,

In memory of my aunt,

To my mother, for all her sacrifices and patience,

To my husband, my daughters Issra and Yasmine and my son Adam for being great pillars of support and love,

Myriam Bounhas

June, 2021

Acknowledgements

First and most of all thanks to GOD ALMIGHTY for without His graces and blessings this habilitation would not have been possible. I owe my gratitude to all those people who have contributed to the production of this work and because of whom my research experience has been developed.

Foremost, I would like to thank my principal collaborator Dr. Henri Prade. I have been fortunate to have a research tutor who gave me the opportunity to explore on my own and at the same time the guidance to recover when my steps faltered. I especially want to thank Dr. Henri whose support and guidance made my habilitation possible. He has been actively interested in my ideas and has always been available to advise me even it sometimes represents an imposition on his time to perform this. I am very grateful for his patience, motivation, enthusiasm, and immense experience that, taken together, make him a great mentor.

I also owe a debt of gratitude to my collaborator Pr. Gilles Richard that have been always there to listen and give advice. I am deeply grateful to him for his encouragement and support; commenting on my views and helping me understand and enrich my ideas. I am also thankful to him for the long discussions that helped me sort out the technical details of my work.

I want to express my thinks and gratitude to my co-authors Pr. Marc Pirlot and Olivier Sobrie, for their professionalism, the valuable discussions and comments regarding my research that made an important part of this habilitation completed.

Besides my co-authors, I would like to specially thank Pr. Nahla Ben Amor for providing me advising and support that made this work achievable and for Dr. Mathieu Serrerier for helping me to perform hard and long experimentations on his machine.

I would like to thank the rest of my habilitation committee: Pr. Salah Ben Abdallh, Pr. Lamjed Ben Said, Pr. Slim Bechick, Pr. Zied Elouedi and Pr. Nahla Ben Amor for their insightful comments, and interesting questions.

I want to express my thinks and love to my mother, my brothers and my aunt Tibra

whose sacrifices and great efforts have enabled me to reach the present position in life. I am forever indebted to my mother and my aunt for their unceasing prayers and patience.

There are no words to say thank you to my husband Dr. Bilel Elayeb who has been a true and great supporter during my good and bad times and has unconditionally helped me scientifically and morally to attain my goal. These past several years have not been an easy ride, both academically and personally. I truly thank Bilel for his sacrifices and for sticking by my side, even when I was irritable and depressed. This work would not have been possible without your support.

Finally, I would like to thank Elayeb family, especially Ms. Fatma Ben Smida and Mr. Nafti Elayeb, my sisters and brothers in law for being always on my side.

Abstract

Analogy is at the heart of human thinking that an individual can usually apply when he/she is confronted with new situations. Analogical reasoning helps to address unknown situations based on what we already know. It has been proven that this type of reasoning is mostly naturally applied by a child to approach what he does not know.

Analogical reasoning in AI consists to develop analogical learning systems (AAS) that mimic humans' ability to learn by analogy. Closely related to analogical reasoning, the idea of analogical proportions makes a comparison between pairs of situations. Analogical proportions are relations of the form "A is to B as C is to D", denoted as A: B :: C: D. Such a proportion expresses that "A differs from B as C differs from D". In other words, the pair (A, B) is analogous to the pair (C, D). Analogical proportions are a special case of a more general case which are the logical proportions.

Several classification methods based on logical proportions have been proposed in the last decade. Given an object D to be classified, the basic idea of such classifiers is to look for triples of examples (A, B, C), in the learning set, that form a logical proportion with D, on a maximum set of attributes. In this context, we assume that objects or situations A, B, C, D are represented by a vector of attribute values.

In this habilitation, we investigate the possibility of applying logical proportions as a basic tool to address two well-known problems in machine learning: classification and user's preferences learning. On the first hand, we studied logical proportions as a basic tool for solving classification problems. We have proposed two families of classifiers based on logical proportions and dealing with different types of data for this purpose: the first family, using homogeneous proportions (known as analogical proportions), is based on the idea of searching for triples of examples in the training set that form a $valid$ analogical proportion with the new example to be classified. These classifiers are denoted AP-classifiers. The second family, using rather heterogeneous proportions, is based on the idea that an example should be added to the class where it is the least at odds w.r.t. the elements already in the class. We evaluate the index of oddness of an item with regard to all items of the class for this effect. The experimentation of such classifiers shows their efficiencies in terms of classification of nominal and numerical data.

On the other hand, we study the problem of preferences learning in which we also make predictions as in case of classification. However, in this case, the goal is to predict a user's preference between two given choices. For this end, we propose and compare two approaches based on analogical proportions: the first one uses triples of pairs of items for which preferences are known and which make analogical proportions, altogether with the new pair to be predicted while making sure that no contradictory trade-offs are created. The second approach exploits pairs of compared items one by one: for predicting the preference between two items, one looks for another pair of items for which the preference is known such that, attribute by attribute, the change between the elements of the first pair is the same as between the elements of the second pair. The reported experiments, both on real data sets and on generated datasets suggest the effectiveness of these approaches.

Key words: Analogical Proportions, Logical Proportions, Classification, Analogy-based classification, Oddness-based classification, Evenness-based classification, Oddness, Evenness, Preference Learning, Analogy-based Preference Learning, Weighted sum, Sugeno Integral.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction and background on logical proportions

1.1 Introduction

Knowledge reasoning is the basic component for developing intelligent learning systems (ILSs) in different domains. An intelligent system inquires knowledge from the environment that may be represented in a particular way such as a set of facts, symbols, or relationships. This knowledge needs to be stored, organized and analyzed in a pre-processing phase by ILSs.

Once a knowledge base is built, such system must also be able to acquire new knowledge by interacting again with the environment and then to infer new knowledge based on previously stored facts in the knowledge base. This inference process is useful for solving problems and making decisions regarding new situations. Although, a computer is not always able to think as a human expert can, the process of producing new knowledge, by inferring known facts, may be considered as a kind of creating intelligence that allows to the computer to make reasoning and judgments in a similar way as human do.

A panoply of artificial intelligence techniques has been developed in order to imitate intelligent human behaviour. Using the available knowledge on a given problem, these techniques try to reproduce the process of human reasoning as close as possible. The general issue of simulating intelligence has been applied in a number of specific sub-problems such as: deduction, learning, reasoning, decision making, and knowledge representation.

$Machine\ Learning$ is one of the most central branches of AI. It consists in the development of programs that can process data and learn on their own, without our continuous supervision. There are many popular ML applications that concern a variety of fields. For example, we can cite the pattern recognition, such as text and voice recognition, videos surveillance, fraud detection and game programs. In machine learning, there are different

domains of interest, especially supervised, unsupervised and semi-supervised learning depending on whether outputs (for the given problem) are given or not for the learner when accomplishing the learning task. Among these domains we are interested to supervised learning and especially to two topics of interest $Classification$ and *Preference learning*. The crossing point between these two domains is in the prediction power: in classification, the goal is to predict the $label$ of the class while in preference learning the goal is to predict the *preference relation* between two items.

In the last decade, analogical reasoning has received increasing attention within the community of AI-based learning systems as a tool for problem solving, deduction and learning. Analogy is at the heart of human thinking that every one may apply when he/she is faced to new situations. Analogical reasoning helps to approach unknown situations from what we already know [61]. It has been proven that this is the most natural way children make reasoning about what they don't know. For example in psychology, many reasoning tests are based on analogy for evaluating intelligence. The main interest of analogical reasoning in AI is on developing analogical learning systems (ALSs) that mimic the ability of humans to learn by analogy. Closely related to analogical reasoning, the idea of *analogical proportions* establishes a parallel between pairs of situations. These proportions are a special case of a more general *logical proportions*.

In this habilitation, we focus on studying logical proportions that are the basic ingredients of all approaches that we propose as a solution to tackle classification and preference learning problems. Logical proportions are Boolean formulas involving 4 variables. They have been deeply investigated in [53]. In this chapter, we first recall the background on logical proportions and we mainly focus on the proportions that we will use as a tool for classification and preference learning solving problems. Then we introduce the two research axes developed in this habilitation thesis. The first axe tackles the use of logical proportions as a suitable tool for classification. Similarly to analogical reasoning, regarded as useful for drawing conclusions, we expect that analogical proportions may help to classify examples using an analogical inference that we define in the next chapter. The second axe is related to preference learning in which we still make prediction as in case of classification, however, the goal in this case is to predict user preferences between two items or examples.

This chapter is organized as follows. The next section recall the basic definitions and main properties of logical proportions. We distinguish two types of these proportions, the homogeneous and heterogeneous proportions and we discuss the characteristics of each of them. In Section 3, we describe the research axes achieved in this habilitation thesis.

1.2 Background on logical proportions

A logical proportion states a relation between 4 items that is expressed in terms of comparisons between pairs of items, each item being represented as a set of Boolean features. Considering 2 Boolean variables a and b corresponding to the same feature attached to 2 items A and B, $a \wedge b$ and $\overline{a} \wedge \overline{b}$ indicate that A and B behave similarly w.r.t. the given feature (they are called "similarity" indicators), $a \wedge \overline{b}$ and $\overline{a} \wedge b$ indicate that A and B behave differently (they are called "dissimilarity" indicators). When we have 4 items A, B, C, D, for comparing their respective behavior in a pairwise manner, we are led to consider logical equivalences between similarity, or dissimilarity indicators, such as $a \wedge b \equiv c \wedge d$ for instance. This enables us to define a logical proportion [53]:

Definition 1 *A logical proportion $T(a, b, c, d)$ is the conjunction of two equivalences between indicators for (a, b) on one side and indicators for (c, d) on the other side.*

For instance, $((\overline{a} \wedge \overline{b}) \equiv (c \wedge \overline{d})) \wedge ((\overline{a} \wedge b) \equiv (\overline{c} \wedge d))$ is a logical proportion. It has been established that there are 120 syntactically and semantically distinct logical equivalences. There are two ways for distinguishing remarkable subsets among the 120 proportions: either by investigating their structure, or by investigating their semantics (i.e. their truth table). In this section, we shall see that both investigations lead to the same conclusion: there is a class of 8 proportions which stands out of the crowd. This class can be subdivided into 2 sub-groups of 4 proportions.

Indeed a property that appears to be paramount in many reasoning tasks is code independency: there should be no distinction when encoding information positively or negatively. In other words, encoding truth (resp. falsity) with 1 or with 0 (resp. with 0 and 1) is just a matter of convention, and should not impact the final result. When dealing with logical proportions, this property is called *code independency* and can be expressed as

$$T(a, b, c, d) \rightarrow T(\overline{a}, \overline{b}, \overline{c}, \overline{d})$$

From a structural viewpoint, remember that a proportion is built up with a pair of equivalences between indicators chosen among 16 equivalences. So, to ensure code independency, the only way to proceed is to first choose an equivalence then to pair it with its counterpart where every literal is negated: for instance $a \wedge b \equiv \overline{c} \wedge d$ should be paired with $\overline{a} \wedge \overline{b} \equiv c \wedge \overline{d}$ in order to get a code independent proportion. This simple reasoning shows that we have only 16/2 = 8 code independent proportions whose logical expressions are given below.

$$\mathbf{A}\text{:}((a \wedge \overline{b}) \equiv (c \wedge \overline{d})) \wedge ((\overline{a} \wedge b) \equiv (\overline{c} \wedge d))$$

$$\mathbf{R}\text{:}((a \wedge \overline{b}) \equiv (\overline{c} \wedge d)) \wedge ((\overline{a} \wedge b) \equiv (c \wedge \overline{d}))$$

$$\mathbf{P}\text{:}((a \wedge b) \equiv (c \wedge d)) \wedge ((\overline{a} \wedge \overline{b}) \equiv (\overline{c} \wedge \overline{d}))$$

$$\mathbf{I}\text{: } ((a \wedge b) \equiv (\overline{c} \wedge \overline{d})) \wedge ((\overline{a} \wedge \overline{b}) \equiv (c \wedge d))$$

$$\mathbf{H_1}\text{: } ((a \wedge \overline{b}) \equiv (c \wedge d)) \wedge ((\overline{a} \wedge b) \equiv (\overline{c} \wedge \overline{d}))$$

$$\mathbf{H_2}\text{: } ((\overline{a} \wedge b) \equiv (c \wedge d)) \wedge ((a \wedge \overline{b}) \equiv (\overline{c} \wedge \overline{d}))$$

$$\mathbf{H_3}\text{: } ((a \wedge b) \equiv (c \wedge \overline{d})) \wedge ((\overline{a} \wedge \overline{b}) \equiv (\overline{c} \wedge d))$$

$$\mathbf{H_4}\text{: } ((a \wedge b) \equiv (\overline{c} \wedge d)) \wedge ((\overline{a} \wedge \overline{b}) \equiv (c \wedge \overline{d}))$$

Only 4 among these proportions make use of similarity and dissimilarity indicators without mixing these types of indicators inside one equivalence: for this reason, these 4 proportions A, R, P, I are called *homogeneous proportions*. For instance, an informal reading of A would be: "a differs from b as c differs from d and vice versa." This expresses the meaning of an analogical proportion, i.e., a statement of the form "a is to b as c is to d". We can consider A as a Boolean counterpart to the idea of numerical proportion, either geometric, i.e., $\frac{a}{b} = \frac{c}{d}$, or arithmetic $a - b = c - d$.

The idea of a proportion suggests that some stability properties hold w.r.t. permutations. Indeed, we can permute variables and check, for instance, if a given proportion still holds when permuting the 2 first variables. We denote p_{ij} the permutation of variable in position i with variable in position j. For instance, p_{14} permutes the variables in extreme positions 1 and 4, while p_{23} permutes variables in mean positions. And $p_{12}(a) = b, p_{12}(b) = a, p_{12}(c) = c, p_{12}(d) = d$.

Definition 2 *A proportion T is stable w.r.t. permutation p_{ij} iff*

$$T(a, b, c, d) \rightarrow T(p_{ij}(a), p_{ij}(b), p_{ij}(c), p_{ij}(d))$$

It can be checked that A is stable w.r.t. the extremes p_{14} or the means p_{23} permutations. P is stable for p_{12} and p_{34} permutations, while R is stable for p_{13} and p_{24}. Moreover A, R, P, I are symmetrical (i.e. $T(a, b, c, d) \rightarrow T(c, d, a, b)$). This is observable on their truth tables: See the top part of Table 1.1, where only the 6 patterns that make the logical proportions true appear). Besides, I is the only logical proportion that is stable w.r.t. any permutation of two of its variables. This noticeable result is proved in [51].

Moreover, R and P are closely related to A via permutations. Namely we have

$$A(a, b, c, d) \equiv P(c, b, a, d) \equiv R(b, a, c, d)$$

In fact, when d is fixed, exchanging the variables a, b, c amounts to move from one homogeneous proportion to another, or to remain stable ($A(a,b,c,d) = A(a,c,b,d)$; $P(c,b,a,d) = P(b,c,a,d)$; $R(b,a,c,d) = R(c,a,b,d)$), with I remaining an exception. Thus A, R, P collectively maintain a form of exchangeability property with respect to a, b, c, while I ensures it by itself. These exchangeability properties are of particular interest when applying homogeneous logical proportions to classification.

The 4 remaining code independent logical proportions H_1, H_2, H_3, H_4 are called *heterogeneous proportions*: it is clear from their logical expression that they mix similarity and dissimilarity indicators inside each equivalence. Their truth tables are shown in the bottom part of Table 1.1, where only the 6 patterns that make them true appear. The index i in H_i refers to a position inside the formula $H_i(a,b,c,d)$. Namely, as can be checked, in each of these 6 patterns there is a minority value (i.e., the value having the smallest number of occurrences in the pattern, then this value is like an intruder among the other values), and i is the only position where the minority value never appears among the 6 4-tuples of values that make H_i true.

Table 1.1: Homogeneous/heterogeneous proportions valid patterns

A	**R**	**P**	**I**
0 0 0 0	0 0 0 0	0 0 0 0	1 1 0 0
1 1 1 1	1 1 1 1	1 1 1 1	0 0 1 1
0 0 1 1	0 0 1 1	1 0 0 1	1 0 0 1
1 1 0 0	1 1 0 0	0 1 1 0	0 1 1 0
0 1 0 1	0 1 1 0	0 1 0 1	0 1 0 1
1 0 1 0	1 0 0 1	1 0 1 0	1 0 1 0
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

By examining the truth table of the heterogeneous proportions in Table 1.1, we get the following properties:

$$A(a,b,c,d) \wedge R(a,b,c,d) \wedge P(a,b,c,d) \wedge I(a,b,c,d) = \bot$$

together with

$$(A(a,b,c,d) \wedge R(a,b,c,d) \wedge P(a,b,c,d)) \equiv Eq(a,b,c,d)$$

where $Eq(a,b,c,d) = 1$ if $a = b = c = d$ and $Eq(a,b,c,d) = 0$ otherwise. Similarly, for heterogeneous proportions, we have

$$H_1(a,b,c,d) \wedge H_2(a,b,c,d) \wedge H_3(a,b,c,d) \wedge H_4(a,b,c,d) = \bot$$

which implies:

$$(H_1(a,b,c,d) \wedge H_2(a,b,c,d) \wedge H_3(a,b,c,d)) \rightarrow \neg H_4(a,b,c,d) \tag{1.1}$$

Obviously, we have similar properties by permuting the indexes of the H_i's. The meaning of the conjunction $H_1(a,b,c,d) \wedge H_2(a,b,c,d) \wedge H_3(a,b,c,d)$ will be discussed in the following section, devoted to heterogeneous proportions.

1.2.1 Specificity of Heterogeneous proportions

In order to get a clear understanding of the heterogeneous proportions and to extract relevant properties, we now investigate their truth tables.

Heterogeneity and exchangeability

Still within Table 1.1, an obvious semantics appears: H_i holds when there are exactly 3 parameters with identical Boolean values (=1 for example) and the parameter in position i is one of these identical values.

Definition 3 *Given 4 Boolean values a, b, c, d in this order such that 3 of them are identical and the remaining one is different, the position $i \in [1,4]$ of this remaining value is called the intruder position or the intruder for brevity.*

Then, H_i holds iff there is an intruder among the 4 values a, b, c, d and the intruder position is not i. This suggests that H_i should be stable w.r.t. the permutations which do not affect position i. In fact a little bit more can be established:

Property 1 *Apart from I, H_i are the only logical proportions stable w.r.t any permutation which does not affect position i.*

The special case of I stable w.r.t. any permutation has been already proved in [53]. Table 1.1 allows to check that the H_i's are stable w.r.t. the permutations which do not affect position i. Showing that they are the only ones among the 120 logical properties stable w.r.t. these permutations requires a tedious checking procedure that cannot be summarized here.

Property 1 is quite satisfactory and confirms the informal semantics of H_i. At this point, we see that heterogeneous proportions allow to single out a particular position among an ordered list of 4 values. This position targets the value which is definitely not an intruder among the multiset of 4 items. For instance, when H_i is valid, the value in position i is not an intruder. We shall see in the following section that this property can be used to check the oddness of a given item w.r.t. a multiset of elements. This will be useful when using heterogeneous proportions to classification. More importantly, this gives a clear semantics to the conjunction $H_1(a,b,c,d) \wedge H_2(a,b,c,d) \wedge H_3(a,b,c,d)$ above: $H_1(a,b,c,d) \wedge$

$H_2(a,b,c,d) \wedge H_3(a,b,c,d)$ holds iff among the 4 values a, b, c, d, there is an intruder and this intruder is d. This will be the basis of our oddness measure.

In the next subsection, we establish some results about the parity of the number of 1 or 0 in truth tables for heterogeneous proportions, which are contrasted with homogeneous proportions. This leads to a model of oddness of a given value, among a multiset of 4 values.

Parity of the number of 1 or 0 in tables

Since logical proportions are Boolean formulas involving 4 variables, their truth tables have 16 rows, where only 6 lead to 1 (see [53] for a complete investigation). One could ask if any truth table having 6 lines leading to 1 and 10 lines leading to 0 corresponds to a logical proportion. A simple numbering argument shows that this is not the case. On top of that, we can build classes of patterns which cannot be valid for any proportion:

Property 2 *There is no logical proportion that is true for the four elements of the set of valuations* $\{0111, 1011, 1101, 1110\}$. *The same holds for* $\{1000, 0100, 0010, 0001\}$.

Proof: An equivalence between indicators is of the form $l_1 \wedge l_2 \equiv l_3 \wedge l_4$. If this equivalence is valid for $\{0111, 1011\}$, it means that its truth value does not change when we switch the truth value of the 2 first literals from 0 to 1: there are only 2 indicators for a and b satisfying this requirement: $a \wedge b$ and $\overline{a} \wedge \overline{b}$. If this equivalence is still valid for $\{1101, 1110\}$, its truth value does not change when we switch the truth value of the 2 last literals from 0 to 1: there are only 2 indicators for c and d satisfying this requirement: $c \wedge d$ and $\overline{c} \wedge \overline{d}$. Then the equivalence $l_1 \wedge l_2 \equiv l_3 \wedge l_4$ is just $a \wedge b \equiv c \wedge d$, $a \wedge b \equiv \overline{c} \wedge \overline{d}$, $a \wedge b \equiv \overline{c} \wedge \overline{d}$ or $\overline{a} \wedge \overline{b} \equiv \overline{c} \wedge \overline{d}$. None of these equivalences is true for the four elements of the set of valuations $\{0111, 1011, 1101, 1110\}$. The same reasoning is still applicable for the other class. □

Applying a similar reasoning, we can build other set of valuations which cannot make simultaneously true a logical proportion.

Property 3 *A logical proportion cannot be made true by a set of 4 valuations including 3 valuations of one of the classes appearing in Property 2 and where the 4th valuation is just the negation componentwise of the remaining valuation of the class.*

For instance, there is no logical proportion true for $\{0111, 1011, 1101, 0001\}$ or for $\{0111, 0100, 1101, 1110\}$. This remark helps establishing the following result:

Property 4 *Heterogeneous proportions are the only proportions whose all valid patterns have an odd number of 1.*

Proof: From the truth tables, we observe that only valid patterns for heterogeneous proportions have an odd number of 1. Let us now consider a proportion whose 6 valid patterns carry an odd number of 1. As there are exactly 8 patterns with an odd number of 1, and thanks to the previous property, this proportion includes necessarily 3 patterns from each of the previous classes. If the valid patterns in one class are obtained from the valid patterns from the other class just by negating all the variables, the proportion is code independent and then, it is a heterogeneous proportion. In the opposite case, it means that we have at least one pattern in the first class with no negated counterpart in the other class: for instance, $1110, 1101, 1011$ are valid but 0001 is not a valid pattern, leaving only $1000, 0100, 0010$ to complete the truth table of a logical proportion. Then property 3 tells that there is no proportion valid for $1110, 1101, 1011, 1000$. □

A similar property holds for homogeneous proportions:

Property 5 *Homogeneous proportions are the only proportions whose all valid patterns have an even number of 1.*

1.2.2 Specificity of Analogical proportions

As stated before, an analogical proportion is a statement of the form "a is to b as c is to d" (usually denoted $a : b :: c : d$ and where the type of a, b, c, d is not specified for now), expressing informally that "a differs from b as c differs from d" and vice versa. As it is the case for numerical proportions, this statement is supposed to still hold when the pairs (a, b) and (c, d) are exchanged, or when the mean terms b and c are permuted (see [51] for a detailed discussion). When a, b, c, d are numbers, arithmetical (resp. geometrical) numerical proportions assert equality between two differences: $a - b = c - d$ (resp. ratios: $\frac{a}{b} = \frac{c}{d}$). They are at the root of the idea of analogical proportions. In the following subsections, we shall only focus on the case where a, b, c, d are Boolean truth variables, i.e. taking their values in $\mathbb{B} = \{0, 1\}$. In Chapter 2, we will show how analogical proportions can be extended to graded truth values when variables take their values in $[0, 1]$ and we also discuss the case of nominal attributes. This will enable us to deal with all kind of attributes.

Case of Boolean attributes

When considering Boolean variables, a simple way to abstract the symbolic counterpart of numerical proportions has been given in [50] by focusing on indicators that capture the ideas of "similarity" and "dissimilarity". Namely, for a pair (a, b) of Boolean variables, four indicators are associated to such a pair, namely the Boolean functions:

$-$ $a \wedge b$ and $\neg a \wedge \neg b$: they are respectively *positive similarity* and *negative similarity* indicators ; $a \wedge b$ (resp. $\neg a \wedge \neg b$) is true iff only both a and b are true (resp. false);

– $a \wedge \neg b$ and $\neg a \wedge b$: they are *dissimilarity* indicators ; they are true iff only one of a or b is true and the other is false.

As introduced before, a logical proportion is a conjunction of two equivalences between such indicators, and the best "clone" of numerical proportions among logical proportions is the analogical proportion [51], defined as:

$$a : b :: c : d = (a \wedge \neg b \equiv c \wedge \neg d) \wedge (\neg a \wedge b \equiv \neg c \wedge d) \tag{1.2}$$

This logical expression of an analogical proportion, using only dissimilarities, could be informally read as *what is true for a and not for b is exactly what is true for c and not for d, and vice versa*. It perfectly fits with the reading "a differs from b as c differs from d and vice versa". As such, a logical proportion is a Boolean formula involving 4 variables and it can be easily checked on its truth table (Table 1.2) that the logical expression of $a : b :: c : d$ satisfies symmetry ($a : b :: c : d \Rightarrow c : d :: a : b$) and central permutation ($a : b :: c : d \Rightarrow a : c :: b : d$), which are key properties of an analogical proportion, acknowledged for a long time, while $a : b :: a : b$ (and consequently $a : a :: b : b$, thanks to the central permutation property) always hold true. Thus, in terms of generic patterns, we see that analogical proportion always holds for the three following patterns: $s : s :: s : s$, $s : s :: t : t$ and $s : t :: s : t$ where s and t are distinct values. Table 1.2 exhibits the six Boolean patterns (among $2^4 = 16$ possible ones), which make true an analogical proportion. Another remarkable property of analogical proportion that can be observed on Table 1.2 is the independence with respect to the positive or negative encoding of properties, namely we have $a : b :: c : d \Rightarrow \neg a : \neg b :: \neg c : \neg d$. Moreover, with this definition, the analogical proportion is transitive in the following sense:

$$(a : b :: c : d) \wedge (c : d :: e : f) \Rightarrow a : b :: e : f$$

Table 1.2: Valuations where $a : b :: c : d$ is true

a	b	c	d	$a : b :: c : d$
0	0	0	0	1
1	1	1	1	1
0	0	1	1	1
1	1	0	0	1
0	1	0	1	1
1	0	1	0	1

Thanks to properties of Boolean algebra, it can be easily seen that formula (1) is equivalent to:

$$((a \rightarrow b) \equiv (c \rightarrow d)) \wedge ((b \rightarrow a) \equiv (d \rightarrow c)) \tag{1.3}$$

but also, which is less obvious, that (1) is equivalent to:

$$((a \wedge d) \equiv (b \wedge c)) \wedge ((\neg a \wedge \neg d) \equiv (\neg b \wedge \neg c))$$

and thus to

$$((a \wedge d) \equiv (b \wedge c)) \wedge ((a \vee d) \equiv (b \vee c)) \tag{1.4}$$

where there is no negation operator [47]. Formula 1.4 can be viewed as the logical counterpart of a well-known property of geometrical proportions: the product of the means is equal to the product of the extremes.

Note that the last two expressions now refer to similarity indicators. Moreover, the analogical proportion "a is to b as c is to d" now reads "what a and d have in common, b and c have it also (both positively and negatively)", which is a less straightforward reading of the idea of analogy than the one associated with 1.2 or 1.3.

One of the side product of geometrical proportions is the well-known "rule of three" allowing to compute a suitable 4th item $x = d$ in order to complete a proportion $\frac{a}{b} = \frac{c}{x}$. This property has a counterpart in the Boolean case where the problem can be stated as follows. Given a triple (a, b, c) of Boolean values, does it exist a Boolean value x such that $a : b :: c : x = 1$, and in that case, is this value unique? It is easy to see that there are cases where the equation has no solution since the triple a, b, c may take $2^3 = 8$ values, while $a : b :: c : d$ is true only for 6 distinct 4-tuples. Indeed, the equations $1 : 0 :: 0 : x = 1$ and $0 : 1 :: 1 : x = 1$ have no solution. It is easy to prove that the analogical equation $a : b :: c : x = 1$ is solvable iff $(a \equiv b) \vee (a \equiv c)$ holds true. In that case, the *unique* solution is given by $x = a \equiv (b \equiv c)$.

It is worth mentioning that a notion closely related to analogical proportion, but introduced before their Boolean formalization, namely *Analogical Dissimilarity* denoted AD, has been defined in [2]. Roughly speaking, $AD(a, b, c, d)$ is just the number of Boolean values which have to be switched to make $a : b :: c : d$ a valid proportion. For instance, $AD(0, 1, 1, 0) = 2, AD(0, 1, 1, 1) = 1$, and $AD(a, b, c, d) = 0$ iff $a : b :: c : d$ holds. So $AD(a, b, c, d) \in \{0, 1, 2\}$. AD can be extended in a straightforward manner to Boolean vectors as follows: $\Sigma_{i=1}^{n} AD(a_i, b_i, c_i, d_i) \in [0, 2n]$. Then for Boolean vectors, $AD(a, b, c, d) = 0$ iff $a : b :: c : d$ holds.

Extension to Boolean vectors

Representing objects with a single Boolean value is not generally sufficient and we have to consider situations where items are represented by *vectors* of Boolean values, each component being the value of a binary attribute. A simple extension of the previous definitions to Boolean vectors in $\mathbb{B}^n$ of the form $\vec{a} = (a_1, \cdots, a_n)$ can be done as follows:

$$\vec{a} : \vec{b} :: \vec{c} : \vec{d} \text{ iff } \forall i \in [1, n],\ a_i : b_i :: c_i : d_i \tag{1.5}$$

Obviously, all the basic properties (symmetry, central permutation) still hold for vectors. On top of that, the equation solving process is still valid and provides a new insight about analogical proportion: analogical proportions are *creative*. Indeed, let us consider the following example where:

$$\vec{a} = (1, 0, 0)$$

$$\vec{b} = (0, 1, 0)$$

$$\vec{c} = (1, 0, 1)$$

Solving the analogical equation $\vec{a} : \vec{b} :: \vec{c} : \vec{x}$ yields

$$\vec{x} = (0, 1, 1)$$

which is a new vector *different* from $\vec{a}, \vec{b}$ and $\vec{c}$.

We now show the specificity of heterogeneous (and homogeneous) proportions from a reasoning point of view.

1.2.3 Inference and univocal proportions

There is a way to infer unknown properties of a partially known object D starting from the knowledge we have about its other specified properties, and assuming that a logical proportion T holds componentwise with three other objects A, B, C, also represented in terms of the same n Boolean features. This can be done via an induction principle that can be stated as follows (where J is a subset of $[1, n]$, and x_i denotes the truth value of feature i for object $X \in \{A, B, C, D\}$):

$$\frac{\forall i \in [1, n] \setminus J, T(a_i, b_i, c_i, d_i)}{\forall i \in J, T(a_i, b_i, c_i, d_i)}$$

This can be seen as a continuity principle assuming that if it is known that a proportion holds for some attributes, this proportion should still hold for the other attributes. It generalizes the inference principle used with the analogical proportion [63, 51] for prediction and classification purposes. From a strict logical viewpoint, this inference rule is unsound as there is no guarantee that the conclusion holds when the premisses hold. Nevertheless, specially when the ratio $\frac{|J|}{n}$ is close to 1, which means that proportions hold on a large number of attributes, it is natural to consider that such a proportion may also hold on the small number of remaining attributes.

This principle requires the unicity of the solution of equation $T(a, b, c, x) = 1$ where x is unknown, when it exists. Namely, given 3 Boolean values a, b, c, we want to determine for what logical proportion T the equation $T(a, b, c, x) = 1$ is solvable, and in such a case, if the solution is unique.

Definition 4 *If, when the equation $T(a, b, c, x) = 1$ is solvable, the solution is unique, then the proportion T is said to be* 4-univocal. *In a similar manner, one may define proportions that are* 1, 2, *or* 3*-univocal. T is univocal when it is i-univocal for every $i \in [1, 4]$.*

First of all, it is easy to see that there are always cases where the equation $T(a, b, c, x) = 1$ has no solution, whatever the proportion T. Indeed, the triple a, b, c may take $2^3 = 8$ values, while any proportion T is true only for 6 distinct valuations, leaving at least 2 cases with no solution. For instance, when we deal with H_4, the equations $H_4(0, 0, 0, x)$ and $H_4(1, 1, 1, x)$ have no solution.

We have the following result:

Property 6 *The homogeneous and the heterogeneous proportions are the only proportions which are univocal.*

Proof: From the truth tables, we see that the 2 types of proportions satisfy the property. Now, a proportion which is not i-univocal is necessarily valid both for a pattern with an odd number of 1, and for a pattern with an even number of 1. Then, properties 4 and 5 exclude homogeneous and heterogeneous proportions. □

1.3 Research axes

1.3.1 Principal contributions

This habilitation thesis contributes mainly in the two following axes:

- **Research axe 1: Logical proportions-based classification**: This axe includes our principal contributions in this thesis in which we study logical proportions as a basic tool for solving classification problems. We propose two families of classification approaches based on logical proportions and dealing with different types of data for this purpose. We have developed principally two kind of classifiers based on logical proportions and applying the above inference principle: the first family (using homogeneous proportions) proposes two new Analogical proportions-based classifiers denoted AP-Classifiers. The second family, using heterogeneous proportions and applying the inference process in different way, are called $Oddness$-based classifiers.

  1. *Analogical proportions-based classification:* In this context, the three following main contributions have been achieved:
     - Deal with numerical and nominal data: We have extended the use of analogical proportions to handle numerical or nominal data. We propose new definitions in this context.

- AP-Classifier: using *triples* of items, we have build up a valid proportion with the new item to be classified. Based on this principle, we have developed a new analogical proportion-based classification algorithm.
- A rule-based AP-classifier: using *pairs* of items, we first build a set of inference rules (in a preliminary step) to be used for classification purpose.

2. *Oddness/Evenness-based classification:*
   - $Oddness$ and $evenness$ indexes: Based on heterogeneous proportions, we first develop new $Oddness$ and $Evenness$ indexes that measure the extent to which an item may be seen as an intruder in a subset of items first in case of Boolean data. Then these new indexes have been extended to deal with numerical data, vectors, subset of different sizes and also missing data.
   - $Oddness$-Classifier: Based on the oddness index, we have proposed a generic algorithm able to deal with Boolean, nominal or numerical data.
   - Optimized $Oddness$ classifiers: To reduce the cubic complexity of the Oddness classifiers and starting from the observed results, we propose two options to improve $Oddness$ classifiers using pairs rather than triples of items with more constrained pairs:
     * Odd2 using a remote element.
     * Odd2 using two nearest neighbors.
   - $Evenness$ classifier: using the evenness index, we propose a new algorithm dealing with Boolean data.

- **Research axe 2: Analogical proportions-based preference learning**: Observing the success of Analogical Proportions to solve classification problems for a variety of datasets types, has led us to investigate more its ability to solve other kind of problems. In this second axe, we are especially interested to Preference Learning problem and we study the efficiency of Analogical inference principle for making preference prediction. If compared to the classification datasets, preferences are more structured in the sense that they are expected not to exhibit contradictory trade-offs and to be monotone, which has to be taken into account in the learning process.

  In this axe, two basic contributions have been achieved:

  - In the context of Boolean setting, we apply a vertical reading of Analogical Proportions when applying the analogical-based inference principle. We have developed two algorithms for predicting preferences that exploits triples of pairs of preferences picked from a training set. The first algorithm only exploits the given set of examples, the second one completes this set with preferences deducible from this set under a monotony assumption.
  - Next, we extend the previous Analogical preference learning to the multi-valued setting where Analogical Proportions valuations are just a matter of

degree in $[0, 1]$ instead of $\{0, 1\}$. In this context, we apply either a vertical or horizontal reading of proportions and we propose three new algorithms either using triples of pairs of preferences or simply exploiting these preference pairs one by one.

1.3.2 Research collaboration

This habilitation thesis has been achieved through a collaboration and contributions of many researchers from different domains either graduate students or international experts related to the two research axes. The following Table summarizes the degree of involvement of each of these collaborators in each research axe.

Table 1.3: Collaborators and their involvements in the two research axes

Collaborator	**University**	**Axes**
Pr. Henri Prade	University Paul Sabatier, France	1,2
Pr. Gilles Richard	University Paul Sabatier, France	1
Pr. Marc Pirlot	Mons University, Belgium	2
Dr. Mathieu Serrurier	University Paul Sabatier, France	1
Dr. Olivier Sobrie	Mons University , Belgium	2
Toumather Nsibi (Master student)	University of Tunis	2
Marouane Essid (Master student)	University of Tunis	1

1.4 Conclusion

This chapter is devoted to provide the necessary background on logical proportions as a basic tool used to design several approaches in this thesis. Especially, we have explained the specificity of heterogeneous and homogeneous logical proportions. Then, we have shortly introduced a summary on the two basic research axes developed in this thesis: designing new approaches for classification using both analogical and heterogeneous proportions and proposing new algorithms for user preference prediction based on analogical proportions.

In next chapters, a detail regarding each of the previously introduced contributions will be provided.

Chapter 2

Analogy-based classification

2.1 Introduction

Numerical proportions play an important role in our perception and understanding of reality. Indeed proportions are a matter of comparisons expressed by differences or ratios that are equated to other differences or ratios. Two centuries ago, Gergonne [30, 31] was the first to explicitly relate numerical (geometric) proportions to the ideas of interpolation and regression. In fact, geometrical proportions exhibit a simple but effective extrapolation power since, knowing 3 elements a, b, c, we can easily compute a last one d such that $\frac{a}{b} = \frac{c}{d}$ (known as the *rule of three*).

Numerical proportions may be considered as particular instances of the so-called "analogical proportions" which are statements of the form "a is to b as c is to d", often denoted $a : b :: c : d$. This supposes that a, b, c, d refer to the same category of items, which can thus be described in the same terms. Such a proportion expresses that "a differs from b as c differs from d", as well as "b differs from a as d differs from c" [47]. In other words, the pair (a, b) is analogous to the pair (c, d) [33]. More recently, diverse formal views of analogical proportions have been developed in algebraic or logical settings [63, 43, 44], in such a way that the essential properties of numerical proportions still hold, and especially their extrapolation power. For instance, when analogical proportions are defined in terms of subsets of properties that hold true in a given situation, each variable a, b, c and d refers to a situation described by a vector of feature values [39, 46, 47, 50].

Based on such formal approaches, analogical proportions have proved to be a valuable tool in morphological linguistic analysis [64, 38], in solving IQ tests such as Raven progressive matrices [58, 21] as well as in classification tasks where results competitive with the ones of classical machine learning methods have been first obtained by [2, 45]. In this chapter, we focus on this specific type of machine learning application, namely classification and we show how analogical proportions can be useful for building a family of classifiers named: *Analogical Proportions-based classifiers*.

This chapter is organized as follows: In Section 2.2, we first establish the basis for AP-classification. Then, we propose a generic algorithm (AP-classifier) that covers all types of data and looks for triples of items to build up a valid proportion with the new item to be classified. To predict the class for this new item, the algorithm computes an analogical score for each triple in the training set and cumulates these scores for each possible class. In Section 2.3, we show that there exists an alternative method to build analogical proportion based-learners by statically building a set of inference rules during a preliminary training step. This gives birth to a new classification algorithm that deals with pairs rather than with triples of examples.

2.2 AP-Classification

2.2.1 Motivation

The purpose of this section is twofold. First, we make precise the link between the analogical jump that characterizes usual analogical inference and an analogical proportion-based pattern of inference. Second, we provide some examples where the analogical proportion-based inference pattern can be successfully used for classification or interpolation. This will motivate the design of analogical proportion-based classifiers (AP classifiers for short) in the next sub-section.

Analogical jump and analogical proportion

In its simplest form, analogical reasoning, without any reference to the notion of proportion, is usually viewed as a way to infer some new fact on the basis of a single observation. Analogical reasoning has been mainly formalized in the setting of first order logic [23, 41]. A basic pattern for analogical reasoning is then to consider 2 terms s and t, to observe that they share a property P, and knowing that another property Q also holds for s, to infer that it holds for t as well. This is known as the "analogical jump" and can be described with the following simplified inference pattern, leading (possibly) to a wrong conclusion:

$$\frac{P(s) \;\; P(t) \;\; Q(s)}{Q(t)} \quad (AJ)$$

Making such an inference pattern valid would require the implicit hypothesis that P determines Q inasmuch as $\nexists u \; P(u) \wedge \neg Q(u)$. This may be ensured if there exists an underlying functional dependency, or more generally, if it is known for instance that when something is true for an object of a certain type, then it is true for all objects of that type. Otherwise, without such guarantees, the result of an analogical inference may turn to be definitely wrong.

	P	Q	s	t	
$\vec{a}$	1	0	1	0	$P(s)$
$\vec{b}$	1	0	0	1	$P(t)$
$\vec{c}$	0	1	1	0	$Q(s)$
$---$	$---$	$---$	$---$	$---$	$---$
$\vec{d}$	0	1	0	1	$Q(t)$

Table 2.1: A syntactic view of analogical jump

In the Boolean case, to link the above analogical pattern with the concept of analogical proportion, it is tempting to write something like: $P(s) : P(t) :: Q(s) : Q(t)$ since we have 4 terms which obey, at least from a syntactic viewpoint, the structure of an analogical proportion. Indeed, it is sufficient to encode each piece of information in a binary way according to the presence or the absence of P, Q, s, or t in the corresponding term, and we get the encoding d of $Q(t)$ via the equation solving process as in Table 2.1.

In that case, $\vec{a} = P(s), \vec{b} = P(t), \vec{c} = Q(s), \vec{d} = Q(t)$ are encoded as Boolean vectors where the semantics carried by the predicate symbols P and Q is not considered.

In [69, 2], the authors take a similar inspiration where, starting from Boolean datasets and focusing on binary classification problem, they apply the following inference principle:

$$\frac{\vec{a} : \vec{b} :: \vec{c} : \vec{d}}{cl(\vec{a}) : cl(\vec{b}) :: cl(\vec{c}) : cl(\vec{d})} \quad AP^1$$

It means that if 4 Boolean vectors build a valid analogical proportion, then it should be true that their classes build also a valid proportion. Starting from this viewpoint, in the case where $\vec{a}, \vec{b}, \vec{c}$ are in a sample set, i.e., their classes are known, and $\vec{d}$ being the object to be classified, if the equation $cl(\vec{a}) : cl(\vec{b}) :: cl(\vec{c}) : x = 1$ is solvable (in that case, we say that the triple $(\vec{a}, \vec{b}, \vec{c})$ is *class solvable*), they allocate its solution to $cl(\vec{d})$ just by applying the previous principle. Their experience highlight the predictive power of this principle. In [15] has been shown that AP principle is just a particular instance of (AJ).

In the pattern AP, we transfer the identity of differences pertaining to pairs $(\vec{a}, \vec{b})$ and $(\vec{c}, \vec{d})$ to the relation between their classes. This enables us to predict the missing information about $\vec{d}$, using AP as an extrapolation principle. This is obviously a form of reasoning that is not sound, but which may be useful for trying to guess unknown values.

[1] From now on, we denote AP-classifier, any classifier derived from the AP rule

Some arguments in favor of the study of AP-classifiers

The previous subsection has shown that the AP-inference just corresponds to the usual view of analogical inference, and as such, its conclusion are brittle. Still, we now provide some arguments or clues showing that AP may make sense for classification.

Some experimental evidences First of all, it should be reminded that the authors of [2, 45] have been the first to design classifiers for Boolean data based on an AP principle. For this purpose, they define what they call the *Analogical Dissimilarity AD* over Boolean values: this can be viewed as a way to measure how far is a Boolean proportion from being a valid analogical proportion, since $AD(\vec{a}, \vec{b}, \vec{c}, \vec{d}) = 0$ iff $\vec{a} : \vec{b} :: \vec{c} : \vec{d}$ holds, as recalled in Chapter 1. Then adding AD component-wise allows the authors to extend AD to Boolean vectors, leading to a flexible implementation of AP principle by minimizing the cumulated AD. On a set of Boolean UCI benchmarks, this implementation has led to very good quality results in terms of accuracy and outperforms standard classifiers on some benchmarks such as `Monk2` (`Monk2` is a two-classes dataset defined by the constraint that the sum of the components of a nominal data vector is equal to 2, before binarisation takes place). This work has provided the first empirical validation of the AP principle applied to classification.

Analogical classification and exclusive or In a two-classes classification problem with Boolean data, the class of a given vector is a function f of its components. If we deal with vectors in $\mathbb{B}^n$, we have exactly 2^{2^n} such functions. It is worth pointing out that the AP principle always leads to a correct answer in case of some particular functions such as constants, projections, and the exclusive or (see [15] for details).

The fact that AP always yields a good result for the exclusive or has been first observed experimentally [20, 34]. A similar, simpler reasoning indicate that the application of AP cannot lead to a wrong result for projections. In fact, it has been recently formally proved [22] that the AP principle is sound as soon as the labeling function is an affine Boolean function (this means in practice that the function is a constant, a projection, xor function or $\equiv$ function over some subsets of the n attributes). Moreover it can be shown that there is no other Boolean function for which this is true [22]. These results make clear that what we have explicitly shown for a two-dimensional XOR is valid for any dimension [15].

An example with a non affine Boolean function As soon as the labeling function is not a Boolean affine function, we are sure there exist some triples that will induce wrong prediction. Nevertheless, it is still possible to get good results with the AP principle for non affine functions. Indeed, it is sufficient that in most situations, the candidate triples majoritarily yield correct answers. We now provide an example of such function f (where $\oplus$ denotes the xor operator):

$f(x_1, \ldots, x_n) = x_1 \oplus x_2$ if $\Sigma_{i=1}^{n} x_i < n - 1$

and $f(x_1, \ldots, x_n) = min(x_1, \ldots, x_n)$ otherwise.

With $n = 7$, the first condition covers 120 inputs while the second one covers only 8 inputs. It can be computed that the number of non trivial triples (i.e. where $a \neq b$ or $a \neq c$) for which analogy is solvable componentwise is 2.064.512, among which only 187.620 are class solvable. Among these 187.620 triples $(\vec{a}, \vec{b}, \vec{c})$, 160.224 are such that the solution of the class equation $f(\vec{a}) : f(\vec{b}) :: f(\vec{c}) : x = 1$ is exactly $f(\vec{d})$, where $\vec{d}$ is such that $\vec{a} : \vec{b} :: \vec{c} : \vec{d}$ holds componentwise and only a small minority (27.396) leads to the wrong label. As expected, when running AP classifier on this artificial dataset, we get an average accuracy of 97%, while k-NN yields only 93%. This illustrates the fact that good performance are still possible for functions which are not affine strictly speaking. A phenomenon of this type may take place in the case of Monk2 dataset of the UCI repository where excellent results were reported for AP-classifier as recalled above.

Analogical proportion and interpolation Analogy-based inference applies to any type of data. The arithmetical proportion view of analogy (already mentioned in Chapter 1) applies to real numbers and can be easily extended componentwise to real-valued vector spaces. In that case, $\vec{a} : \vec{b} :: \vec{c} : \vec{d}$ simply means that the 4 items are the vertices of a parallelogram. The existence of such a parallelogram in a linguistic dataset have been early pointed out in [60].

Besides, the relation between analogical proportion and interpolation, in the numerical case, is also worth mentioning. Indeed, let us consider the multiple-valued modeling of the analogical proportion (that will be defined in the next sub-section). It has been shown that an analogical proportion agrees with the idea of linear interpolation and extrapolation, since, in particular, the solution of $a : x :: x : b = 1$ is $x = \frac{a+b}{2}$ in the numerical case. More generally, the following result proved and discussed in [25], is noticeable:

Proposition : *if $\vec{a}$, $\vec{d}$ are the coordinates of points in a n-dimensional space, i.e. $\vec{a} = (a_1, \cdots, a_n)$, $\vec{d} = (d_1, \cdots, d_n)$, the solution of the analogical proportion equation $\vec{a} : \vec{x} ::_A \vec{x} : \vec{d} = 1$ is the midpoint $\vec{x} = ((a_1 + d_1)/2, \cdots, (a_n + d_n)/2)$ of the segment $\vec{a}\vec{d}$ in the n-dimensional space.*

Indeed for each component, the equation reads : $x_i - a_i = d_i - x_i$, which yields $x_i = (a_i + d_i)/2$. The solution of equation $\vec{a} : \vec{x} ::_A \vec{x} : \vec{d} = 1$ is thus obtained by a linear interpolation. This points out that AP, when applied to rescaled numerical data, performs a linear extrapolation.

In the next section, we discuss different methodologies for the implementation of AP-classifiers. First, we present a brute force method then we propose an optimized algorithm that reduces the search space of triples.

2.2.2 Contribution

In this Section, we aim to show how analogical proportions may be useful for building analogical-based classifiers that will be described at the end of this section. To be able to deal with all kind of attributes (Boolean, nominal or numerical), we first provide necessarily definitions and extensions of analogical proportion, defined for Boolean case in Chapter 1, to graded truth values when variables take their values in $[0, 1]$. The we will also discuss the case of nominal attributes.

Extension to Real-valued attributes

When dealing with a real-valued attribute with domains $[\underline{m}, \overline{m}]$, it is always possible to rescale these attributes such as the new value x_{new} associated to a current value x is computed as follows:

$$x_{new} = \frac{x - \underline{m}}{\overline{m} - \underline{m}}$$

and as such, can be viewed as a truth value. If we consider the Boolean expression of the analogical proportion given by formula (1), one may think of many possible multiple-valued extensions, depending on the choice of the connectives associated to $\neg$, $\wedge$ and $\equiv$. Then, it is important to make proper choices that are in agreement with the intended meaning of analogical proportion. Some properties seem very natural to preserve, such as

- i) the independence with respect to the positive or negative encoding of properties (one may describe a price as the extent to which it is cheap, as well as it is not cheap), which leads to require that $\neg a : \neg b :: \neg c : \neg d$ holds if $a : b :: c : d$ holds (as it is already the case with Boolean truth values);
- ii) the definition should agree with the Boolean definitions in the limit case where a, b, c, d take their values in $\{0, 1\}$.

A careful analysis of these requirements, ensuring that the value of b can be retrieved from the value of a and the ones of $a \wedge \neg b$ and $\neg a \wedge b$, leads to choose (for the sake of simplicity, we keep the same notation for a variable and its truth value) [56, 25, 52]:

- negation: $\neg a = 1 - a$;
- Łukasiewicz implication: $a \rightarrow b = \min(1, 1 - a + b)$ (or equivalently $a \wedge \neg b = \max(0, a - b)$);
- conjunction: $a \wedge b = \min(a, b)$;
- equivalence: $a \equiv b = \min(a \rightarrow b, b \rightarrow a) = 1 - |a - b|$.

When starting from formula (2), this leads to the following expression which both generalizes the Boolean case to multiple-valued entries and introduces a graded view of the analogical proportion:

$$a : b :: c : d = 1 - |(a - b) - (c - d)| \text{ if } a \geq b \text{ and } c \geq d, \text{ or } a \leq b \text{ and } c \leq d$$

$$a : b :: c : d = 1 - \max(|a - b|, |c - d|) \text{ otherwise} \quad (A)$$

This is appropriate for a numerical setting, and clearly agrees with the difference-based semantics of analogical proportions. As can be seen under definition (A), $a : b :: c : d = 1$ if and only if $(a - b) = (c - d)$. We thus recognize the arithmetical proportion.[2]

When starting from definition (3), involving the operator $\vee$ defined as

$$a \vee b = \neg(\neg a \wedge \neg b) = 1 - \min(1 - a, 1 - b) = \max(a, b)$$

we get another expression:

$$a : b :: c : d = \min(1 - |\min(a, d) - \min(b, c)|, 1 - |\max(a, d) - \max(b, c)|)$$

which can be rewritten as:

$$a : b :: c : d = 1 - \max(|\min(a, d) - \min(b, c)|, |\max(a, d) - \max(b, c)|) \quad (A^*)$$

It can be checked that $a : b :: c : d = (1 - a) : (1 - b) :: (1 - c) : (1 - d)$ for both (A) and (A^*). Unfortunately, the equivalence between the 2 definitions A and A^* is no longer valid in the multiple-valued case. Roughly speaking, for A^* to hold (i.e., to have value 1), the pair (a, d) and the pair (b, c) should have the same $\min$ and $\max$, so $A^*(0, 0.5, 0.5, 1)$ has the value 0.5 and then does not hold at degree 1. Indeed under definition (A^*), $a : b :: c : d = 1$ if and only if $a = b$ and $c = d$, or if $a = c$ and $b = d$. But applying the definition of A shows that $A(0, 0.5, 0.5, 1) = 1$. Note that, $A^*(a : x :: x : b) = 1$ has no solution if $a \neq b$.

Nevertheless, when a, b, c, d are restricted to $\{0, 1\}$, the 2 definitions A and A^* coincide with definition (1) for the Boolean case, which highlights the agreement between these extensions and the original idea of analogical proportion.

In fact, A^* provides a more restrictive view of analogical proportion than A: For instance, in the case of a tri-valued semantics where the domain of the variables is reduced to

[2]The geometrical proportion could be retrieved as well, choosing Goguen implication $s \to t = \min(1, t/s)$ and $s \to t = 1$ if $s = 0$, product conjunction $s \wedge t = s \cdot t$, and equivalence $s \equiv t = \min(s/t, t/s)$ [47]. It yields

$$a : b :: c : d = \min\left(\frac{\min(1, \frac{b}{a})}{\min(1, \frac{d}{c})}, \frac{\min(1, \frac{d}{c})}{\min(1, \frac{b}{a})}\right) . \min\left(\frac{\min(1, \frac{a}{b})}{\min(1, \frac{c}{d})}, \frac{\min(1, \frac{c}{d})}{\min(1, \frac{a}{b})}\right).$$

with the convention 0/0 = 1. Clearly, $a : b :: c : d = 1$ if and only if $a/b = c/d$. In spite of this nice property, it can be checked that $1/2 : 0 :: 1 : 0 = 1$, which is not satisfactory, since we expect here a value less than 1 [56]. For this reason, we do not consider this option further in this paper.

$\{0, 0.5, 1\}$, the truth table of A has 19 lines leading to 1 among $3^4 = 81$ possible patterns, but A^* has only 15 such lines [52].

Moreover, as can be checked from its definition, that on $[0, 1]$, $A^*(a, b, c, d) = 1$ *only* for the 3 patterns $(a, b, c, d) = (s, s, s, s)$ or (s, t, s, t), or (s, s, t, t). This contrasts with $A(a, b, c, d)$ which is equal to 1 as soon as $a - b = c - d$.

Lastly, we can extend the notion of "valid" proportion to vectors in $[0, 1]^n$ as in the Boolean model with (where P denotes A or A^*):

$$P(\vec{a}, \vec{b}, \vec{c}, \vec{d}) = 1 \text{ iff } \forall i \in [1, n],\ P(a_i, b_i, c_i, d_i) = 1 \tag{2.1}$$

In the frequent case where $P(a_i, b_i, c_i, d_i) \in]0, 1[$ for some index i, $P(\vec{a}, \vec{b}, \vec{c}, \vec{d})$ is not defined, and we have different options to allocate a truth value to the whole proportion. Obviously, a standard option is to compute their mean value $\frac{\Sigma_{i=1}^{n} P(a_i,b_i,c_i,d_i)}{n}$. Let us observe that $\frac{\Sigma_{i=1}^{n} P(a_i,b_i,c_i,d_i)}{n} = 1$ if and only if the analogical proportion holds perfectly on every component. We will use this definition to design AP-classifiers.

Inference in the numerical Case In the numerical case, we consider 4 real-valued vectors $\vec{a}, \vec{b}, \vec{c}, \vec{d}$ over $[0, 1]^n$ (the numerical values are previously rescaled in the unit interval as explained before, but still with discrete classes). Then, we interpret the value of an attribute as a truth value (corresponding to the extent to which the property underlying the attribute holds): thus the value of $P(a_i, b_i, c_i, d_i)$ (where P refers to A or A^*) can always be computed using either the extension A or the extension A^*, and belongs also to $[0, 1]$. So the following inference principle strictly clones the Boolean case :

$$\frac{P(\vec{a}, \vec{b}, \vec{c}, \vec{d}) = 1}{cl(\vec{a}) : cl(\vec{b}) :: cl(\vec{c}) : cl(\vec{d})}$$

Actually, the resulting truth value $P(\vec{a}, \vec{b}, \vec{c}, \vec{d})$ is rarely equal to 1 but can be close to 1. So, as we shall see, has to be adapted for a proper implementation. The idea is to compute a kind of truth value attached to $\vec{a} : \vec{b} :: \vec{c} : \vec{d}$ as explained above (using min or a mean), and then to associate this value to the statement that $cl(\vec{a}) : cl(\vec{b}) :: cl(\vec{c}) : cl(\vec{d})$ would hold. We indicate in the following how we take advantage of this value in the algorithms.

Extension to nominal attributes

When it comes to nominal attributes whose domains are finite sets of values without a particular meaningful ordering, both definitions A and A^* no longer applies since they involve $\min$ and $\max$ operators. However, it can be noticed that the 3 patterns making $A^*(a, b, c, d) = 1$ may be regarded as the universal patterns of analogy whatever the domain $\mathcal{V}$ to which a and b belong. This leads to a natural definition of analogical proportion for nominal values as follows:

$$a : b :: c : d = 1 \text{ iff } (a,b,c,d) \in \{(s,s,s,s),(s,t,s,t),(s,s,t,t) \| \ s,t \in \mathcal{V}\} \\ \text{and } a : b :: c : d = 0 \text{ otherwise} \tag{2.2}$$

For instance, for an attribute such as *color* whose values belong to

$$\mathcal{V} = \{red, green, blue, yellow\}$$

we have $red : green :: red : green = 1$, but $red : yellow : red : green = 0$. Note that, in the above definition, there is no longer intermediary value as given by A^* when $a, b, c, d \in [0, 1]$[3].

It is quite common, in the case of nominal attributes, to binarize their values. This case can be handled as easily as the binary case. Indeed, consider an attribute i whose values belong to domain $\mathcal{V} = \{v_1, \cdots, v_m\}$, it can be straightforwardly binarized by means of m attributes $\alpha_j, j \in [1, m]$ with the following semantics:

$$\alpha_j = 1 \text{ iff attribute } i \text{ has value } v_j \text{ and 0 otherwise}$$

This amounts to replace a nominal attribute taking m values by the m Boolean attributes α_j and to have a function bin defined as:

$$bin : \mathcal{V} \to \mathbb{B}^m \text{ with } bin(v_j) = (0, \ldots, 0, 1, 0, \ldots, 0) \text{ where 1 is in position j}$$

As an illustration, let us consider the previous *color* attribute: it is binarized via a bin function as follows:

$$bin(red) = 1000, bin(green) = 0100, bin(blue) = 0010, bin(yellow) = 0001$$

Let us now consider 4 vectors a, b, c, d such that attribute i is nominal, taking its values in $\mathcal{V} = \{v_1, \cdots, v_m\}$. Moreover, assume without loss of generality that:

$$a_i = v_1, b_i = v_2, c_i = v_1, d_i = v_2$$

Since $bin(v_1) = (1, 0, \ldots, 0)$ and $bin(v_2) = (0, 1, 0, \ldots, 0)$, the situation is pictured in Table 2.2 .

As can be seen on this table, an analogical proportion holds true between the four objects for each binary attribute α_j. This is clearly equivalent to the compact encoding in terms of nominal values namely here: $v_1 : v_2 :: v_1 : v_2$. This expresses the following property:

$$\vec{a} : \vec{b} :: \vec{c} : \vec{d} \text{ (with def. 2.2)} \quad \text{iff} \quad bin(\vec{a}) : bin(\vec{b}) :: bin(\vec{c}) : bin(\vec{d}) \text{ (with def. 1.5)}$$

[3]Note that in the nominal case, $a : b :: c : d = 0$ iff $(a,b,c,d) \in \{(s,t,t,s),(s,s,s,t),(s,s,t,s),(s,t,s,s),(t,s,s,s) \| \ s,t \in \mathcal{V}\}$, while $A^*(a,b,c,d) = 0$ only for the patterns of the form $(1, 0, s, 1)$ or $(0, 1, s, 0)$, $\forall s \in [0, 1]$ (and the other patterns obtained from these two by symmetry and central permutation) [25]. As expected, when $s, t \in \{0, 1\}$, the two sets of patterns coincide.

	$\ldots$	α_1	α_2	α_3	$\cdots$	α_m	$\ldots$	
$\vec{a}$	$\ldots$	1	0	0	$\cdots$	0	$\ldots$	v_1
$\vec{b}$	$\ldots$	0	1	0	$\cdots$	0	$\ldots$	v_2
$\vec{c}$	$\ldots$	1	0	0	$\cdots$	0	$\ldots$	v_1
$\vec{d}$	$\ldots$	0	1	0	$\cdots$	0	$\ldots$	v_2

Table 2.2: Handling non binary attributes

This means that binarizing does not change the status of a proportion. Thus it is not necessary to binarize a nominal attribute to evaluate if an analogical proportion holds.

In many applications and datasets, nominal features may be somehow mid-way between nominal and numerical data, inducing a notion of ordering and/or a notion of proximity. A good instance of this is coming from systems where the user has to grade items with assessments of the type 1,2,3,4,5 stars. This problem should be addressed here by using the A^* definition, rather than the A definition, since A^* allows to keep track of such a graded assessment in the same way as the handling of nominal values.

So given 4 hybrid vectors involving binary, nominal and rescaled real-valued attributes, we have two options to check if together they make an analogical proportion componentwise:

1. Either we use A^* (covering both Boolean and numerical attributes) and its nominal counterpart for nominal attributes (formula 2.2), which provides a rather homogeneous treatment (since A^* returns '1' in the same situations as formula 2.2 does),
2. Or if we want to acknowledge that the distance between real-values is meaningful, we use A, keeping formula 2.2 for nominal attributes.

Analogical Proportion-based classifiers

In the context of classification, we assume that objects or situations A, B, C, D are represented by vectors of attribute values, denoted $\vec{a}, \vec{b}, \vec{c}, \vec{d}$. Applying the AP principle, we propose a new analogical proportion-based approach to classification. This approach relies on the idea that the unknown class $x = cl(\vec{d})$ of a new instance $\vec{d}$ may be predicted as the solution of an equation expressing that the analogical proportion $cl(\vec{a}) : cl(\vec{b}) :: cl(\vec{c}) : x$ holds between the classes. This is done on the basis of triples $(\vec{a}, \vec{b}, \vec{c})$ of examples of the sample set that are such that the analogical proportion $\vec{a} : \vec{b} :: \vec{c} : \vec{d}$ holds componentwise for all or for a large number of the attributes describing the items.

A simple option is to consider all triples $(\vec{a}, \vec{b}, \vec{c})$ in $Solv(\mathcal{S}^3)$, then, for each of them,

to compute a truth value $P(\vec{a}, \vec{b}, \vec{c}, \vec{d})$ as the average of the truth values obtained in a componentwise manner as defined in the previous sub-section. Then we have to aggregate these truth values in order to provide a label for the new item $\vec{d}$ to be classified. This brute force method computing all the suitable triples is cubic and is prohibitive in terms of runtime execution. Obviously, we may think of an offline pre-processing step to make easier the retrieval of these triples (since all triples should be class solvable), as discussed in [8]. We may also think of significantly reducing the number of triples to be considered. This is what we propose in the following.

Instead of systematically surveying $Solv(\mathcal{S}^3)$, we restrict the search for suitable triples $(\vec{a}, \vec{b}, \vec{c})$ by constraining $\vec{c}$ to be one of the k nearest neighbors of $\vec{d}$ w.r.t. Manhattan distance, for some chosen k. In that case, we apply the AP inference principle but restricted to the subset of $Solv(\mathcal{S}^3)$ where $\vec{c}$ is one of the nearest neighbours of $\vec{d}$.

Past experiments [8],[10] have shown that this is not harmful for the results justifying to follow this option which decrease the complexity of the algorithm (instead of being cubic, the algorithm become quadratic). Moreover, the experiments reported in Section 2.2.3 show that the choice of $\vec{c}$ among nearest neighbours does not prevent to get competitive results.

More formally, for each candidate label l, we compute the sum of the truth values $P(\vec{a}, \vec{b}, \vec{c}, \vec{d})$ for all triples $(\vec{a}, \vec{b}, \vec{c})$ whose solution of the corresponding class equation $cl(\vec{a}) : cl(\vec{b}) : cl(\vec{c}) : x = 1$ is l and where $\vec{c} \in N_k(\vec{d})$ the set of the k nearest neighbours of $\vec{d}$. This sum can be viewed as the total credit for the label l. More formally, if we denote $Solv(\mathcal{S}^3)(l)$ the set of triples in $Solv(\mathcal{S}^3)$ such that the solution of the class equation is l, the total credit associated to l is just:

$$SumP(l) = \sum_{\vec{c} \in N_k(\vec{d}),\ (\vec{a},\vec{b},\vec{c}) \in Solv(\mathcal{S}^3)(l)} P(\vec{a}, \vec{b}, \vec{c}, \vec{d})$$

We then allocate to $\vec{d}$ the label l having the highest credit.

$$label(l) = argmax_l(SumP(l))$$

In case of tie, we implement a majority vote. Let us note that, in the case of Boolean or nominal attributes, applying blindly this rule will lead to consider triples where analogical proportions may hold only on a strict subset of $\{1, \ldots, n\}$: for such a triple $P(\vec{a}, \vec{b}, \vec{c}, \vec{d}) = \frac{j}{n} < 1$ where j is the number of components where analogical proportion holds. In practice, in the Boolean case, there are many triples whose $P(\vec{a}, \vec{b}, \vec{c}, \vec{d}) = 1$ and adding values less than 1 does not bring any improvement. This is why, for Boolean and nominal datasets, we further restrict the sum to those triples where $P(\vec{a}, \vec{b}, \vec{c}, \vec{d}) = 1$. In case where no triple (a,b,c) such that P(a,b,c,d)=1 can be found, we would select the triple(s) for which P(a,b,c,d) is maximal.

This leads to Algorithm 1. Up to the previous adjustment, this algorithm handles all types of data: Boolean, nominal and numerical data.

It is quite clear that our procedure relies on the existence of class solvable triples in the training set, which make an analogical proportion with the new item (at least to some extent). In the opposite case, no classification can be suggested and the algorithm returns "unclassified". This might happen in particular when the training set is very small. However, this situation did not occur with the UCI benchmarks we consider. When there are several candidate labels i.e. $unique(maxi, SumP(l))$ is not valid, we implement a majority vote among the candidate labels. This situation, very unlikely with numerical features, may happen with Boolean or nominal data.

Algorithm 1 AP-classifier

Input: $k > 1, \vec{d} \notin \mathcal{S}$ a new instance to be classified
For each label l **do** $SumP(l) = 0$ **end For**
for each $\vec{c}$ in $N_k(\vec{d})$ **do**
  for each pair $(\vec{a}, \vec{b})$ in $|\mathcal{S}|^2$ **do**
    if $cl(\vec{a}) : cl(\vec{b}) :: cl(\vec{c}) : x$ has solution l **then**
      $SumP(l) += P(\vec{a}, \vec{b}, \vec{c}, \vec{d})$
    end if
  end for
end for
$maxi = max\{SumP(l)\}$
if $(maxi \neq 0)$ **then**
  if $(unique(maxi, SumP(l)))$ **then**
    $cl(\vec{d}) = argmax_l\{SumP(l)\}$
  else
    Majority vote
  end if
else
  UNCLASSIFIED
end if
return $cl(\vec{d})$

2.2.3 Experimental validation

We provide experimental results for the AP-classifier and we compare the results to other ML classifiers. This experimental study is based on several datasets taken from the U.C.I. machine learning repository [42]. A brief description of these data sets is given in Table 2.3. Since we have chosen to deal with both nominal or numerical features in this study, the first part in this Table 2.3 includes datasets with categorical or Boolean attribute values and the second part includes datasets with only numerical attributes. In order to apply multiple-valued semantics framework, all numerical attributes are rescaled. More precisely, we just

Table 2.3: Description of datasets

Datasets	Instances	Nominal Att.	Binary Att.	Numerical Att.	Classes
Balance	625	4	20	-	3
Car	743	7	21	-	4
TicTacToe	521	9	-	-	2
Monk1	432	6	15	-	2
Monk2	432	6	15	-	2
Monk3	432	6	15	-	2
Breast Cancer	286	9	-	-	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
Ecoli	336	-	-	7	8
Letter	1076	-	-	16	26

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 dataset.

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, we have to tune the parameters of the AP-classifier as well as ones of the classical classifiers (with which we compare our approach) before performing this cross-validation.

In order to do that, we *randomly* choose a fold (as recommended by [68]), we keep only the corresponding training set (i.e. which represents 90% of the full dataset). 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 dataset. To be sure that our results are stable enough, we run each algorithm (with the previous procedure) 5 times so we have 5 different parameter optimizations. The displayed β is the average value over the 5 different values (one for each run). The results shown in Table 2.4 are the average values obtained from 5 rounds of this complete process.

Results of AP-classifier

In Table 2.4, we provide mean accuracies and standard deviations for Algorithm 1 in case of nominal or numerical data. We also include testing results for other ML classifiers for comparison. Each value given in this table is the average of 5 different runs.

For AP-classifier, we run our tests both for A and A^* definitions of the graded analogical proportion while Formula 2.2 is applied in case of nominal datasets. For each classifier, the displayed results correspond to the optimal value of the parameter β (for AP-classifier and k-NN, β being the number of nearest neighbors) obtained as a result of the optimization step in the inner cross validation.

To compare our results to those of the brute force algorithm that considers all possible triples $(\vec{a}, \vec{b}, \vec{c})$ in $Solv(\mathcal{S}^3)$ when computing the sum of truth values $P(\vec{a}, \vec{b}, \vec{c})$, we also tested the efficiency of this algorithm on some datasets. We get the following accuracies for the three tested datasets: "Iris" : 89.0 ± 8.62, "Heart": 66.01 ± 4.96 and "Wine": 39.16 ± 5.71.

Let us summarize and analyze the main facts that can be observed in Table 2.4:

1. If we compare the results of the three datasets given above in case of the brute force algorithm to the results of AP-classifier given in Table 2.4, we note that the later largely outperforms the brute force algorithm. This validates the use of a particular set of triples, i.e. those where one of the items is among the k nearest neighbors of the item to be classified. This means that there is no need to use all possible triples in the dataset for classification.

2. Overall, AP-classifier exhibits good classification accuracies and is able to deal with any type of data: Boolean, nominal or numerical.

3. The best accuracies are achieved for large values of parameter k for most of the nominal datasets. In the numerical case and for half of the datasets, good accuracies are obtained using only one nearest neighbor ($k = 1$).

4. AP-classifier seems to perform little better (if compared to the state of the art classifiers) in the nominal case than in the numerical case. Remember that, in the nominal case, only triples with $P(a, b, c, d) = 1$ are used in the summation, i.e: those having analogical proportion holding on all components. So we may say that, using a selected set of triples for classification, can improve the efficiency of the classifier rather than blindly use all possible triples.

5. For numerical data, the proposed algorithm performs slightly better when we use definition A for most datasets. It may be the case that A^*, providing a more restrictive view of analogical proportion than A, removes too many candidates from the potential voting triples. It remains to investigate if there is a way to qualify a target

dataset as being more suitable for classification using A or using A^*. This is an open problem.

6. For nominal data, if we compare results of AP-classifier to those shown in [8], it is clear that the two classifiers perform in the same way for the first 6 nominal datasets.

7. To investigate more this comparison, we also tested the previous algorithm [8], suitable for nominal data only, on the data sets "Breast Cancer", "Voting" and "Hayes-Roth". We observe that AP-classifier is significantly better than the algorithm in [8]. In fact, when we have many nominal attributes with a large number of values, the condition $Dis(a, b) = Dis(c, d)$ in [8] is obviously more difficult to satisfy, since it amounts to require the analogical proportion to perfectly hold on all attributes. Otherwise the corresponding triple will be discarded. On the other hand, in AP-classifier, in case no triples with $P(a, b, c, d) = 1$ could be found, we select the best triple (with highest P) and this allows for a relaxation of the requirement that the analogical proportion should hold on any attribute: in fact, AP-classifier benefits from the summation of elementary evaluations of analogical proportion on each attribute. This explains why AP-classifier performs significantly better than that in [8] for this kind of nominal data set.

Table 2.4: Results for AP-classifier and other ML classifiers obtained with the best parameter β

Datasets	**Algo1**		**KNN**		**C4.5**		**JRIP**		**SVM(RBFKernel)**		**SVM(PolyKernel)**	
	Accuracy	β	*Accuracy*	β	*Accuracy*	β	*Accuracy*	β	*Accuracy*	β	*Accuracy*	β
Balance	86.35±2.27	11	84.05±2.6	11	63.79±4.33	0.3	72.74±3.48	6	99,17±0,34	(32768,0.00195)	98,53±0,26	(128,1)
Car	94.16±4.11	11	92.38±2.51	1	90.84±3.61	0.5	86.58±3.67	8	99,37 ±0,12	(32768,0.03125)	99,19±0,19	(32768,2)
TicTac.	100	3	99.23±1.23	1	88.48±3.72	0.3	96.78±4.14	8	100	(32768,0.03125)	100	(32768,1)
Monk1	99.77±0.71	7	98.37±2.78	2	99.36±0.64	0.4	90.99±13.15	2	100	(32768,0.5)	100	(32768,6)
Monk2	99.77±0.7	11	65.29±1.74	11	67.13±0.61	0.1	64.64±3.69	4	100	(32768,0.03125)	100	(32768,2)
Monk3	99.63±0.7	9	99.14±1.49	1	99.82±0.18	0.2	98.95±1.48	2	100	(32768,0.5)	100	(32768,7)
Breast Cancer	73.68±6.36	10	72.81±7.65	9	71.58±6.55	0.2	70.11±8.59	2	72,86±1,15	(8192,0.00195)	74,19±1,20	(0.03125,2)
Voting	94.73±3.72	7	92.5±3.59	4	96.38±2.63	0.2	95.84±2.39	4	96,37±0,10	(32, 0.03125)	95,72±0,21	(0.03125,2)
Hayes-Roth	79.29±9.3	7	61.41±10.31	3	68.2±6.66	0.2	83.26±9.04	4	79,70±1,55	(32768,0.0078)	79,85±2,05	(32,1)
	A	**A***										
	Accuracy β	*Accuracy* β	*Accuracy*	β	*Accuracy*	β	*Accuracy*	β	*Accuracy*	β	*Accuracy*	β
Diabetes	73.28±4.18 11	73.13±4.66 9	73.42±3.65	11	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	97.01±3.35 4	96.87±1.51 4	96.7±1.73	3	94.79±3.19	0.2	95.87±2.9	4	96,74±0,12	(2,2)	96,92 ±0,23	(2,1)
Heart	81.9±7.64 10	81.75±3.31 9	82.23±10.1	11	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)
Ionosphere	90.55±4.05 1	90.41±3.47 1	90.8±3.39	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	94.89±6.4 5	94.55±5.44 7	94.88±6.4	3	94.28±5.19	0.2	93.65±5.24	6	94,13±1,28	(32768,0.5)	96,13±0,99	(512,1)
Wine	98.12±3.94 9	97.55±3.72 9	97.75±3.91	7	94.23±5.54	0.1	94.99±3.49	8	98,20±0,47	(32768,2)	98,53±0,75	(2,1)
Sat. Image	94.96±1.87 1	94.75±1.41 1	94.89±2.77	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)
Glass	72.99±7.82 1	73.23±7.44 1	73.04±12.07	1	69.92±7.4	0.2	69.06±6.28	5	68,50±1,57	(2,8)	73,01±1,50	(2048,2)
Ecoli	86.32±5.59 7	86.78±5.25 9	85.37±6.58	5	82.6±6.39	0.2	81.56±6.04	5	87,50±0,30	(2,8)	87,50±0,30	(8,1)
Segment	96.84±0.78 1	96.84±1.27 3	96.76±1.11	1	95.77±1.77	0.2	94.55±1.96	6	96,98± 0,25	(2048,0.125)	97,14±0,17	(8,4)
Letter	76.29±3.51 1	75.83±3.16 1	75.93±2.35	1	63.38±4.04	0.2	62.6±5.42	9	83,59 ±0,55	(32768,0.5)	82,93±0,54	(0.5,3)
Average	89.52	89,45	86.36		83.79		84.35		91.05		91.25	

Feature selection In order to study the sensitivity of analogical classifiers to irrelevant features, we apply feature selection to some datasets having high number of features and reduced number of instances. For this purpose, we use the feature selection option:

"CfsSubsetEval" provided by Weka that evaluates a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. In Table 2.5, we save results of AP-classifier applied to these datasets after selecting the most relevant features.

Table 2.5: Accuracies given as mean and standard deviation for AP-classifier applied to datasets after feature selection

Datasets	Nbr Features	Relevant Features	Using A	Using A*
Heart	13	7	83.13±8.02	83.27±7.42
Ionosphere	34	12	92.78±3.8	92.65±3.56
Wine	13	11	98.01±3.81	98.92±2.17
Sat.Image	36	11	95.07±1.62	95.43±1.92
Segment	19	6	96.2±1.19	95.6±1.74

If we compare results in Table 2.5 after applying feature selection with those given in Table 2.4 before feature selection, we observe (as expected) that the AP-classifier is more efficient when applied to the dataset with only selected features, especially for definition A^* and for the two first datasets. For other datasets: "Wine" (in case of definition A), "Sat.Image" and "Segment", we cannot see a significant improvement in terms of accuracy when we use a selected set of relevant features. We note that, for the "Wine" dataset, only two features are removed while more than half of the features are removed for "Heart" and "Ionosphere" datasets. This may explain the significant effect on results for the first two datasets. Regarding "Sat.Image" and "Segment", since the number of instances is high, this helps the classifier to achieve good accuracy even with the original set of features.

Comparison with other classifiers

In order to evaluate the efficiency of the AP-classifier, we compare its 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 $k = 1, 2, ..., 11$.
- **C4.5**: generating a pruned or unpruned C4.5 decision tree. We tune the classifier with different confidence factor used for pruning $C = 0.1, 0.2, ..., 0.5$.
- **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 $O = 2, 4, ...10$ 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 γ ($\gamma = 2^{-15}, 2^{-13}, ..., 2^3$) and the Poly-Kernel and we tune its *degree d*($d = 1, 2, ...10$)(also

called $exponent$). We also tune the complexity parameter $C = 2^{-5}, 2^{-3}, ..., 2^{15}$ with each type of kernel as recommended by [68].

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 2.3. 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 dataset, then we train and test the classifier using this selected value of this parameter. Classification results for k-NN, C4.5, JRIP and SVMs, displayed in Table 2.4, 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).

Evaluation of AP-classifier with regard to other ML classifiers

If we compare our results to those of the well known algorithms shown in Table 2.4, we note that:

- The AP-classifier performs more or less in the same way as the best known algorithms. Especially, Algorithm 1 outperforms all other classifiers Except SVM for 15 out of 20 datasets and performs similar to SVM for datasets "TicTac.", "Monk1", "Monk2", "Monk3" , "Hayes-Roth", "W.B. Cancer" and "Glass". Especially, we note that Algorithm 1 largely outperforms the k-NN, C4.5 and JRip and performs similar to SVM for the dataset "Monk2". In fact our algorithm perfectly classifies this dataset (as SVM) while other classifiers achieve maximum 67% of accuracy on this dataset.

- The computed average accuracy over 20 datasets for each classifier, confirms our observations and show that the AP-classifier is ranked the second just after SVM classifier.

- The classification success of Algorithm 1 for "Sat.Image", "Segent" "Glass", "Ecoli" and "Letter" (which have multiple classes) demonstrates its ability to deal with multiple class datasets.

- The analogy-based classifiers seem to be efficient when classifying data sets with a large number of attributes as in the case of "Ionosphere" and "Sat.Image" for example.

- Although, our algorithms do not use the same order on $\mathcal{S}^3$ as the approach developed in [57], we note that our results are better than the results obtained by the previous approach for numerical data, for the tested data sets "Diabetes" and "Sat. Image".

Statistical evaluation of AP-classifier The comparative studies with other classifiers are first carried out through the Friedman test [27]. It is a non-parametric 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 [18] 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.

Since the Friedman test, comparing AP-classifier (using definition A) and all other ML classifiers, provided a significant p-value$= 1.193e-09$, we then apply a post-hoc test after Conover and we use Bonferroni-type adjustment of the p-values. In Table 2.6, 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 show that:

- The SVM using RBF and Poly-Kernel significantly outperforms the IBK, the C4.5 and the JRIP classifiers. It also outperforms AP-classifier.
- It is clear that AP-Classifier significantly outperforms the IBK, the C4.5 and the JRIP classifiers.

Table 2.6: Results for the post-hoc test after Conover for ALL classifiers. 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)
C4.5	0.14901	-	-	-	-
JRIP	**0.00016**$_*$	0.68692	-	-	-
SVM(RBFKernel)	**1.4e-09***	**3.7e-15***	$<$ **2e-16***	-	-
SVM(PolyKernel)	**3.9e-12***	$<$ **2e-16***	$<$ **2e-16***	1.00000	-
AP-Classifier	**0.00659***	**1.6e-07***	**1.1e-11***	**0.00659**$_*$	**7.0e-05**$_*$

Nearest neighbors and AP-classifier

In this section, we focus more particularly on the links and differences between the AP approach and the k-NN approach.

First we would like to check if voting pairs $(\vec{a}, \vec{b})$ are close or not to the item to be classified. For this purpose, we compute for AP-classifier, among all these voting pairs, the frequency of those ones that are close to $\vec{d}$ (which is about the same as being close to $\vec{c}$). In case this frequency is rather low, we can conclude that our algorithms are not just another view of k-NN algorithms. From a practical point, we proceed as follows:

- In the nominal case, we compute the percentage of voting pairs $(\vec{a}, \vec{b})$ such that

  $$max(H(\vec{a}, \vec{c}), H(\vec{b}, \vec{c})) \leq \theta(n)$$

  where n is the number of features, H denotes the Hamming distance and $\theta(n)$ is a threshold depending on n. We consider 4 thresholds 1, 2, $\frac{n}{3}$ and $\frac{n}{2}$. For instance, $\theta(n) = 1$ is a threshold telling that $\vec{c}$ differs from $\vec{a}$ and $\vec{b}$ on maximum one feature, which means $\vec{c}$ is $very$ close to $\vec{a}$ and $\vec{b}$. $\theta(n) = \frac{n}{2}$ means that $\vec{c}$ differs from $\vec{a}$ and $\vec{b}$ on less than half of the features, which shows that $\vec{c}$ is relatively close to $\vec{a}$ and $\vec{b}$. If the percentage is low, it means the way we attribute a label to $\vec{d}$ is not by using its nearest neighbors!

- In the numerical case, we adopt the same method but using the extended version of Hamming distance, the $l1$ distance to deal with numerical values keeping the same threshold:

  $$max(l1(\vec{a}, \vec{c}), l1(\vec{b}, \vec{c})) \leq \theta(n)$$

The results for the nominal case are shown in Table 2.7 and those for the numerical case are in Table 2.8. Note that the Balance and Iris datasets contain only 4 attributes, this is why we only provide results for the column n/2 (column 2 is clearly the same). In the nominal case, we note that:

- The frequency of voting pairs $(\vec{a}, \vec{b})$ that are $very$ close to $\vec{c}$ (the Hamming distance is less or equal to one or two features) is very low. The highest frequency of voters that differs from $\vec{c}$ on only one (resp. two) attribute(s) is around 0.2% (resp. 2%) on all datasets.

- The frequency of voting pairs $(\vec{a}, \vec{b})$ differing from $\vec{c}$ on at least half of the features (last column in Table 2.7) is maximum 3% for the three first datasets, 16% for next three datasets, except for the Voting dataset for which the frequency is a bit higher.

These results show that voters $(\vec{a}, \vec{b})$ remain quite far from the item $\vec{d}$ to be classified in the nominal case.

In the numerical case (in Table 2.8), we can see that:

- For some datasets, voting pairs $(\vec{a}, \vec{b})$ seem relatively close to $\vec{c}$. The frequency of voting pairs differing from $\vec{c}$ up to one feature is about 30% for Iris, Glass, Ecoli and

Table 2.7: Frequency of voters that are close to c in the nominal case

Datasets	$\theta(n)$			
	1	2	n/3	n/2
Balance	0.0007	-	0.0004	0.0336
Car	0.0003	0.011	0.011	0.011
TicTacToe	0	0.0011	0.0079	0.0075
Monk1	0.0006	0.0193	0.0191	0.160
Monk2	0.0006	0.0189	0.0189	0.159
Monk3	0.0006	0.0199	0.0202	0.162
Voting	0.0018	0.0126	0.1244	0.303

W.B. Cancer datasets. This may mean that, for these datasets, examples have often relatively similar attribute values.

- For other datasets, the frequency of voters that are close to $\vec{c}$ on all features except one or two is very low as in the case of Wine, Heart and Sat.Image. In Table 2.3, we can see that these datasets include the largest numbers of features. As said before, dataset examples are likely to be more dissimilar in presence of a large number of attributes.
- At least 80% of the voting pairs differ from $\vec{c}$ on at least half of the features (last column in Table 2.8) if we except W. B. Cancer (where it is about 60%) and Heart (about 30%).

In the numerical case, the frequency of voters that are close to $\vec{c}$ is relatively high if compared to the nominal case. This is due to the cumulative effect of the extended Hamming distance that considers all features when computing the distance between pairs in case of numerical data, while in the nominal case, only completely opposite attribute values are considered in the Hamming distance.

Table 2.8: Frequency of voters that are close to c in the numerical case

Datasets	$\theta(n)$			
	1	2	n/3	n/2
Diabetes	0.126	0.857	0.857	0.999
W. B. Cancer	0.28	0.350	0.432	0.6025
Heart	0.0006	0.0152	0.326	0.321
Iris	0.341	-	0.341	0.813
Wine	0.0004	0.091	0.7836	0.999
Sat. Image	0.0001	0.044	0.523	0.894
Glass	0.352	0.727	0.341	0.990
Ecoli	0.295	0.939	0.938	0.998

Lastly, we want to check if the instance d is classified in the same way as the k-NN classifier. For this purpose, we computed the frequency of the case where both Algorithm 1 and kNN predict the correct label for d (this case is noted TT), the frequency of the

case where both classifiers predict an incorrect label (noted FF), the frequency where AP-classifier'prediction is correct and kNN'prediction is wrong (TF) and the frequency where AP-classifier prediction is wrong and kNN prediction is correct (FT). Results are collected in Table 2.9.

Table 2.9: Frequency of correctly/incorrectly classified instances $\vec{d}$ classified same/not same as the k-NN

Datasets	TT	FF	TF	FT
Balance	0.822	0.118	**0.041**	0.019
Car	0.911	0.045	**0.03**	0.015
TicTac.	0.992	0.0	**0.008**	0.0
Monk1	0.975	0.001	**0.023**	0.001
Monk2	0.652	0.001	**0.346**	0.001
Voting	0.909	0.036	**0.039**	0.017
Hayes-Roth	0.583	0.177	**0.209**	0.03
Diabetes	0.712	0.25	0.017	**0.021**
Cancer	0.964	0.024	**0.008**	0.004
Heart	0.795	0.166	0.02	0.02
Ionosphere	0.901	0.092	**0.005**	0.002
Iris	0.942	0.049	**0.007**	0.003
Wine	0.972	0.014	**0.009**	0.005
Glass	0.69	0.24	**0.038**	0.032
Ecoli	0.838	0.121	**0.025**	0.015

In this table, we note that:

- For most cases, AP-classifier and k-NN agree and classify instances in the same way (the highest frequency is seen in column TT).
- From Table2.4, we can see that the two classifiers achieve these results for different values of the optimized parameter k. For most of the tested datasets, AP-classifier uses larger number of nearest neighbors than the k-NN.
- This means that, for small values of k, we are still close to the logic of the classical k-NN classifier. Analogical classifiers gain their benefits (through using voter triples) when a largest variety of nearest neighbors are used.
- If we compare results in column TF and FT, we can see that the frequency of cases where AP-classifier yields the correct prediction while k-NN not (column TF) is significantly better than the opposite case (column FT). For example for "Monk2" and "Hayes-Roth", more than 20% of the total correctly classified instances are classified differently from the k-NN.
- This may explain good classification accuracies obtained with AP-classifier for these datasets which is significantly better than that obtained with the classical k-NN. For the dataset "Monk2" for example, if we compare results given in Table 2.4

for IBK and for our Algorithm, we can see that AP-classifier achieves very good results (99.77%) while IBK only performs 65%. In fact, we may think that good accuracies for these datasets are due to the high number of instances NOT classified in the same way as the nearby instance $\vec{c}$.

In fact, AP-classifier benefits from two basic differences if compared to the classic k-NN i) using large amount of voters for classification and ii) more complex calculation of the sum of elementary truth values P for each voter.

While k-NN uses a simpler voting-based strategy that directly applies a vote on the nearest neighbors classes to guess the class for $\vec{d}$ without any interest to the distance calculation (in k-NN, what counts is the rank of the nearest neighbor not its $exact$ distance to $\vec{d}$). These observations confirm the fact that AP-classifiers cannot be reduced to k-NN classifiers. However, we have to acknowledge that small improvements could normally be achieved using a weighted version of k-NN.

2.2.4 Scientific impact

This research work have been elaborated first on two conference papers respectively devoted to Boolean [8] and to numerical data [10]. A fully rewritten version of the conference papers, in particular, a deeper investigation of the ideas underlying the procedure, new algorithms and new experiments on a large variety of datasets have been published in a journal paper (see [15]).

2.3 AP-Rule-based Classification

2.3.1 Motivation

As introduced in the previous sections, many analogical proportion-based classification methods have been introduced a few years ago. The amazing results (at least in terms of accuracy) that have been obtained from such techniques are not easy to justify from a theoretical viewpoint. These techniques look in the training set for suitable triples of examples that are in an analogical proportion with the item to be classified, on a maximal set of attributes. This can be viewed as a lazy classification technique since, like k-NN algorithms, there is no static $model$ built from the set of examples.

For this reason we suggest an alternative method to build analogical proportion-based classifiers by statically building a set of inference rules during a preliminary training step. This creates a new classification algorithm that deals with pairs rather than with triples of examples. Our interest accorded to rule-based classification is motivated by the fact that

symbolic rules of the form if-then rules are very useful in most of existing expert systems in their knowledge representation. This is due to the naturalness and explanation ability of such rules that can offer to knowledge representation and reasoning in expert systems.

2.3.2 Contribution

We claim here that analogical classifiers behave as if a set of rules was build inductively during a pre-processing stage. To support intuition, we use an example inspired from the Golf data set (UCI repository [42]). This data set involves 4 multiple-valued attributes:

1: Outlook: sunny or overcast or rainy. ; *2: Temperature: hot or mild or cool* ;

3: Humidity: high or normal. ; *4: Windy: true or false*.

Two labels are available: 'Yes' (play) or 'No' (don't play).

Main assumptions. Starting from a finite set of examples, 2 main assumptions are made regarding the behavior of the class cl:

- Since the target relation cl is assumed to be a function, when 2 distinct vectors $\vec{x}$ and $\vec{y}$ have different labels ($cl(\vec{x}) \neq cl(\vec{y})$), the cause of the label switch is to be found in the switches of the attributes that differ. Take $\vec{x}$ and $\vec{y}$ in the Golf data set, as:

  $\vec{x} = (overcast, mild, high, false)$ and $cl(\vec{x}) = Yes$

  $\vec{y} = (overcast, cool, normal, false)$ and $cl(\vec{y}) = No$

  then the switch in attributes 2 and 3 is viewed as the cause of the 'Yes'-'No' switch.

- When 2 distinct $\vec{x}$ and $\vec{y}$ are such that $cl(\vec{x}) = cl(\vec{y})$, this means that cl does not preserve distinctness, i.e. cl is not injective. We may then consider that the label stability is linked to the particular value arrangement of the attributes that differ.

Patterns. Let us now formalize these ideas. Given 2 distinct vectors $\vec{x}$ and $\vec{y}$, they define a partition of $[1, n]$ as $A(\vec{x}, \vec{y}) = \{i \in [1, n] | x_i = y_i\}$ and $D(\vec{x}, \vec{y}) = [1, n] \setminus A(\vec{x}, \vec{y}) = \{i \in [1, n] | x_i \neq y_i\}$. Given $J \subseteq [1, n]$, let us denote $x|_J$ the subvector of x made of the x_j, $j \in J$. Obviously, $\vec{x}|_{A(\vec{x},\vec{y})} = \vec{y}|_{A(\vec{x},\vec{y})}$ and, in the binary case, when we know $\vec{x}|_{D(\vec{x},\vec{y})}$, we can compute $\vec{y}|_{D(\vec{x},\vec{y})}$. In the binary case, the pair $(\vec{x}, \vec{y})$ allows us to build up a *disagreement pattern* $Dis(\vec{x}, \vec{y})$ as a list of pairs $(value, index)$ where the 2 vectors differ. with $n = 6, \vec{x} = (1, 0, 1, 1, 0, 0), \vec{y} = (1, 1, 1, 0, 1, 0)$, $Dis(\vec{x}, \vec{y}) = (0_2, 1_4, 0_5)$. It is obvious that having a disagreement pattern $Dis(\vec{x}, \vec{y})$ and a vector $\vec{x}$ (resp. $\vec{y}$), we can get $\vec{y}$ (resp. $\vec{x}$). In the same way, the disagreement pattern $Dis(\vec{y}, \vec{x})$ is deducible from $Dis(\vec{x}, \vec{y})$. For the previous example, $Dis(\vec{y}, \vec{x}) = (1_2, 0_4, 1_5)$.

In the categorical case, the disagreement pattern is a bit more sophisticated as we have to store the changing values. Then the disagreement pattern $Dis(\vec{x}, \vec{y})$ becomes a

list of triple $(value1, value2, index)$ where the 2 vectors differ, with $value1$ being the attribute value for x and $value2$ being the attribute value for y. For instance, with the previously described Golf dataset, for the pair of given examples $\vec{x}$ and $\vec{y}$, $Dis(\vec{x}, \vec{y})$ is $\{(mild, cool)_2, (high, normal)_3\}$. Then we have two situations:

1. $\vec{x}$ and $\vec{y}$ have different labels, i.e. $cl(\vec{x}) \neq cl(\vec{y})$. Their disagreement pattern $Dis(\vec{x}, \vec{y})$ is called a *change pattern*. Then $Dis(\vec{y}, \vec{x})$ is also a change pattern.

2. $\vec{x}$ and $\vec{y}$ have the same label $cl(\vec{x}) = cl(\vec{y})$. Their disagreement pattern $Dis(\vec{x}, \vec{y})$ is called a *no-change pattern*. Then $Dis(\vec{y}, \vec{x})$ is also a no-change pattern.

To build up a change (resp. no-change) pattern, we have to consider all the pairs $(\vec{x}, \vec{y})$ such that $cl(\vec{x}) \neq cl(\vec{y})$ (resp. such that $cl(\vec{x}) = cl(\vec{y})$). We then build 2 sets of patterns P_{ch} and P_{noch}, each time keeping only one of the 2 patterns $Dis(\vec{x}, \vec{y})$ and $Dis(\vec{y}, \vec{x})$ to avoid redundancy. As exemplified below, these 2 sets are not disjoint in general. Take $n = 6$, and assume we have the 4 binary vectors $\vec{x}, \vec{y}, \vec{z}, \vec{t}$ in a training set TS:

- $\vec{x} = (1, 0, 1, 1, 0, 0), \vec{y} = (1, 1, 1, 0, 1, 0)$ with $cl(\vec{x}) = 1$ and $cl(\vec{y}) = 0$. Then, for $(\vec{x}, \vec{y})$, the disagreement pattern is a change pattern, i.e., $(0_2, 1_4, 0_5) \in P_{ch}$.

- $\vec{z} = (0, 0, 1, 1, 0, 1), \vec{t} = (0, 1, 1, 0, 1, 1)$ with $cl(\vec{z}) = cl(\vec{t})$. They have the same disagreement pattern as $\vec{x}$ and $\vec{y}$, which is now a no-change pattern $(0_2, 1_4, 0_5) \in P_{noch}$.

Now, given an element $\vec{x}$ in TS whose label is known, and a new element to be classified $\vec{y}$, if the disagreement pattern $Dis(\vec{x}, \vec{y})$ belongs to $P_{ch} \cap P_{noch}$, we do not get any hint regarding the label of $\vec{y}$. Then we remove the patterns from $P_{ch} \cap P_{noch}$: the remaining patterns are the *valid patterns* (still keeping the same notations for the resulting sets).

Rules. Thanks to the concept of pattern, it is an easy game to provide a formal definition of the 2 above principles. We get 2 general classification rules, corresponding to dual situations, for a new element $\vec{y}$ to be classified:

$$\text{Change Rule: } \frac{\exists \vec{x} \in TS, \exists D \in P_{ch} | (Dis(\vec{x}, \vec{y}) = D) \vee (Dis(\vec{y}, \vec{x}) = D)}{cl(\vec{y}) \neq cl(\vec{x})}$$

$$\text{NoChange Rule: } \frac{\exists \vec{x} \in TS, \exists D \in P_{noch} | (Dis(\vec{x}, \vec{y}) = D) \vee (Dis(\vec{y}, \vec{x}) = D)}{cl(\vec{y}) = cl(\vec{x})}$$

NoChange rules tell us when a new item $\vec{y}$ to be classified should get the class of its associated example $\vec{x}$, and Change rules tell the opposite. Let us note that if there is no valid pattern, then we cannot build up any rule, then we cannot predict anything! This has never been the case for the considered benchmarks.

It is straightforward to implement the previous ideas.

1. Construct from TS the sets P_{ch} and P_{noch} of all disagreement patterns.
2. Remove from P_{ch} and from P_{noch} the patterns belonging to $P_{ch} \cap P_{noch}$ to get the set of valid patterns.

The remaining change patterns in P_{ch} and no-change patterns in P_{noch} are used to build up respectively the *Change Rule Set* R_{ch} and *No-Change Rule Set* R_{noch}. In this context, we have implemented two different classifiers: the *Change Rule based Classifier* (CRC) and the *No Change Rule based Classifier* ($NCRC$), which have the same principles in all respect. The only difference is in the classification phase where the CRC only uses the set P_{ch} of pattern and applies the Change rules, whereas the second classifier $NCRC$ uses the no-change patterns P_{noch} and applies the No-Change rules to classify new items.

Classification. The classification process for CRC and $NCRC$ are detailed in the following Algorithms 2 and 3, where the Boolean function $Analogy(x, x', y)$ is true if and only if $card(\{cl(x), cl(x'), cl(y)\}) \leq 2$. For the NCRC, the $Analogy(x, x', y)$ always has a solution since classes associated to any No-Change rule r in R_{noch} are homogeneous. In terms of complexity, the algorithms are still cubic in the size of TS since the disagreement pattern sets have a maximum of n^2 elements and we still have to check every element of TS to build up a relevant pair with $\vec{y}$.

Algorithm 2 Change Rule Classifier

Given a new instance $\vec{y'} \notin TS$ to be classified.
$CandidateRules(c_j) = 0$, for each $j \in [1, l]$ (in the binary class case, $l = 2$).
for each $\vec{y}$ in TS **do**
  Construct the disagreement patterns $D(\vec{y}, \vec{y'})$ and $D(\vec{y'}, \vec{y})$
  for each change rule $r \in R_{ch}$ // r has a pattern $D(\vec{x}, \vec{x'})$ **do**
    if $Analogy(x, x', y)$ AND ($D(\vec{y}, \vec{y'}) = D(\vec{x}, \vec{x'})$ OR $D(\vec{y'}, \vec{y}) = D(\vec{x}, \vec{x'})$) **then**
      if $(cl(\vec{x}) = cl(\vec{y}))$ **then** $c^* = cl(\vec{x'})$ **else** $c^* = cl(\vec{x})$ **end if**
      $CandidateRules(c^*) + +$.
    end if
  end for
end for
$cl(\vec{y'}) = arg\max_{c_j} CandidateRules(c_j)$

With our approach, contrary to k-nn approaches, we always deal with pairs of examples: i) to build up the rules, ii) to classify a new item, we just associate to this item another one to build a pair in order to trigger a rule. Moreover, the two pairs of items involved in an analogical proportion are not necessarily much similar as pairs, beyond the fact they should exhibit the same dissimilarity. An analogical view of the nearest neighbor principle could be "close/far instances are likely to have the same/possibly different class", making an assumption that the similarity of the classes is related to the similarity of the instances. This does not fit, e.g., our No-Change rules where the similarity of the classes is associated with dissimilarities of the instances. More generally, while k-nn-like classifiers

Algorithm 3 No Change Rule Classifier

Given a new instance $\vec{y'} \notin TS$ to be classified.
$CandidateRules(c_j) = 0$, for each $j \in Dom(c_j)$.
for each $\vec{y}$ in TS **do**
  Construct the disagreement patterns $D(\vec{y}, \vec{y'})$ and $D(\vec{y'}, \vec{y})$
  for each no change rule $r \in R_{noch}$ // r has a pattern $D(\vec{x}, \vec{x'})$ **do**
    if $Analogy(x, x', y)$ AND ($D(\vec{y}, \vec{y'}) = D(\vec{x}, \vec{x'})$ OR $D(\vec{y'}, \vec{y}) = D(\vec{x}, \vec{x'})$) **then**
      $c^* = cl(\vec{y})$
      $CandidateRules(c^*) + +$.
    end if
  end for
end for
$cl(\vec{y'}) = arg\max_{c_j} CandidateRules(c_j)$

focus on the neighborhood of the target item, analogical classifiers "take inspiration" of information possibly far from the immediate neighborhood.

Example. Let's continue with the previous Golf example to show the classification process in Algorithm 2. Given three change rules r_1, r_2 and r_3:

$$r_{1(Yes-No)} = \{(sunny, overcast)_1, (false, true)_4\}$$

$$r_{2(No-Yes)} = \{(cool, mild)_2, (high, normal)_3\}$$

$$r_{3(No-Yes)} = \{(rainy, overcast)_1, (false, true)_4\},$$

and a new instance $\vec{y'}$ to be classified: $\vec{y'} : overcast, mild, normal, true, \rightarrow ?$

Assume that there are three training examples $\vec{y_1}$, $\vec{y_2}$ and $\vec{y_3}$ in T_s:

$$\vec{y_1} : sunny, mild, normal, false, \rightarrow Yes$$

$$\vec{y_2} : overcast, cool, high, true, \rightarrow No$$

$$\vec{y_3} : rainy, mild, normal, false, \rightarrow No$$

We note that disagreement patterns p_1, p_2 and p_3 corresponding respectively to the pairs $(\vec{y_1}, \vec{y'})$, $(\vec{y_2}, \vec{y'})$ and $(\vec{y_3}, \vec{y'})$ match respectively the change rules r_1, r_2 and r_3. Inferring the first rule predict a first candidate class "No" for $\vec{y'}$. In the same manner the second rule predict a class "Yes" and the third one also predict "Yes". The rule-based inference produces the following set of candidate classes for $\vec{y'}$: $Candidate = \{No, Yes, Yes\}$. So the most plausible class for $\vec{y'}$ is "Yes".

2.3.3 Experimental validation

This section provides experimental results for the two proposed classifiers. The experimental study is based on some data sets selected from the U.C.I. machine learning repository [42]. A description of these data sets is in Table 2.3.

We note that for all classification results given in the following, only half of the training set is used to extract patterns. We ensured that all class labels are represented in this data set. The classification results for the CRC or NCRC are summarized in the first and second columns of Table 2.10. We also tested a hybrid version of these classifiers called *Hybrid Analogical Classifier (HAC)* based on the following process. Given an instance $\vec{y}'$ to classify,

1. Merge the two rule subsets R_{ch} and R_{noch} into a single rule set R_{chnoch}.
2. Assign to y' the class label with the highest number of candidate rules in R_{chnoch}.

Classification results for HAC are given in Table 2.10, where we also give the mean number of Change (MeanCh) and No-Change rules (MeanNoCh) generated for each data set.

Table 2.10: Classification accuracies: mean and standard deviation of 10 cross-validations

Datasets	**CRC**	**NCRC**	**HAC**	**MeanCh**	**MeanNoCh**
Breast cancer	50.03 ± 8.03	**74.03±7.48**	73.39±8.44	6243.4	8738.5
Balance	82.82 ±5.8	**91.02±4.44**	90.51 ± 4.27	31736.2	20805.4
Tic tac toe	**98.3±5.11**	**98.3±5.11**	**98.3±5.11**	74391.9	86394.2
Car	79.54±4.23	**95.02± 2.16**	92.6 ±2.69	36526.6	20706.1
Monk1	90.52±6.16	**100±0**	99.54 ±1.4	9001.2	8644.6
Monk2	78.02 ±4.71	**100±0**	94.68 ± 4.38	7245.9	10607.8
Monk3	91.93±7.04	**97.93±1.91**	**97.93±1.91**	10588.0	10131.7

By analyzing classification performance in Table 2.10 we can see that:

• Overall, the analogical classifiers show good performance to classify test examples (at least for one of CRC and NCRC), especially NCRC.

• If we compare classification results for the two analogical classifiers, CRC and NCRC, we see that NCRC seems to be more efficient than CRC for almost all data sets, except the case of "Tic tac toe" where the two classifiers have the same accuracy.

• HAC shows good performance if compared to CRC and very close accuracies to NCRC for "Balance, Tic tac toe, Monk1 and Monk3". For the remaining datasets, the lower classification accuracy of Change rules may affect the efficiency of HAC.

• The analogy-based classifiers seem to be very efficient when classifying data sets with a limited number of attribute values and seems to have more difficulties for classifying data sets with a large number of attribute values. In order to evaluate analogical classifiers on such a dataset, we tested CRC and NCRC on "Cancer" (9 attributes, each of them having 10 different labels). From this additional test, we note that analogical classifiers are significantly less efficient on "Cancer" when compared to the state of the art algorithms. By contrast, if we look at the 3 "Monks" and "Balance" data sets, we note that these data sets have a smaller number of attributes and more importantly all attributes have a reduced number of possible values (the maximum number of possible attribute values in "Balance" and "Monks" is 5, and most of attributes have only 3 possible labels). This clearly departs from the "Cancer" situation. So we may say that this latter dataset is closer to a data set with numerical rather than categorical data. The proposed classifiers are basically designed for handling categorical attributes. We plan to extend analogical rule-based classifiers in order to support numerical data in future.

• In Table 2.10 we see that a huge number of rules of the two kinds are generated. We may wonder if a reduced subset of rules could lead to the same accuracy. This would mean that there are some redundancy among each subset of rules, raising the question of how to detect it. We might even wonder if all the rules have the same "relevance", which may also mean that some rules have little value in terms of prediction, and should be identified and removed. This might also contribute to explain why CRC has results poorer than NCRC in most cases.

• In the case of NCRC, we come apparently close to the principle of a k-NN classifier, since we use nearest neighbors for voting, but here some nearest neighbors are disqualified because there is no NoChange rule (having the same disagreement pattern) that supports them.

2.3.4 Scientific impact

This contribution have been published in a conference paper (see [9]).

2.4 Conclusion and discussion

In this chapter, using analogical proportions, we have presented two new ways to classify data, with Boolean, nominal or numerical attributes. We have first proposed a generic algorithm that covers all types of data and looks for triples $\vec{a}, \vec{b}, \vec{c}$ to build up a valid proportion with the new item $\vec{d}$ to be classified, where we take $\vec{c}$ as a neighbor of $\vec{d}$ in order to reduce the search space. An alternative approach to build analogical proportion-based learners is to statically build a set of inference rules during a preliminary training step. This creates a new classification algorithm that deals with pairs rather than with triples of

examples.

On the basis of the reported experiments, we may globally conclude that the AP-classifier achieves a reduced complexity when compared to analogical classifiers taking into account all triples, while maintaining an accuracy statistically better than the k-NN and in many case equivalent to SVM. Nevertheless, the execution time of the AP-classifier is still high. It would remain to find classes of problems where AP-classifiers are of particular interest. It may be the case when we have relatively few data at hand. However, it is worth pointing out that the AP-classifier relies on ideas quite different from the other existing classifiers, while providing results of similar quality.

Chapter 3

Oddness/evenness-based classification

3.1 Introduction

Analogical proportions, treated in Chapter 2, have recently been shown to belong to a larger family of so-called logical proportions [50]. These later relate a 4-tuple of Boolean variables, where the 8 code-independent logical proportions are of particular interest since their truth status remain unchanged if a property is encoded positively or negatively. These 8 logical proportions divide into 4 *homogeneous* proportions, which include the analogical proportion and 3 related proportions, and 4 *heterogeneous* proportions [54]. An heterogeneous proportion expresses the idea that there is an intruder among the 4 truth values, which is forbidden to appear in a specific position. Intuitively speaking, an item properly assigned to a class should not be (too much) an intruder in this class. It suggests that heterogeneous proportions can also be considered as a basis for classification by considering that a new item can be added to a class only if its addition leaves the class as even/not odd as possible.

In this chapter, we are rather interested to heterogeneous proportions as a counter part to homogeneous proportions studied in Chapter 2 and we explain how such proportions can be a useful tool for building a second family of classifiers named: *Oddness-based classifiers*.

This chapter is organized as follows: In Section 3.2, based on heterogeneous proportions, we first define a new oddness index that measures the extent to which an item may be seen as an intruder in a subset of items. Then based on this index, we propose a simple procedure to evaluate the oddness of an item with respect to a whole class in a local view, where the new item, should not appear at odds with a maximum number of (small) subsets of a considered class. In Section 3.3, starting from the results of Section 3.2 showing the efficiency of the classifier using subset of pairs, a more optimized algorithm, focusing especially on this type of subsets, have been proposed. In this later two options are investigated: in the first one, one of the two items of the subset is chosen among the k nearest

neighbors and the other one among the most remote elements to the item to be classified. In the second option, both items are selected among the k nearest neighbors keeping them distinct to each other. In Section 3.4, we propose an evenness index which is not the exact opposite of oddness index. As in the case of oddness-classification, an evenness-based classifier has been proposed that estimates the evenness of an item with respect to a whole class.

3.2 Oddness-based classification

3.2.1 Motivation

Many successful approaches have been proposed for classification purposes for a long time. Quite surprisingly, it has been recently shown that it was also possible to build another kind of classifiers based on analogical proportions. The fact that analogical proportions belong to a larger class of formulas, namely logical proportions, including heterogeneous proportions, led us to wonder if this latter kind of proportions might also be used with success to build classifiers. We have investigated this option in this work.

As introduced before, it is a commonsense principle to consider that a class cannot be reasonably assigned to a new item if this item would appear to be odd with respect to the known members of the class. On the contrary, the item should be even with respect to these class members for entering the class. A particular implementation of this principle has been recently explored in [12] by judging to what extent the item conforms with the majority of elements in any triple of members of the class. The idea of considering triples as a basis for estimating the evenness of the item with respect to the class has been motivated by two facts. First, triples are the only subsets where when the new item conforms with the minority, there is no longer any majority (with respect to a given feature) in the triple augmented with the new item. Second, being odd with respect to three other elements is closely connected with the idea of heterogeneous logical proportions, themselves dual of the homogeneous logical proportions where analogical proportion is a prominent case [54] (also used successfully in classification [45, 8, 15]).

We propose an oddness measure which in the case of triples can still be related to heterogeneous logical proportions, but may apply as well to subsets of any size, and can be extended to numerical features in a straightforward manner. Thus in this work, the evaluation of the oddness of an item with respect to a class relies on a local view, where the new item, should not appear at odds with a maximum number of (small) subsets of a considered class. This leads to a set of classifiers based on the oddness measure that we proposed.

3.2.2 Contributions

Based on heterogeneous proportions, we first define an index to evaluate the $oddness$ of an item (denoted x or d in the following) w.r.t. a multiset of Boolean values where the multiset is a triple. Then, we extend this index to multi-valued logic, before generalizing the extended oddness index to multisets of any size. Moreover, the values x or in S may be thought of as the values of the same feature for different items. Then we build an oddness measure from this index by cumulating them over features, and by considering collections of multisets S within the examples describing the same class $\mathcal{C}$ in a training set. This builds the basis for an Oddness-based classifier that is described at the end of this section.

Oddness index

The idea of oddness introduced below directly relies on the evaluation of the extent to which a new item to be added to a subset reinforces its heterogeneity and so appears as an $intruder$ in it.

An oddness measure for Boolean data

Let us remember the meaning of H_i: H_i holds iff there is an intruder among a, b, c, d and the parameter in position i is not this intruder. As shown in Table 3.1, each proportion H_i provides a piece of knowledge on the intruder and when combined with other pieces, we can pick out which one is the intruder among a, b, c and d. For example $H_1(a, b, c, d)$ = $H_2(a, b, c, d)$ = $H_3(a, b, c, d) = 1$ means that there is an intruder which is out of the multiset $\{a, b, c\}$.

Then we define the oddness of d w.r.t. $\{a, b, c\}$ by the following formula:

$$Odd(\{a, b, c\}, d) =_{def} H_1(a, b, c, d) \wedge H_2(a, b, c, d) \wedge H_3(a, b, c, d) \tag{3.1}$$

As an immediate consequence of equation 3.1, we have:

$$Odd(\{a, b, c\}, d) \rightarrow \neg H_4(a, b, c, d)$$

Due to the permutation properties of the H_i's, the right hand side of this definition is stable w.r.t. any permutation of a, b, c, then the multiset notation on the left hand side is justified. The truth table of Odd is given in Table 3.1.

It is clear that Odd holds only when the value of d is seen as odd among the other values: d is the intruder. Moreover Odd does not hold in the opposite situation where there is a majority among values in a, b, c, d and d belongs to this majority (e.g. $Odd(\{0, 1, 0\}, 0) = 0$), or there is no majority at all (e.g. $Odd(\{0, 1, 1\}, 0) = 0$).

A simple observation of Table 3.1 shows that the oddness index can be rewritten as

$$Odd(\{a, b, c\}, d) \equiv ((a \vee b \vee c) \not\equiv d) \wedge (a \wedge b \wedge c) \not\equiv d)) \tag{3.2}$$

Table 3.1: H_1, H_2, H_3 and Odd truth values

a	b	c	d	$\mathbf{H_1}$	$\mathbf{H_2}$	$\mathbf{H_3}$	$\mathbf{Odd}$
0	0	0	0	0	0	0	0
0	0	0	1	**1**	**1**	**1**	**1**
0	0	1	0	1	1	0	0
0	0	1	1	0	0	0	0
0	1	0	0	1	0	1	0
0	1	0	1	0	0	0	0
0	1	1	0	0	0	0	0
0	1	1	1	0	1	1	0
1	0	0	0	0	1	1	0
1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	1	0	1	0
1	1	0	0	0	0	0	0
1	1	0	1	1	1	0	0
1	1	1	0	**1**	**1**	**1**	**1**
1	1	1	1	0	0	0	0

Given a multiset a, b, c of 3 identical Boolean values, $Odd(\{a, b, c\}, d)$ can then act as a flag indicating if the 4th value d is different from the common value of a, b, c. Then the value d is at odds w.r.t. the other values.

Extension to numerical data

It is possible to extend the previous $oddness$ measure in order to handle variables with graded values (i.e. variables whose values belong to $[0, 1]$, after a suitable normalization of numerical data). For now, this oddness is just 0 or 1 (i.e. the truth value of $Odd(\{a, b, c\}, d)$), but we would like to consider tuples such as $(0.1, 0.2, 0.1, 0.8)$ and still consider that the 4th value is somewhat odd w.r.t. the 3 other ones. Using the standard counterpart of the logical connectives in the $[0, 1]$-valued Łukasiewicz logic [59], where:

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

Then, a direct translation of formula 3.2, is:

$$Odd(\{a, b, c\}, d) =_{def} \min(|\max(a, b, c) - d|, |\min(a, b, c) - d|) \tag{3.3}$$

It is easy to check that Odd remains code independent with graded values, i.e. changing values into their complement to 1.

Note that the expression of Odd given here is no longer the conjunction of the *multiple-valued extensions* of H_1, H_2, H_3 as given in [54], which would lead to a less satisfactory measure of oddness. Indeed, we are here interested in the oddness of d w.r.t. a multiset $\{a, b, c\}$, and not in picking out an intruder in the multiset $\{a, b, c, d\}$ as in [54].

Oddness with respect to multisets of various size Since we are interested in checking if d seems an intruder in a given multiset, we may consider multisets of any size when defining the oddness index. Indeed, the previous oddness index is not limited to multisets $\{a, b, c\}$ with 3 elements, and can be easily generalized to an index of oddness $Odd(S, x)$ of an item x w.r.t. a multiset S of any size. In the Boolean case, the oddness of a given Boolean value x with respect to a given multi-set of Boolean values $S = \{a_i \mid i \in [1, n]\}$ is defined as follows:

$$Odd(S, x) = Odd(\{a_i | i \in [1, n]\}, x) =_{def} \neg(\bigvee_{i\in[1,n]} a_i \equiv x) \wedge \neg(\bigwedge_{i\in[1,n]} a_i \equiv x) \quad (3.4)$$

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 multi-set notation. In the particular case of three Boolean variables ($n = 2$), this formula leads to the truth table in Table 3.2, in the case of four Boolean variables ($n = 3$), to the truth table in Table 3.3

Table 3.2: $odd(\{a_1, a_2\}, x)$ 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

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 multi-set 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 (e.g. $odd(\{0, 1, 0\}, 0) = 0$), or when there is no majority at all (e.g. $odd(\{0, 1, 1\}, 0) = 0$).

In the numerical case, this definition can easily be extended to a multiset S of values in $[0, 1]$ as follows:

Table 3.3: $odd(\{a_1, a_2, a_3\}, x)$ 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

$$Odd(S, x) =_{def} \min(|\max(S) - x|, |\min(S) - x|) \tag{3.5}$$

When restricted to Boolean values, this formula reduces to formula 3.4. 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:

- $odd(\{u, \cdots, u\}, v) = |u - v|$. 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.
- $odd(\{u, \cdots, u, v, \cdots, v\}, v) = 0$ which is consistent with the expected semantics of odd since v belongs to the multi-set $\{u, \cdots, u, v, \cdots, v\}$.
- if $\min_{i\in[1,n]} a_i \leq x \leq \max_{i\in[1,n]} a_i$, formula 3.5 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 multi-set. The oddness can be high (> 0.5) only if x is outside the convex hull of the multi-set.

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 $n \geq 3$.

- 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.

- When $n = 2$, formula 3.5 reduces to:
  $odd(\{a, b\}, x) = \min(|x - \max(a, b)|, |x - \min(a, b)|)$, i.e., $odd(\{a, b\}, x) = \min(|x - a|, |x - b|)$, i.e., the min of the distances between x and each of a and b. This is also the classical distance between x and the set $\{a, b\}$.

- 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 $\{a_i | i \in [1, n]\}$, i.e., $d(\{a_i | i \in [1, n]\}, x) = \min_{i \in [1,n]} |x - a_i|$. 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 3.5 , odd is the minimum of 2 distances to elements in the multi-set $\{a_i | i \in [1, n]\}$, while $d(\{a_i | i \in [1, n]\}, x)$ is the minimum of the distances to the n elements of the multi-set. 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.

As it may be the case for real datasets, we may have missing values. Obviously, when there is a missing value in the multiset S of size n, then a simple option is to consider the multiset S' of size $n - 1$ and to consider $Odd(S, x) = Odd(S', x)$ where S' has no longer any missing value.

Oddness measure for vectors

When it comes to real life application, it is not enough to represent individuals with a single Boolean or real value. Generally, individuals are encoded by a set of features. Based on the previously defined oddness measure, we have to define a new measure suitable for vectors. When dealing with vectors $\overrightarrow{x} \in [0, 1]^n$, Boolean vectors are also covered as a particular case. The Odd measure, defined respectively by equations (3.4) and (3.5) , are used to estimate to what extent a value x can be considered as odd among a multiset S of values. Thanks to the two latter formulas, assuming the independence of features, it is natural to compute the $oddness$ of a vector $\overrightarrow{x}$ as the sum of the $oddness$ for each feature $x_i \in \overrightarrow{x}$, as follows:

$$Odd(\mathcal{S}, \overrightarrow{x}) =_{def} \Sigma_{i=1}^{n} Odd(S_i, x_i) \in [0, n] \tag{3.6}$$

where x_i is the i-th component of $\overrightarrow{x}$, S_i is the multiset gathering the i-th components of the vectors in $\mathcal{S}$.

If our aim is to measure oddness, high values of $Odd(\mathcal{S}, \overrightarrow{x})$ (close to n) means that, for $many$ features, $\overrightarrow{x}$ appears as an intruder and may reduce the homogeneity when going from $\mathcal{S}$ to the multiset $\mathcal{S} \cup \{\overrightarrow{x}\}$. If $Odd(\mathcal{S}, \overrightarrow{x}) = 0$, no feature indicates that $\overrightarrow{x}$ behaves as an intruder and there is no obstacle for $\overrightarrow{x}$ to join the multiset S[1].

Global oddness measure

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

Indeed, we have to be careful because then $\{\overrightarrow{a_i} | i \in [1, n]\}$ is summarized by two vectors made respectively by the minimum and the maximum of the feature values among the examples of $\mathcal{C}$ (due to formulas 3.5 and 3.6). These two vectors have high chance to be *fictitious* in the sense that, usually, they are not elements of $\mathcal{C} = \{\overrightarrow{a_i} | i \in [1, n]\}$. Approximating our knowledge of the set $\{\overrightarrow{a_i} | i \in [1, n]\}$ 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 $odd(\{\overrightarrow{a_i} | i \in [1, n]\}, \overrightarrow{x})$ may not be a good marker of the oddness of x w.r.t. $\{\overrightarrow{a_i} | i \in [1, n]\}$ 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 $\mathcal{C}$, then compute $odd(S, \overrightarrow{x})$ and finally add all these atomic oddness indices to get a global measure of oddness of $\overrightarrow{x}$ w.r.t. $\mathcal{C}$. This approach leads to the following definition for oddness:

$$ODD_m(\mathcal{C}, \overrightarrow{x}) = \frac{1}{\binom{|\mathcal{C}|}{m}} \Sigma_{\mathcal{S} \subseteq \mathcal{C} s.t. |\mathcal{S}| = m} Odd(\mathcal{S}, \overrightarrow{x})$$

Clearly, the number $\binom{|\mathcal{C}|}{m}$ of subsets $\mathcal{S} \subset \mathcal{C}$ of size m is an increasing function of $|\mathcal{C}|$. When $\mathcal{C}$ is large, this number is not far from $|\mathcal{C}|^m$.

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

- For singletons:

$$ODD_1(\mathcal{C}, \overrightarrow{x}) = \frac{1}{|\mathcal{C}|} \Sigma_{\overrightarrow{a} \in \mathcal{C}} Odd(\{\overrightarrow{a}\}, \overrightarrow{x})$$

[1] It is clear that when dealing with classification task, $\mathcal{S}$ is just a set of examples, without any repetition, but obviously, its projections componentwise, the S_i's are multisets of Boolean or real values.

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

- For pairs:

$$ODD_2(\mathcal{C}, \overrightarrow{x}) = \frac{1}{\binom{|\mathcal{C}|}{2}} \Sigma_{\overrightarrow{a}, \overrightarrow{b} \in \mathcal{C}} Odd(\{\overrightarrow{a}, \overrightarrow{b}\}, \overrightarrow{x})$$

  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, $\binom{|\mathcal{C}|}{2}$ is more or less equal to $|\mathcal{C}|^2$, in practice, we simply use a normalization factor equal to $\frac{1}{\mathcal{C}^2}$.

- For triples:

$$ODD_3(\mathcal{C}, \overrightarrow{x}) = \frac{1}{\binom{|\mathcal{C}|}{3}} \Sigma_{\overrightarrow{a}, \overrightarrow{b}, \overrightarrow{c} \in \mathcal{C}} Odd(\{\overrightarrow{a}, \overrightarrow{b}, \overrightarrow{c}\}, \overrightarrow{x})$$

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

In the above evaluation, we consider *all* the subsets of $\mathcal{C}$ 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.

Algorithm

In the context of classification, our aim is to avoid oddness when classifying a new item. For this purpose, we first extend Odd index to deal with vectors instead of simple Boolean or numerical values, and then, build up a global oddness of an item $\vec{x}$ w.r.t. a class $\mathcal{C}$.

Let TS be a training set composed of instances $(\overrightarrow{z}, cl(\overrightarrow{z}))$, where $\overrightarrow{z} \in \mathbb{B}^n$ or $\mathbb{R}^n$, $cl(\overrightarrow{z})$ is the label of $\overrightarrow{z}$. Given a new instance $\overrightarrow{x} \notin TS$ without label, we have to allocate a label to $\overrightarrow{x}$ by looking for the class that better maintains its homogeneity when $\overrightarrow{x}$ is added to it. More formally, given the set $\mathcal{C}$ of instances in TS having the same label c, we estimate to what extent $\mathcal{C} \cup \{\overrightarrow{x}\}$ is odd or even. Based on the $oddness$ measure defined before, the idea is then to assign to $\overrightarrow{x}$ the label corresponding to the class $minimizing$ the oddness when $\overrightarrow{x}$ is added.

The previous procedure can be described with the pseudo-code of Algorithm 4. Algorithm 4 can deal with missing values thanks to the remark just before the extension to vectors in Section 3.2.2.

Algorithm 4 Oddness-based algorithm

Input: a training set TS of examples $(\overrightarrow{z}, cl(\overrightarrow{z}))$
a non nul integer m
a new item $\overrightarrow{x}$,
Partition TS into sets $\mathcal{C}$ of examples having the same label c.
for each $\mathcal{C}$ **do**
Compute $ODD_m(\mathcal{C}, \overrightarrow{x})$
end for
$cl(\overrightarrow{x}) = argmin_{\mathcal{C}} ODD_m(\mathcal{C}, \overrightarrow{x})$
return $cl(\overrightarrow{x})$

3.2.3 Experimental validation

We provide experimental results obtained with $Oddness$-based classifiers and we compare them to other machine learning classifiers.

The experimental study is based on 19 datasets taken from the U.C.I. machine learning repository [42]. A brief description of these data sets is given in Table 2.3 of Chapter 2. Since an oddness-based classifier is able to deal with both Boolean or multi-valued features in this study, we select from this Table 2.3, 8 datasets with categorical or Boolean attribute values and 10 datasets with only numerical attributes. In order to apply the Boolean and multiple-valued semantics framework, all categorical attributes are binarized and all numerical attributes are rescaled.

In terms of protocol, we apply the same testing protocol described in Section 2.2.3 of Chapter 2. Each displayed parameter in Tables 3.5 and 3.6 (that we denote β) is the average value over the 5 different values (one for each run). The results shown in Tables 3.5 and 3.6 are the average values obtained from 5 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 sub-section), we simply apply a standard 10 fold cross-validation still repeated 5 times to get stable results.

Table 3.4 provides mean accuracies and standard deviations obtained with our implementations of Odd_1, Odd_2 and Odd_3.

These preliminary results show that:

- 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%.
- For ten 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).

Table 3.4: 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**

- 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 subsection, we apply different optimizations, focusing on Odd_2 classifier.

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 $\vec{x}$ *in the class*. In that case, we have to adjust the normalization factor which becomes $\frac{1}{k\times|\mathcal{C}^{n-1}|}$ in the Odd_n formula.

In Table 3.5, 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.

As we can see in this Table 3.5, 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.

Table 3.5: Classification accuracies given as mean and standard deviation with improved $Odd_1(NN)$, $Odd_2(NN, Std)$ and $Odd_3(NN, Std, Std)$ obtained with the best parameter β.

Datasets	$Odd_1(NN)$		$Odd_2(NN, Std)$		$Odd_3(NN, Std, Std)$	
	Acc.	β	Acc.	β	Acc.	β
Balance	82.79±4.29	9	87.4±4.08	21	88.62±3.4	2
Car	91.95±3.31	8	92.04±4.04	7	90.93±4.03	5
Monk1	99.91±0.09	6	99.77±0.23	4	99.31±3.39	1
Monk2	66.54±2.1	20	64.45±3.1	19	60.93±4.16	17
Monk3	99.95±0.05	3	99.95±0.72	1	99.95±0.05	2
Spect	82.74±6.49	13	83.95±4.72	9	84.1±4.58	5
Voting	93.04±3.2	9	94.23±3.95	10	93.81±2.86	11
Hayes-Roth	63.17±12.41	14	78.35±11.94	3	79.37±9.74	7
Diabetes	75.06±3.25	17	76.28±3.83	17	75.91±4.58	15
Cancer	97.07±2.19	3	97.27±1.34	5	97.04±2.24	5
Heart	82.33±5.15	13	82.52±7.87	13	82.2±4.01	13
Magic	78.78±1.58	13	79.05±3.13	16	74.53±3.02	17
Ionosphere	90.61±3.84	1	92.09±3.32	8	90.55±4.05	14
Iris	94.56±4.11	9	94.97±4.41	9	94.64±5.32	7
Wine	97.55±3.07	11	98.47±2.52	5	97.16±4.76	5
Sat. Image	94.79±2.78	3	95.03±3.08	1	93.43±2.38	1
Segment	96.76±1.3	2	96.67±1.44	1	95.31±2.14	3
Glass	72.87±8.1	3	75.84±9.8	3	72.22±8.19	3
Letter	75.66±3.34	2	75.86±4.57	2	73.8±2.81	2
Average	**86.11**		**87.59**		**86.52**	

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.

Looking at the results of $Odd_2(NN, Std)$ in Table 3.5, we can draw the following conclusions:

- It is clear that this optimized classifier is significantly more efficient than the basic classifier Odd_2 for most data-sets. 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.

- 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 us "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 [13] [16]. It remains to investigate what would be a suitable value for k leading to the best accuracy for any dataset.

- 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 multi-valued oddness measure correctly extends the Boolean case.

To evaluate the efficiency of the $oddness$-classifiers, we compare their accuracy to existing classification approaches previously presented in Section 2.2.3 of Chapter 2: IBK, $C4.5$, $JRIP$ and SVM. We also keep the same tuned parameter, except for IBK we extend the values of parameter k to $k = 1, 2, ..., 21$.

Table 3.6: 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	(32768,0.00195)	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	(32768,0.03125)	99,19±0,19	(32768,2)
Monk1	99.95±0.05	3	97.77±2.95	0.1	94.63±9.29	3	100	(32768,0.5)	100	(32768,6)
Monk2	67.04±1.19	13	95.32±2.22	0.4	79.68±6.14	10	100	(32768,0.03125)	100	(32768,2)
Monk3	99.95±0.05	1	100.0±0.0	0.1	99.91±1.44	2	100	(32768,0.5)	100	(32768,7)
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	(32768,0.0078)	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	(32768,0.5)	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	(32768,2)	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±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	(32768,0.5)	82,93±0,54	(0.5,3)
Average	**86.43**		**86,81**		**85,63**		**90,87**		**90,97**	

In Table 3.6, it has to be noted that, in the case of Boolean or nominal data, results of ML classifiers slightly differ from those previously shown in Table 2.4 of Chapter 2 since feature values, for these datasets, has been binarized in this case.

If we compare the results of $Odd_2(NN, Std)$ classifier to those of machine learning algorithms in Table 3.6, we note that:

- 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".

- 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".

- 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.
- $Odd_2(NN, Std)$ has close classification results to those of analogy-based classifier in the numerical case [10, 15] for most datasets. In the Boolean case, both $oddness$-based and $analogy$-based classifiers [8, 15] achieve good results for "Balance", "Car", "Monk1" and "Monk3". For "Monk2" dataset, 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 [22].

3.2.4 Publications

This research work have been published first in a conference paper (see [13]) then have been gathered and substantially extended in a journal paper(see [16]).

3.3 Oddness-based classification using more constrained pairs

3.3.1 Motivation

In Section 3.2, 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 contribution, 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.

3.3.2 Contributions

We have investigated two different options in this context:

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 $\overrightarrow{x}$. 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 $\overrightarrow{b}$ as an MRE might be justified by the fact that, in this case, the interval of values $[a_i, b_i], i = 1, ..., m$ remains sufficiently large since vectors $\overrightarrow{a}$ and $\overrightarrow{b}$ 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, $Odd(\overrightarrow{x}, \mathcal{C})$ 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 2 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.

Odd2 using two nearest neighbors

After experimenting the first option, 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.

3.3.3 Experimental validation

Results

In Table 3.7 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. In Table 3.7, 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 $Odd_2(NN, NN)$.

From Table 3.7, we can draw the following comments:

- $NMRE$ 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.

Table 3.7: Classification accuracies given as mean and standard deviation with $Odd_2(NMRE)$ and $Odd_2(NN, NN)$ obtained with the best parameter β.

Datasets	$Odd_2(NMRE)$		$Odd_2(NN, NN)$	
	Acc.	β	Acc.	β
Balance	82.62±3.06	(16,13)	83.9±5.02 (-)	17
Car	91.19±2.68	(6,20)	92.45±3.88(.)	7
Monk1	99.43±0.79	(7,7)	99.81±0.67(.)	6
Monk2	65.42±3.27	(19,2)	67.46±3.25(+)	16
Monk3	99.37±1.4	(5,12)	100(.)	5
Spect	83.14±8.15	(14,16)	84.55±4.42(+)	16
Voting	93.45±4.99	(5,15)	95.33±2.84(+)	17
Hayes-Roth	74.59±8.93	(4,19)	77.73±9.62(-)	7
Diabetes	75.5±4.53	(16,11)	75.28±4.26(-)	18
Cancer	97.15±2.61	(4,9)	97.24±1.63(.)	12
Heart	83.41±7.04	(10,5)	82.54±5.06(.)	19
Magic	78.06±3.48	(14,18)	79.23±2.41(.)	18
Ionosphere	91.18±4.51	(1,18)	91.78±4.95(-)	15
Iris	94.94±4.33	(11,9)	94.57±4.11(-)	14
Wine	97.2±3.19	(3,15)	98.37±2.77(.)	9
Sat. Image	94.49±2.08	(2,12)	95.38±2.59(+)	5
Segment	96.67±1.25	(2,9)	96.79±1.17(+)	4
Glass	74.94±8.58	(3,5)	77.93±7.27(+)	3
Letter	75.1±2.77	(3,17)	78.04±3.76(+)	5
Average	**86.73**		**87.81**	

- The average value of parameter k' over all datasets is higher than that of parameter k.
- $NMRE$ is efficient to classify both Binary or numerical datasets as in the case of Odd_2 classifier.
- 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.

Results of $Odd_2(NN, NN)$ in Table 3.7 shows that:

- If we compare results of the classifier using pairs with two nearest neighbors to those of the basic Odd_2 classifier in Table 3.4, it is clear that this third optimized option also performs largely better than the basic Odd_2 classifier.
- In order to compare the $Odd_2(NN, NN)$ to the $Odd_2(NN, Std)$ classifier, in the $Odd_2(NN, NN)$ column of Table 3.7 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 7 datasets and worst for 5 datasets.

Table 3.8: 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***

Statistical evaluation of $Oddness$-classifiers

As in case of Analogy-based Classifiers described in Chapter 2, we first carry out a Friedman test [27] then a post-hoc test after Conover [18] to calculate the corresponding levels of significance.

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$=$ $1.295e-9$, we then apply a post-hoc test after Conover and we use Bonferroni-type adjustment of the p-values.

In Table 3.8, 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:

- 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 $Odd_2(NN, Std)$.
- It is clear that $Odd_2(NN, Std)$ significantly outperforms the IBK, the C4.5, the JRIP classifiers and also $Odd_2(NMRE)$.

- 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.
- The $Odd_2(NN, NN)$ classifier is largely better than IBK, the C4.5 and the JRIP.
- 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.
- 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 3.5 since they have close average accuracies other 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.

Comparison of Oddness-based classifiers 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 least 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 $\overrightarrow{x}$ 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 [19].

The experimental results given in the previous sections (see for example first column in Table 3.5) 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.

Table 3.9: Comparative study between oddness classifiers and IBk in terms of procedure and complexity

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

In Table 3.9, 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 $Odd2(NN, Std)$ 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:

- more than 18000 for Odd2,

- Almost 2900 for the Odd2(NN,Std) with k=15,

- ONLY about 100 for Odd2(NN,NN) with k=15.

3.3.4 Publications

This contribution have been achieved first in the context of a conference paper (see [14]). This work have been extensively developed in a journal paper (see [17].

3.4 Evenness-based classification

3.4.1 Motivation

Similarly to the oddness index, and still based on heterogenous proportions, we consider that a new item can be added to a class only if its addition leaves the class as even as possible: the new item should rarely be an intruder with respect to any triple of items known to be in the class.

In the evenness view, triples are the only subsets where when the new item conforms with the minority for a given Boolean feature, there is no longer any majority (with respect to this feature) in the triple augmented with the new item. Then one can estimate to what extent a new item fits with the majority of elements in any triple of members of a class on a set of features.

3.4.2 Contributions

In the following, we define a new Evenness index for Boolean data then extend it to deal with feature vectors and finally propose a global evenness measure. This latter is useful to build an evenness-based classifier. We also lay bare the relation between evenness and oddness

Evenness index

Adopting a dual viewpoint, we may want to know if adding a new element to a given subset of items keeps it as homogeneous as it is, i.e., the newcomer does not appear as an intruder in this subset, and rather agrees with its majority. Homogeneity can be considered as a kind of $evenness$ of the newcomer w.r.t. the existing items of the subset. In this subsection, we advocate a way of judging $evenness$ on the basis of the majority, if any, inside the triples. Contrary to the oddness definition, where all $H_i, i = 1, 2, 3$ are required to define the oddness index, only H_4 is needed for defining an evenness index, thus denoted $Even_4$.

An evenness measure for Boolean data Since the idea is to agree with a majority, we notice that the smallest multisets S of elements where majority makes sense are clearly

triples. Let consider three Boolean values a, b, c in S. Then, in a Boolean world, there are two possibilities, either $a = b = c$, or two of the three are equal. In both cases, a strict majority takes place. Let m denote the majority value. Now consider the newcomer d, either $d = m$, and m remains the majority value in $\{a, b, c, d\}$, or $d \neq m$, and *there is no longer any majority in* $\{a, b, c, d\}$ (two values are equal to 1 and two values to 0). Only with the first case, d conforms to the majority.

Note that if we consider *larger* subsets S, even with only 4 elements rather than 3, it becomes possible that the newcomer increases an existing minority, without changing the majority. Indeed, the majority value that may be shared by 3 elements in the 4-elements multiset will then remain unchanged in the 5-elements multiset resulting from the arrival of a fifth element whatever its value. A similar phenomenon takes place if we start with larger subsets S having 5 elements or more. So we are losing a distinctive property of 3-elements subsets which have a different majority behavior depending if d conforms or not to the majority in the 3-elements subset. *This means that triples are the only subsets such that adding an item that conforms to the triple minority destroys the majority*. Thus, 3-elements subsets are able to clearly discriminate, among different d those that conform to the majority of the triple.

The idea of majority just described helps us to define a new $evenness$ measure via the heterogeneous proportions. Let us recall the semantics of H_i: H_i holds iff there is an intruder among a, b, c, d and the parameter in position i is not this intruder. As a consequence, H_i implies that there is a majority of values among (a, b, c, d) and the value in position i conforms to the majority of values appearing among the 3 other positions (i.e. the multiset of values $\{a, b, c, d\}$ is more or less even). But the reverse implication does not hold since when the 4 parameters have identical value, $\forall i \in [1, 4], H_i(a, b, c, d) = 0$. Then, to have a concise Boolean definition for "there is a majority of values among the parameters a, b, c, d and the parameter in position i belongs to this majority of values", we need to consider the case where all the values are identical by using the following formula:

$$Even_i(a, b, c, d) =_{def} H_i(a, b, c, d) \vee Eq(a, b, c, d) \tag{3.7}$$

where $Eq(a, b, c, d) =_{def} (a \equiv b) \wedge (b \equiv c) \wedge (c \equiv d)$. Thus, with $Even_i$ we take into account the special case where all the values are equal. The truth table of $Even_4$ is given in Table 3.10. It is clear that $Even_4$ holds only when the value of d belongs to a majority of the parameter's values. And $Even_4$ does not hold in an opposite situation where there is no majority of values as it is the case for $Even_4(0011)$ or $Even_4(0110)$.

The situations where $Even_4(a, b, c, d) = 1$ exactly cover the two cases already mentioned where d is identical to the majority value in the triple $\{a, b, c\}$ (is not the intruder), namely either $a = b = c$, or two of the three are equal to d. So the fact that d joins $\{a, b, c\}$, when $Even(a, b, c, d) = 1$, leaves the resulting subset as *even* as it was, hence the name, and in fact the majority is reinforced by the arrival of d. Note also that $Even_4(a, b, c, d)$ is left unchanged by any permutation of $\{a, b, c\}$. This means that the ordering inside triples

Table 3.10: H_4, Eq and $Even_4$ truth values

a	b	c	d	$\mathbf{H_4}$	$\mathbf{Eq}$	$\mathbf{Even_4}$
0	0	0	0	0	1	1
0	0	0	1	0	0	0
0	0	1	0	1	0	1
0	0	1	1	0	0	0
0	1	0	0	1	0	1
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	1	0	1
1	0	0	0	1	0	1
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	1	0	1
1	1	0	0	0	0	0
1	1	0	1	1	0	1
1	1	1	0	0	0	0
1	1	1	1	0	1	1

does not matter. Besides, $Even_4(a, b, c, d) = Even_4(\overline{a}, \overline{b}, \overline{c}, \overline{d})$ where $\overline{x} = 1$ if $x = 0$ and $\overline{x} = 0$ if $x = 1$, expressing that $Even_4(a, b, c, d)$ does not depend on the way the information is encoded. From now on, $Even_4$ will be denoted $Even$ as this is the only option we use with $i = 4$.

Relation between oddness and evenness in the Boolean case The oddness and evenness Boolean functions have been built by truth tables inspection. However, these 2 functions exhibit noticeable links. Despite the fact that their name might suggest that oddness and evenness capture dual concepts, it is not the case that $Even(a, b, c, d) \equiv \neg Odd(\{a, b, c\}, d)$. In fact, the relations between the 2 measures are as follows (where I denotes the inverse paralogy defined in Chapter1:

Property 7

$$Even(a, b, c, d) \equiv \neg Odd(\{a, b, c\}, d) \wedge \neg I(a, b, c, d)$$
$$Odd(\{a, b, c\}, d) \equiv \neg Even(a, b, c, d) \wedge \neg I(a, b, c, d)$$
$$I(a, b, c, d) \equiv \neg Even(a, b, c, d) \wedge \neg Odd(\{a, b, c\}, d)$$

This can be easily checked on the truth tables. This reflects the fact that Odd and $Even$ and I are mutually exclusive. Let us note that $Even(a, b, c, d) \rightarrow \neg Odd(\{a, b, c\}, d)$, that $\neg Even(a, b, c, d) \rightarrow \neg H_4(a, b, c, d)$ and that $Odd(\{a, b, c\}, d) \rightarrow \neg H_4(a, b, c, d)$ as well.

A more complete discussion of the relation between oddness and evenness can be found in [55].

Dealing with missing values Missing information is quite common in real life datasets and a way to extend the semantics of analogical proportion to deal with this issue has been deeply investigated in [54] for instance. In fact, such an approach can be also applied here, as explained now.

Still keeping a logical approach and considering that '?' denotes a missing value (i.e. an information is unknown), the idea is to extend the truth table of the $Even$ formula as

follows: $Even(?,0,0,0) = Even(0,?,0,0) = Even(0,0,?,0) = 1$, $Even(?,1,1,1) = Even(1,?,1,1) = Even(1,1,?,1) = 1$, and $Even(x,y,z,t) = 0$ for any other pattern including at least a missing value '?'. It is clear that, with the 6 first patterns, whatever the candidate value of the missing feature, the 4th argument belongs to the majority and cannot be an intruder. In all the remaining cases, where we have no certainty regarding the status of d, we adopt a cautious behavior by considering that $Even$ does not hold.

As in the case of Oddness measure, the evenness measure is extended to vectors as:

$$Even(\mathcal{S}, \overrightarrow{x}) =_{def} \Sigma_{i=1}^{n} Even(S_i, x_i) \in [0, n] \tag{3.8}$$

where x_i is the i-th component of $\overrightarrow{x}$, S_i is the multiset gathering the i-th components of the vectors in $\mathcal{S}$. Note that in the case of $Even$, the set $\mathcal{S}$ has exactly 3 elements, but we keep the set notation for sake of notation uniformity.

Global Evenness measure

Our aim here is to maintain homogeneity or evenness inside each class $\mathcal{C}$. Reasoning similar to the previous oddness index leads to a definition of evenness as follows (since we use only subsets of cardinality 3):

$$EVEN(\mathcal{C}, \overrightarrow{x}) = \frac{1}{\binom{|\mathcal{C}|}{3}} \Sigma_{\mathcal{S} \subseteq \mathcal{C} s.t. |\mathcal{S}|=3} Even(\mathcal{S}, \overrightarrow{x}).$$

Exactly the same optimization process applies to EVEN measure, as in case of oddness, leading to:

$$\frac{1}{|\mathcal{C}|^2} \Sigma_{j=1}^{k} (\Sigma_{\mathcal{S} \subseteq \mathcal{C} \setminus \{\overrightarrow{y_j}\} s.t. |\mathcal{S}|=2} Even(\mathcal{S} \cup \{\overrightarrow{y_j}\}, \overrightarrow{x}))$$

We also apply the same Algorithm 4 described in Section 3.2.2 by replacing the ODD measure by the EVEN measure and appropriately assigning the class maximizing Evenness.

3.4.3 Experimental validation

In order to better control the meaning of $EVEN(\mathcal{C}, \overrightarrow{x})$, we may focus on triples for which $\overrightarrow{x}$ is an intruder for at most $n - l$ features, where $l = 0, 1, \cdots$. Instead of keeping all the triples, we can just choose a threshold $l \in [0, n]$, then consider $Even(\overrightarrow{a}, \overrightarrow{b}, \overrightarrow{c}, \overrightarrow{x})$ only for the triples $(\overrightarrow{a}, \overrightarrow{b}, \overrightarrow{c})$ in $\mathcal{C}^3$ such that

$$Even(\overrightarrow{a}, \overrightarrow{b}, \overrightarrow{c}, \overrightarrow{x}) \geq l,$$

i.e. we want $Even$ to hold over at least l features. We denote $EVEN^l(\mathcal{C}, \overrightarrow{x})$ this measure where we just reduce the number of candidate triples by filtering over l. As a consequence, the evenness-based algorithms have 2 parameters: k the number of considered nearest neighbors and l the minimum number of features where the $Even$ proportion should hold.

Results

Table 3.11 provides accuracies results for the evenness-based classifier obtained with a 10-fold cross validation and for two values of k and l (k being the number of nearest neighbors of $\vec{x}$, l refers to the number of attributes j of $\vec{x}$ such that x_j belongs to a majority). The best results are in bold.

Table 3.11: Classification accuracies given as mean and standard deviation obtained with $Even$

Datasets		$Even$			
Value of k		**1**	**3**	**5**	**11**
Balance	l=n	67.29± 5.2	71.48±6.49	76.43±4.85	78.23±4.57
	l=n-1	78.04±4.91	83.28±3.44	**87.08±3.22**	86.48±3.45
Car	l=n	92.6±2.87	92.84±2.82	93.05±2.83	**93.27±2.7**
	l=n-1	89.63±3.43	90.35±3.0	91.58±2.52	91.75±2.42
Spect	l=n	81.53±6.67	81.86±7.93	**82.61±8.16±**	82.32±8.54
	l=n-1	81.21±6.52	81.14±6.83	81.37±6.82	81.74±7.1
Voting	l=n	94.29±3.67	94.99±3.96	94.94±3.96	**95.12±3.71**
	l=n-1	94.25±3.94	94.95±4.24	94.9±4.24	94.99±3.93
Monk1	l=n	**100**	**100**	**100**	**100**
	l=n-1	**100**	99.95±0.05	99.91± 0.64	99.95± 0.05
Monk2	l=n	38.31±4.09	41.37±4.66	45.54±5.04	**50.68±4.3**
	l=n-1	30.87±5.85	34.14±4.46	37.46±4.91	42.61±5.23
Monk3	l=n	**100**	**100**	**100**	**100**
	l=n-1	99.77±0.71	99.22±1.94	98.76±2.42	98.49±2.76

When we analyze results in Table 3.11, we can see that:

• In general, the best classification rates are obtained for $l = n$. This means that the classifier is likely to be more accurate when the classification is made on the basis of triples w.r.t. which $\overrightarrow{x}$ is not an intruder for *any* attributes. However, for some datasets such as Balance and Monk2, the classifier needs to consider more levels l when it is difficult to satisfy the constraint $Even(\overrightarrow{a}, \overrightarrow{b}, \overrightarrow{c}, \overrightarrow{x}) \geq l$ for $l = n$ or even $l = n - 1$. Thus, we also tested smaller levels of l and for "Balance" data set, we get an accuracy equal to 89.25 ± 2.4 for $l = n - 3$.

• The classifier shows good classification results for data sets "Balance", and "Car" (which have multiple classes). This shows that evenness-based classifiers are able to deal with multiple class data sets.

• If we compare results of the evenness-based classifier with machine learning algorithms

in Table 3.6, we note that the proposed classifier is as good as the best known algorithms. Especially, the basic classifier, with large k works as well as any other classifiers for data sets "Balance" (for $l = n - 3$), "Spect.", "Monk1" and "Monk3" (for $l = n$). Moreover, evenness-based classifier outperforms IBK for all data sets except Monk2.

Comparison between oddness-and evenness-based classifiers

Although oddness and evenness indexes are not the exact opposite of each other, the results obtained by minimization in oddness-based classifiers and by maximization in evenness-based classifiers are quite close, as can be seen by comparing Table 3.5 with Table 3.11, for Boolean features. However, the evenness index and measure have not been defined in the case of numerical features.

3.4.4 Publications

This research work have been published first in two conference papers (see [12, 11]) then extended in a journal paper (see [16]).

3.5 Conclusion

Starting from heterogeneous proportions, we have established a way to define an oddness measure and an evenness measure in order to estimate to what extent a new item does not conform, or conforms, to a candidate class. Then, testing on classical benchmarks coming from UCI repository, we have compared an oddness-based classifier and an evenness-based classifier with standards methods in classification, as well as with analogical proportion-based classifiers. Our experiments empirically highlight the good behavior of heterogeneous logical proportion-based classifiers.

On the basis of the reported experiments, we have seen that, regarding oddness measure, Odd_2 classifier (based on ODD_2 measure) stands out of the crowd. In fact, if we are back to the basic brick Odd_2 of ODD_2 measure, it is clear that, when $|S| = 2$, $Odd(S, \vec{x}) = 0$ as soon as $\vec{x} \in S$. This is not generally the case as soon as $|S| > 2$. This suggests that ODD_2, built up on the sum of atomic $Odd(S, \vec{x})$ with $|S| = 2$, may be a better marker of the oddness of a given element $\vec{x}$ inside a class $\mathcal{C}$ than any other measure ODD_i with $i > 2$. Still the global oddness and evenness measures have no obvious remarkable properties. This question may be addressed in future works.

Apart from a formal investigation of the properties of oddness and evenness measures, their merits would need to be studied in greater detail in order, for instance, to more precisely assess the expected accuracy of oddness or evenness-based classifiers. Then,

one might think to use them in conformal predictors [62, 67, 66], as first experimented in [12] with an evenness-based classifier. Indeed the oddness measure may be considered as a non conformity measure, while the evenness measure would be a conformity measure.

Chapter 4

Analogy-based Preference learning

4.1 Introduction

A panoply of research works has been developed to deal with preference learning problems (see, e.g., [28]) that can be stated as the following. Given a set E of preferences of the form $x^k \succeq y^k$ ($k = 1, \cdots, m$), representing what we know about the preferences of an agent about some pairs of choices, can we predict its preference between two other choices x and y? First, an idea is to make the assumption that the preferences of the agent obey a weighted sum aggregation scheme, whose weights are unknown. Then, we might think of finding a sampling of systems of weights summing to 1 that are compatible with the constraints induced by E. But, enumerating the vertices of the polytope defined by the system of inequations corresponding to the preferences in E is a NP hard problem that is not easy at all to handle in practice [36]. That's why we have chosen to explore another route in this study, based on the exploitation of a pattern avoiding contradictory trade-offs, and patterns expressing that preferences should go well with analogical proportions. This idea which may sound fancy at first glance is based on the empirical evidence that analogical proportion-based classifiers work well and the theoretical result that such classifiers make no error in the Boolean case when the labeling function is affine [22]. A result of the same nature might be conjectured when attributes are nominal rather then Boolean.

Analogical reasoning is reputed to be a valuable heuristic means for extrapolating plausible conclusions on the basis of comparisons. A simple form of this idea is implemented by case-based reasoning (CBR) [1], where conclusions known for stored cases are tentatively associated to similar cases. A more sophisticated option relies on the idea of analogical proportions. Given four items $\vec{a}$, $\vec{b}$, $\vec{c}$ and $\vec{d}$ described by their vector of features values for a considered set of features. As described in Chapter1, the inference process is based on triples of vectors rather than taking vectors one by one as in case-based reasoning. The underlying idea is that if four items $\vec{a}$, $\vec{b}$, $\vec{c}$, $\vec{d}$ are making an analogical proportion

on describing features, an analogical proportion may hold as well on another mark pertaining to them, and then if this mark is known for $\vec{a}$, $\vec{b}$ and $\vec{c}$, one may compute it for $\vec{d}$ in such a way that the marks make an analogical proportion.

To the best of our knowledge, the only approach also aiming at predicting preferences on an analogical proportion basis is the recent paper [26], which only investigates "the horizontal reading" of preference relations. However, we are only intended to predict preferences relations in this paper and not a total order on preferences.

This chapter is structured as follows. Section 4.2 introduces the basic background on two analogical readings that are relevant for applying analogical proportion-based inference to preference prediction. Section 4.3 presents a variety of predicting methods either exploiting a set of given examples or using an extension of this set relying on a monotony assumption of the preferences. These methods exploits triples of pairs of items or takes these pairs one by one for prediction and are first applied to the Boolean feature values then extended to the multi-valued setting.

4.2 Analogy and linear utility

In the following, the items we consider are made of preferences between two vectors of criteria values, of the form $a^1 \preceq a^2$, $b^1 \preceq b^2$, $c^1 \preceq c^2$ and $d^1 \preceq d^2$. Then the basic analogical inference pattern applied to compared items is then, $\forall i \in [[1, n]]$,

$a_i^1 : b_i^1 :: c_i^1 : d_i^1$ and $a_i^2 : b_i^2 :: c_i^2 : d_i^2$

$\vec{a^1} \preceq \vec{a^2}$

$\vec{b^1} \preceq \vec{b^2}$

$\vec{c^1} \preceq \vec{c^2}$

$- - - - - - - - - - - - - - - - -$

$\vec{d^1} \preceq \vec{d^2}$.

where $x \preceq y$ expresses that y is preferred to x (equivalently $y \succeq x$); other patterns (equivalent up to some rewriting) exist where, e.g., $\preceq$ is changed into $\succeq$ for i) pairs (b^1, b^2) and (d^1, d^2), or in ii) pairs (c^1, c^2) and (d^1, d^2) (since $a : b :: c : d$ is stable under central permutation) [5].

Following [49], the above pattern corresponds to a *vertical* reading, while another pattern corresponding to the *horizontal* reading can be stated as follows $\forall i \in [[1, n]]$,

$a_i : b_i :: c_i : d_i$

$\vec{a} \preceq \vec{b}$,

$- - - - - -$

$\vec{c} \preceq \vec{d}$.

The intuition behind the second pattern is simple: since $\vec{a}$ differs from $\vec{b}$ as $\vec{c}$ differs from $\vec{d}$ (and vice-versa), and $\vec{b}$ is preferred to $\vec{a}$, $\vec{d}$ should be preferred to $\vec{c}$ as well. The first pattern, which involves more items and more preferences, states that since the pair of items $(\vec{d^1}, \vec{d^2})$ makes an analogical proportion with the three other pairs $(\vec{a^1}, \vec{a^2}), (\vec{b^1}, \vec{b^2}), (\vec{c^1}, \vec{c^2})$, then the preference relation that holds for the 3 first pairs should hold as well for the fourth one.

Besides, the structure of the first pattern follows the axiomatics of additive utility functions, for which contradictory trade-offs are forbidden, namely: if $\forall i, j$,

$$\vec{a^1}_{-i}\alpha \preceq \vec{a^2}_{-i}\beta$$

$$\vec{a^1}_{-i}\gamma \succeq \vec{a^2}_{-i}\delta$$

$$\vec{c^1}_{-j}\alpha \succeq \vec{c^2}_{-j}\beta$$

one cannot have:

$$\vec{c^1}_{-j}\gamma \prec \vec{c^2}_{-j}\delta$$

where $\vec{x}_{-i}$ denotes the n-1-dimensional vector made of the evaluations of $\vec{x}$ on all criteria except the i^{th} one for which the Greek letter denotes the substituted value. This property ensures that the differences of preference between α and β, on the one hand, and between γ and δ, on the other hand, can consistently be compared.

Thus, when applying the first pattern, one may also make sure that no contradictory trade-offs are introduced by the prediction mechanism. In the first pattern, analogical reasoning amounts here to finding triples of pairs of compared items $(\vec{a}, \vec{b}, \vec{c})$ appropriate for inferring the missing value(s) in $\vec{d}$. When there exist several suitable triples, possibly leading to different conclusions, one may use a majority vote for concluding.

Lastly, let us remark that the second, simpler, pattern agrees with the view that preferences wrt each criterion are represented by differences of evaluations. This includes the weighted sum, namely $\vec{b} \succeq \vec{a}$ iff $\sum_{i=1,n} w_i(b_i - a_i) \geq 0$, while analogy holds at degree 1 iff $\forall i \in [[1, n]], b_i - a_i = d_i - c_i$. This pattern does not agree with more general models of additive utility functions, while the first pattern is compatible with more general preference models.

4.3 Analogy-based Preference Learning

4.3.1 Motivation

The notion of analogical proportions and their formalization has raised a trend of interest in the last two decades [40, 70, 63, 47, 53]. Moreover analogical proportion-based classifiers have been designed and experienced with success [2, 45, 15], first for Boolean and then for nominal and numerical attributes. In this case, the predicted mark is the label of the class. Although it is not intuitively obvious why analogical proportion-based inference may work well, one may notice that such a proportion enforces a parallel between four situations in such a way that the change between a and b is the same as the change between c and d. So this inference exploits co-variations.

One may wonder if what is working in classification may also be applied to preference prediction. The aim of work is to check whether analogical proportions may be a suitable tool for predicting preferences. The problem considered is no longer to predict a class for a new item, but a preference relation between two items on the basis of a set of examples made of known comparisons applying to pairs of items. This set of examples plays the role of a case base, where a case is just a pair of vectors describing the two items together with information saying what item is preferred.

Preference learning has become a popular artificial intelligence topic [28, 24, 35, 29]. Preference learning often relies on the assumption that data sets are massively available. Interestingly enough, analogical proportion-based inference may work with a rather small amount of examples, as we shall see. Preference-learning approaches often rely on the hypothesis that known preferences agree with a unique unknown aggregation function or with a conditional preference structure that has to be identified. Analogical proportion-based methods extrapolate predictions from known cases without looking for some underlying explanation model.

4.3.2 Contributions

In order to study the ability of analogical proportions to predict new preference relations from a given set of such relations, while avoiding the generation of contradictory trade-offs, we propose different "Analogy-based Preference Learning" algorithms (APL algorithms for short). The criteria are assumed to be evaluated on a scale $S = \{1, 2, ..., k\}$. Given a set $E = \{\vec{e^j} : \vec{x^j} \succeq \vec{y^j}\}$ of preference examples, where $\succeq$ is a preference relation telling us that choice $\vec{x^j}$ is preferred to choice $\vec{y^j}$.

Given a new pair of items $\vec{d} \notin E$ for which preference is to be predicted, we present two types of algorithms for predicting preferences in the following, corresponding respectively to the "vertical reading" (first pattern) that exploits $triples$ of pairs of items, and to

"horizontal reading" (second pattern) where pairs of items are taken one by one.

Analogy-based Preference Learning: Boolean setting

A first attempt to treat Analogy-based Preference Learning has been developed in [5] that applies the vertical reading and focus only on the Boolean view of analogical Proportions. In this contribution, two algorithms are proposed that look for triples of preferences appropriate for a prediction. The first one only exploits the given set of examples. The second one completes this set with new preferences deducible from this set under a monotony assumption. This completion is limited to the generation of preferences that are useful for the requested prediction. The predicted preferences should fit with the assumption that known preferences agree with a unique unknown weighted average.

Given a new item $\vec{d} : (\vec{x}, \vec{y})$ whose preference is to be predicted, the basic principle of APL is to find the $good$ triples $(\vec{a}, \vec{b}, \vec{c})$ of examples in E (or if possible in $comp(E)$) that form with $\vec{d}$ either the non-contradictory trade-offs pattern (considered in first), or one of the $three$ analogical proportion-based inference patterns (described in Section 1.2.2 of Chapter 1). Such triples, when applicable, will help to guess the preference of $\vec{d}$ by applying a majority vote on the solutions provided by each of these triples.

Let us consider one of the basic patterns:

$$\vec{a} : x_{-i}\alpha \succeq y_{-i}\beta$$
$$\vec{b} : x_{-i}\gamma \preceq y_{-i}\delta$$
$$\vec{c} : v_{-j}\alpha \succeq w_{-j}\beta$$
$$\vec{d} : v_{-j}\gamma \ ? \ w_{-j}\delta$$

where preference of $\vec{d}$ is unknown.

The APL can be described by this basic process:

- For a given $\vec{d}$, search for good triples in E.
- In case no good triples could be found in E, search for such triples in $comp(E)$.
- Apply a majority vote on the candidate solutions of these good triples to predict the preference of $\vec{d}$.

The process of searching for good triples can be summarized by the following 3 steps:

1. **Find good** $\vec{c}$: In the basic pattern, we can see that the item $\vec{c}$ may be any example in the set E which is identical to $\vec{d}$ except on one criterion that is denoted by its index j. The intuitive idea of APL is to start by searching for the $best$ examples $\vec{c} \in E$ that fit the basic pattern considered. As j may be any index in the set of criteria,

a loop on all possible criteria $j \in \{1, ..., n\}$ should be executed in order to find j. Once a candidate $\vec{c}$ is found, this helps to also fix parameters α, β, γ and δ for the $current$ candidate triple. We save such parameters as $param = \{\alpha, \beta, \gamma, \delta, j\}$.

2. **Find good** $\vec{a}$: Once parameters α and β are fixed for each example $\vec{c}$, it is easy to find a good example $\vec{a} \in E$ in which α and β appears on the same criterion, indexed by i. As in the case of $\vec{c}$, a similar process is to be applied to find such examples $\vec{a}$. This helps to fix a new parameter i and update the set of parameters to be $param = \{\alpha, \beta, \gamma, \delta, j, i\}$.

3. **Find good** $\vec{b}$: As a result of the previous step, to each candidate pair $(\vec{a}, \vec{c})$ along with $\vec{d}$ corresponds a set of candidate parameters $param = \{\alpha, \beta, \gamma, \delta, j, i\}$. The last step is to find all good examples $\vec{b} \in E$ to enclose the triple $(\vec{a}, \vec{b}, \vec{c})$, i.e., those that fit exactly the pattern: $p : x_{-i}\gamma, y_{-i}\delta$ regardless of the sign of the preference relation.

The next step of the APL is to predict preference based on the selected good triples. Each candidate triple helps to predict an atomic preference solution for $\vec{d}$ by inference based on any of the previous patterns described in Section 1.2.2. A global preference solution is computed through a majority vote applied on all atomic solutions and finally assigned to $\vec{d}$.

As expected, the proposed APL may fail in case no examples $\vec{c}$ (or $\vec{a}$, or $\vec{b}$) could be found in the set E especially when E has a limited size. To overcome this problem, we expand the set E and search for examples e in $comp(E)$.

For any example $e \in E$ s.t.: $e : x_{-i}\alpha \succeq y_{-i}\beta$, one may produce a new valid preference example by $dominance$ (monotony) defined as:

$$newe \in comp(E) \text{ iff } newe : newx_{-i}\alpha \succeq newy_{-i}\beta \quad \text{and } newx_{-i} \geq x_{-i} \text{ and } y_{-i} \geq newy_{-i} \quad (4.1)$$

For any relation e with opposite preference sign corresponds a $newe$ by reversing the operators.

Algorithms

Based on the previous ideas, two different algorithms for predicting preferences are proposed. The first alternative is to look at all good triples $(\vec{a}, \vec{b}, \vec{c}) \in E$ that provide a solution for the item $\vec{d}$. This first option is described by Algorithm 5.

In case Algorithm 5 fails to find good triples, the second alternative (described by Algorithm 6) aims at expanding the set of preference examples E by searching for good triples $(\vec{a}, \vec{b}, \vec{c})$ in $comp(E)$. In this set, examples are produced by applying dominance (monotony) on elements found in E.

Algorithm 5 APP with restricted set

Input: a training set E of examples with known preferences
a new item $\vec{d} \notin E$ whose preference is unknown.
$PredictedPref = false$
Preprocess: $S_{(\vec{a},\vec{b})} = FindPairs(E)$.
CandidateVote(p)=0, for each $p \in \{\preceq, \succeq\}$
for each $\vec{c} \in E$ **do**
  if IsGood($\vec{c}$) **then**
    for each $(\vec{a}, \vec{b}) \in S_{(\vec{a},\vec{b})}$ **do**
      if IsGood($\vec{a}$) AND IsGood($\vec{b}$) **then**
        $p = Sol(\vec{a}, \vec{b}, \vec{c}, \vec{d})$
        $CandidateVote(p)$++
      end if
    end for
  end if
end for
$maxi = max\{CandidateVote(p)\}$
if $maxi \neq 0$ AND $unique(maxi, CandidateVote(p))$ **then**
  $Preference(\vec{d}) = argmax_p\{CandidateVote(p)\}$
  $PredictedPref = true$
end if
if $PredictedPref$ **then**
  return$Preference(\vec{d})$
else
  return ($not\ predicted$)
end if

Analogy-based Preference Learning: extension to the Multi-valued setting

The analogy-based Preference Learning approaches that handle nominal or numerical attribute values and deals with the multi-valued setting of AP, defined in Chapter 2, are presented in [6] . In this contribution, two approaches based on analogical proportions are presented and compared to previous works. The first one uses triples of pairs of items for which preferences are known and which make analogical proportions, altogether with the new pair. These proportions express attribute by attribute that the change of values between the items of the first two pairs is the same as between the last two pairs. This provides a basis for predicting the preference associated with the fourth pair, also making sure that no contradictory trade-offs are created. Moreover, we also consider the option that one of the pairs in the triples is taken as a k-nearest neighbor of the new pair. The second approach exploits pairs of compared items one by one: for predicting the preference between two items, one looks for another pair of items for which the preference is known such that, attribute by attribute, the change between the elements of the first pair is the same as between the elements of the second pair. The two approaches agree with

Algorithm 6 APP with a completion set

Input:a training set E of examples with known preferences
a new item $\vec{d} \notin E$ whose preference is unknown.
Preprocess: $S_{(\vec{a},\vec{b})} = FindPairs(E)$.
if Algo1($\vec{d}, E$) $= not\ predicted$ **then**
  CandidateVote(p)=0, for each $p \in \{\preceq, \succeq\}$
  for each $\vec{c} \in E$ **do**
    $new_{\vec{c}} = comp(\vec{c})$
    for each $(\vec{a}, \vec{b}) \in E \times E$ **do**
      if IsGood($\vec{a}$) AND IsGood($\vec{b}$) **then**
        $p = Sol(\vec{a}, \vec{b}, new_{\vec{c}}, \vec{d})$
        $CandidateVote(p)$++
      end if
    end for
    if CandidateVote(p)=0, for each $p \in \{\preceq, \succeq\}$ **then**
      for each $\vec{a} \in E$ **do**
        $new_{\vec{b}} = comp(\vec{b})$
        if IsGood($\vec{a}$) AND IsGood($new_{\vec{b}}$) **then**
          $p = Sol(\vec{a}, new_{\vec{b}}, new_{\vec{c}}, \vec{d})$
          $CandidateVote(p)$++
        end if
      end for
    end if
  end for
end if
$Preference(\vec{d}) = argmax_p\{CandidateVote(p)\}$
return $Preference(\vec{d})$

the postulates underlying weighted averages and more general multiple criteria aggregation models. In this contribution, two new algorithms implementing these methods are suggested and tested on a variety of datasets.

APL_3**:** The basic principle of APL_3 is to find triples $t(\vec{a}, \vec{b}, \vec{c})$ of examples in E^3 that form with $\vec{d}$ either the non-contradictory trade-offs pattern (considered in first), or the analogical proportion-based inference pattern. For each triple $t(\vec{a}, \vec{b}, \vec{c})$, we compute an analogical score $A_t(\vec{a}, \vec{b}, \vec{c}, \vec{d})$ that estimates the extent to which it is in analogy with the item $\vec{d}$ using Formula 2.1 in Chapter 2, where P refers to definition A of AP in this context. Then to guess the final preference of $\vec{d}$, for each possible solution, we first cumulate these atomic scores provided by each of these triples in favor of this solution and finally we assign to $\vec{d}$ the solution with the highest score. In case of ties, a majority vote is applied.

The APL_3 can be described by this basic process:

- For a given $\vec{d}$ whose preference is to be predicted.

- Search for solvable triples $t \in E^3$ that make the analogical proportion, linking the 4 preference relations of the triple elements with $\vec{d}$, valid (the preference relation between the 4 items satisfy one of the vertical pattern given in Section 4.2).
- For each triple t, compute the analogical score $A_t(\vec{a}, \vec{b}, \vec{c}, \vec{d})$.
- Compute the sum of these scores for each possible solution for $\vec{d}$ and assign to $\vec{d}$, the solution with the highest score.

APL_1**:**

Applying the "horizontal reading" (second pattern), we consider $only$ one item $\vec{a}$ at a time and apply a comparison with $\vec{d}$ in terms of pairs of vectors rather than comparing simultaneously 4 preferences, as with the first pattern. From a preference $\vec{a} : \vec{a^1} \succeq \vec{a^2}$ such that $(\vec{a^1}, \vec{a^2}, \vec{d^1}, \vec{d^2})$ is in analogical proportion, one extrapolates that the $same$ preference still holds for $\vec{d} : \vec{d^1} \succeq \vec{d^2}$. A similar process is applied in [26] that they called *analogical transfer* of preferences.

Following this logic, for each item $\vec{a}$ in the training set, an analogical score $A(\vec{a^1}, \vec{a^2}, \vec{d^1}, \vec{d^2})$ is computed. As in case of the vertical reading, these atomic scores are accumulated for each possible solution for $\vec{d}$ (induced from items $\vec{a}$). Finally, the solution with the highest score is assigned to $\vec{d}$.

The APL_1 can be described by this basic process:

- For a given $\vec{d} : \vec{d^1}, \vec{d^2}$ whose preference is to be predicted.
- For each item $\vec{a} \in E$, compute the analogical score $A(\vec{a^1}, \vec{a^2}, \vec{d^1}, \vec{d^2})$.
- Compute the sum of these scores for each possible solution for $\vec{d}$ and assign to $\vec{d}$, the solution with the highest score.

Algorithms

Based on the above ideas, we propose two different algorithms for predicting preferences. Let E be a training set of examples whose preference is known. Algorithms 7 and 8 respectively describe the two previously introduced procedures APL_3 and APL_1. Note that in Algorithm 7, to evaluate the analogical score $A_t(\vec{a}, \vec{b}, \vec{c}, \vec{d})$ for each triple t, we choose to consider all the possible arrangements of items $\vec{a}$, $\vec{b}$ and $\vec{c}$, i.e., for each item $\vec{x}$, both $\vec{x} : \vec{x^1} \preceq \vec{x^2}$ and $\vec{x'} : \vec{x^2} \succeq \vec{x^1}$ are to be evaluated. The function $FindCandidateTriples(t)$ helps to find such candidate triples. Since we are dealing with triples in this algorithm, 2^3 candidate triples are evaluated for each triple t. The final score for t is that corresponding to the $best$ score among its candidate triples. In both Algorithms 7 and 8, $P(\vec{x})$ returns the preference sign of the preference relation for $\vec{x}$.

For APL_3, we also consider another alternative in order to drastically reduce the number of triples to be investigated. This alternative follows exactly the same process described by Algorithm 7 except one difference: instead of systematically surveying E^3, we restrict the search for solvable triples $t(\vec{a}, \vec{b}, \vec{c})$ by constraining $\vec{c}$ to be one of the k-nearest neighbors of $\vec{d}$ w.r.t. Manhattan distance (k is a parameter to be tuned). This option allows us to decrease the complexity of APL_3 that become quadratic instead of being cubic. We denote $APL_3(NN)$ the algorithm corresponding to this alternative.

Algorithm 7 $APL3$

Input: a training set E of examples with known preferences
a new item $\vec{d} \notin E$ whose preference $P(\vec{d})$ is unknown.
SumA(p)=0 for each $p \in \{\preceq, \succeq\}$
$BestA_t$=0, $S = \emptyset$, $BestSol = \emptyset$
for each triple $t = (\vec{a}, \vec{b}, \vec{c})$ in E^3 **do**
  S = FindCandidateTriples(t)
  for each candidate triple $ct = (\vec{a'}, \vec{b'}, \vec{c'})$ in S **do**
    if $(P(\vec{a'}) : P(\vec{b'}) :: P(\vec{c'}) : x$ has solution $p)$ **then**
      $A_t = Min(A(a'_1, b'_1, c'_1, d_1), A(a'_2, b'_2, c'_2, d_2))$
      if $(A_t > BestA_t)$ **then**
        $BestA_t = A_t$
        $BestSol = Sol(ct)$
      end if
    end if
  end for
  $SumA(BestSol) += BestA_t$
end for
$maxi = max\{SumA\}$
if $(maxi \neq 0)$ **then**
  if $(unique(maxi, SumA))$ **then**
    $P(\vec{d}) = argmax_p\{SumA\}$
  else
    Majority vote
  end if
else
  No Prediction
end if
return $P(\vec{d})$

4.3.3 Experimental validation

To evaluate the proposed APL algorithms in the Boolean or numerical setting, we have developed an experimental study based on five datasets, the two first ones are synthetic data generated from different functions: weighted average, Tversky's additive difference and Sugeno Integral described in the following.

Algorithm 8 $APL1$

Input: a training set E of examples with known preferences
a new item $\vec{d} \notin E$ whose preference $P(\vec{d})$ is unknown.
SumA(p)=0 for each $p \in \{\preceq, \succeq\}$
$BestA$=0, $BestSol = \emptyset$
for each $\vec{a}$ in E **do**
  $BestA = max(A(a_1, a_2, d_1, d_2), A(a_2, a_1, d_1, d_2))$
  if $(A(a_1, a_2, d_1, d_2) > A(a_2, a_1, d_1, d_2))$ **then**
    $BestSol = P(\vec{a})$
  else
    $BestSol = notP(\vec{a})$
  end if
  $SumA(BestSol) += BestA$
end for
$maxi = max\{SumA\}$
if $(maxi \neq 0)$ **then**
  if $(unique(maxi, SumA))$ **then**
    $P(\vec{d}) = argmax_p\{SumA\}$
  else
    Majority vote
  end if
else
  No Prediction
end if
return $P(\vec{d})$

- **Datasets 1**: we consider only 3 criteria in each preference relation i.e., $n = 3$. We generate different type of datasets:

  1. Examples in this dataset are first generated using a weighted average function (denoted WA in Table 4.1) with $0.6, 0.3, 0.1$ weights respectively for criteria 1, 2 and 3.
  2. The second artificial dataset (denoted TV in Table 4.1) is generated using a Tversky's additive difference model [65], i.e. an alternative a is preferred over b if $\sum_{i=1}^{n} \Phi_i(a_i - b_i) \geq 0$, where Φ_i are increasing and odd real-valued functions. For generating this dataset, we used the piecewise linear functions described in [6].
  3. Then, we generate this dataset using weighted $\max$ and weighted $\min$ which are particular cases of Sugeno integrals, namely using the aggregation functions defined as follows:

$$S_{Max} = \max_{i=1}^{n}(\min(v_i, w_i)),$$

$$S_{Min} = \min_{i=1}^{n}(\max(v_i, 6 - w_i)),$$

where v_i refers to the value of criterion i and w_i represents its weight. In this case, we tried two different sets of weights : $w_1 = 5, 4, 2$ and $w_2 = 5, 3, 3$, respectively for criteria 1, 2 and 3.

- **Datasets 2**: we expand each preference relation to support 5 criteria, i.e: $n = 5$. We apply the weights $0.4, 0.3, 0.1, 0.1, 0.1$ in case of weighted average function and $w_1 = 5, 4, 3, 2, 1$ and $w_2 = 5, 4, 4, 2, 2$ in case of Sugeno integral functions. For generating the second dataset (TV), we used the following piecewise linear functions given in [6]. For the two datasets, weights are fixed on a empirical basis, although other choices have been tested and have led to similar results.

For both datasets, each criterion is evaluated on a scale with 5 levels, i.e., $S = \{1, ..., 5\}$.

To check the applicability of APL algorithms, we also evaluate their efficiency on real data. We consider the following three datasets.

- The **Food dataset** (https://github.com/trungngv/gpfm) contains 4036 user preferences among 20 food menus picked by 212 users. Features represent 3 levels of user hunger; the study is restricted to 5 different foods.
- The **University dataset** (www.cwur.org) includes the top 100 universities from the world for 2017 with 9 numerical features such as national rank, quality of education, etc.
- The **Movie-Lens dataset** (https://grouplens.org) includes users responses in a survey on how serendipitous a particular movie was to them. It contains 2150 user preferences among different movies picked by different users.

Results of the Boolean setting

Figures 4.1 and 4.2 show prediction accuracies of APL algorithms in case of Boolean setting respectively for Datasets 1 and 2 for different sizes of each dataset and different weights (see curves "Algo1_$w_i(E)$" and "Algo2_$w_i(E)$"; the used sets of weights and other curves using $InterE$ data are defined in [5]).

The previous results show the effectiveness of Algorithm 2 as preference predictor which fits with a given weighted average used to produce the preference examples especially for small number of criteria.

It is worth pointing out that the predicted preferences have been evaluated as being valid, or not, on the basis of 3 weighted averages, the ones used for generating the dataset with its three versions. It is clear that a given set E of preference examples is compatible

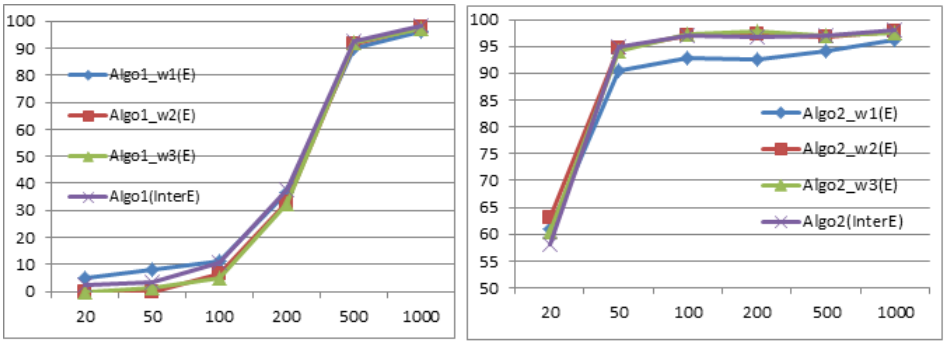

Figure 4.1: Prediction accuracies for Dataset 1 with 3 criteria and different sizes of subsets of data.

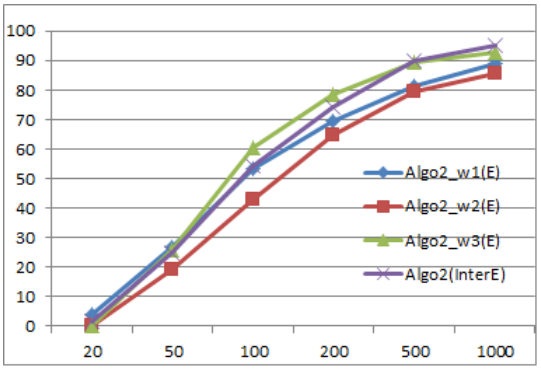

Figure 4.2: Prediction accuracies for Dataset 2 with 5 criteria and different dataset sizes

with a more large collections of weights. Strictly speaking a prediction is valid if it is correct with respect to at least one of the collections of weights compatible with E. As already said, determining all the extreme points of the polytope of the weights compatible with E is quite tricky. So in the above reported experiments, we have compared the prediction to ones obtained by using the weighted averages used in the generation of the training set, and thus the reported accuracies are *lower bounds* of the true accuracies.

Results of the multi-valued setting

Tables 4.1 and 4.2 provide prediction accuracies respectively for synthetic and real datasets for the three proposed APL algorithms as well as Algorithm 1 described in [26] (denoted here "FH18") in the multi-valued setting. The best accuracies for each dataset size are highlighted in bold.

Results in Table 4.1 and 4.2 show that:

- For synthetic data and in case of datasets generated from a weighted average, it is

Table 4.1: Prediction accuracies for Dataset 1 and Dataset 2

Data	**Size**		**APL3**	**APL3(NN)**	k^*	**APL1**	**FH18**	N^*
D1	50	WA	92.4±11.94	89.0±12.72	22	**92.9±11.03**	90.6±12.61	14
		TV	87.8±12.86	88.6±12.71	19	86.8±14.28	**90.4±12.61**	11
		SMax-W1	90.0±12.24	88.8±12.01	19	**90.2±11.66**	88.0±13.02	15
		SMax-W2	91.2±12.91	88.4±13.59	20	**95.8±7.92**	90.2±11.62	15
		SMin-W1	**90.0±13.24**	89.4±11.97	17	88.6±13.18	86.4±13.76	16
		SMin-W2	93.2±10.37	89.6±12.61	20	**94.2±9.86**	92.4±11.48	18
	100	WA	**94.75±6.79**	91.55±8.51	24	93.85±7.51	93.5±7.09	15
		TV	89.6±9.43	86.5±10.41	21	88.0±9.6	**92.1±8.45**	15
		SMax-W1	**93.4±6.5**	91.2±9.31	25	93.2±6.98	91.2±9.14	17
		SMax-W2	93.8±7.17	88.8±8.08	25	**95.3±5.66**	92.9±8.2	15
		SMin-W1	90.4±9.66	89.6±8.57	19	**90.6±9.43**	89.4±8.73	16
		SMin-W2	94.3±6.56	90.1±7.68	21	**96.3±5.09**	94.6±5.68	14
	200	WA	95.25±4.94	92.25±6.28	23	**95.4±4.33**	94.55±5.17	13
		TV	91.25±5.48	91.7±5.93	25	90.1±5.18	**95.1±4.53**	13
		SMax-W1	90.0±5.98	89.3±6.73	24	**90.2±6.2**4	89.4±6.38	14
		SMax-W2	95.7±3.28	92.6±5.35	26	**97.2±2.71**	95.6±4.08	15
		SMin-W1	**92.0±7.17**	91.1±5.86	24	**92.0±5.82**	90.3±6.38	18
		SMin-W2	94.8±4.88	91.55±5.15	26	**97.4±3.35**	95.0±4.45	16
D2	50	WA	**88.3±12.41**	86.2±14.43	20	84.9±15.17	83.5±14.99	15
		TV	**89.4±15.08**	86.0±16.46	17	89.2±14.85	87.6±14.44	17
		SMax-W1	**86.6±13.77**	83.8±14.14	18	86.4±12.18	83.2±15.53	16
		SMax-W2	85.8±15.77	82.4±15.44	20	**87.0±14.58**	81.2±15.7	14
		SMin-W1	**86.6±14.34**	86.2±13.45	23	85.6±14.61	84.0±14.57	16
		SMin-W2	88.8±11.5	86.2±14.29	22	**89.0±11.94**	83.6±14.78	17
	100	WA	**92.0±7.51**	89.0±8.89	22	90.0±8.22	88.3±9.71	15
		TV	90.2±8.18	88.1±8.93	18	**91.4±8.03**	86.8±9.06	15
		SMax-W1	88.0±9.52	87.9±9.56	21	**88.8±9.31**	85.4±10.7	16
		SMax-W2	87.7±9.17	86.1±9.77	24	**90.1±9.73**	85.1±9.86	17
		SMin-W1	89.1±9.15	88.6±10.47	21	**90.2±9.3**	85.6±9.98	16
		SMin-W2	87.4±9.7	83.2±11.99	24	**89.2±9.27**	84.8±10.06	16
	200	WA	**94.7±5.11**	90.2±6.01	24	92.3±5.2	90.0±6.08	17
		TV	91.1±6.33	89.0±6.54	27	**91.9±6.06**	89.65±6.81	16
		SMax-W1	89.15±6.76	88.65±6.58	22	**89.7±6.84**	86.5±7.66	17
		SMax-W2	89.4±6.45	88.15±6.77	27	**91.7±5.65**	86.65±6.71	16
		SMin-W1	90.7±5.79	89.25±6.45	24	**91.3±5.1**	88.8±6.55	15
		SMin-W2	89.75±5.64	88.0±6.33	23	**91.55±5.68**	87.8±6.68	16

clear that APL_3 achieves the best performances for almost all dataset sizes. APL_1 is just after. Note that these two algorithms record all triples/items analogical scores for prediction. We may think that it is better to use all the training set for prediction to be compatible with weighted average examples.

- In case of datasets generated from a Sugeno integral, $APL1$ is significantly better than other algorithms for most datasets sizes and for the two weights W_1 and W_2.

- If we compare results of the three types of datasets: the one generated from a weighted average, from Tversky's additive difference or from a Sugeno integral, globally, we can see that the accuracy obtained for a Sugeno integral dataset is the best in case of datasets with 3 criteria (see for example $APL1$). For datasets with 5 criteria, results obtained on weighted average datasets are better than on the two others. While results obtained for Tversky's additive difference datasets seem less

Table 4.2: Prediction accuracies for real datasets

Dataset	**Size**	**APL3**	**APL3(NN)**	k^*	**APL1**	**FH18**	N^*
Food	200	61.3±8.32	**63.0±9.64**	15	61.05±9.34	57.55±10.41	13
	1000	-	**73.16±3.99**	20	63.11±5.0	63.11±5.54	20
Univ.	200	73.6±9.67	**80.0±8.03**	14	73.6±8.47	75.7±8.29	12
	1000	-	**87.9±3.04**	17	76.76±3.86	83.74±3.26	12
Movie	200	51.9±14.72	49.1±15.2	19	**52.93±13.52**	48.61±14.26	15
	1000	-	**55.06±4.51**	23	54.48±4.7	53.38±5.32	10

accurate in most cases.

- For real datasets, it appears that $APL_3(NN)$ is the best predictor for most tested datasets. To predict user preferences, rather than using all the training set for prediction, we can select a set of training examples, those where one of them is among the k-nearest neighbors.
- $APL_3(NN)$ seems less efficient in case of synthetic datasets. This is due to the fact that synthetic data is generated randomly and applying the NN-approach is less suitable in such cases.
- If we compare APL algorithms to Algorithm1 "FH18", we can see that APL_3 outperforms the latter in case of synthetic datasets. Moreover, $APL_3(NN)$ is better than "FH18" in case of real datasets.
- APL algorithms achieve the same accuracy as Algorithm 1 in [5] with a very small dataset size (for the dataset with 3 criteria built from a weighted average, only 200 examples are used by APL algorithms instead of 1000 examples in [5] to achieve the best accuracy: See the left side of Figure 4.1 for results without completion). The two algorithms have close results for the Food dataset.
- Comparing $vertical$ and $horizontal$ approaches, there is no clear superiority of one view (for the tested datasets).

In Tables 4.3 and 4.4, we compare the best results obtained with our algorithms to the accuracies obtained by finding the weighted sum that best fits the data in the learning sets. The weights are found by using linear programming as explained in [6].

Table 4.3 displays the results obtained using the synthetic datasets generated according to Tverski's model. The results for datasets generated by a weighted average or a Sugeno integral are not reproduced in this table because the weighted sum (WSUM) almost always reaches an accuracy of 100%. Only in three cases on thirty, its accuracy is slightly lower, with a worst performance of 97.5%. This is not unexpected for datasets generated by means of a weighted average, since WSUM is the right model in this case. It is more surprising for data generated by a Sugeno integral (even if we have only dealt here with particular cases), but we get here some empirical evidence that the Sugeno integral can be well-approximated by a weighted sum. The results are quite different for datasets generated by Tversky's model. WSUM shows the best accuracy in two cases; APL1 and APL3,

also in two cases each. Tversky's model does not lead to transitive preference relations, in general, and this may be detrimental to WSUM that models transitive relations.

Table 4.3: Prediction accuracies for artificial datasets generated by the Tverski model

Dataset	Size	APL3	APL1	WSUM
TV (3 features)	50	**87.8± 12.86**	86.8 ± 14.28	82.00 ± 17.41
	100	89.6±9.43	88.0 ± 9.6	**93.00 ± 8.23**
	200	81.25±5.48	90.1 ± 5.18	**91.00 ± 6.43**
TV (5 features)	50	**89.4 ± 15.08**	89.2 ± 14.85	84.00 ± 15.78
	100	90.2 ± 8.18	**91.4 ± 8.03**	87.00 ± 9.49
	200	91.1 ± 6.33	**91.9 ± 6.06**	85.50 ± 6.53

Table 4.4 compares the accuracies obtained with the real datasets. WSUM yields the best results for all datasets except for the "Food" dataset, size 1000.

Table 4.4: Prediction accuracies for real datasets

Dataset	Size	APL3(NN)	APL1	WSUM
Food	200	63.0±9.64	61.05±9.34	**64.00 ± 20.11**
	1000	**73.16±3.99**	63.11±5.0	61.10 ± 10.19
Univ.	200	80.0±8.03	73.6±8.47	**99.50 ± 1.58**
	1000	87.9±3.04	76.76±3.86	**88.70 ± 21.43**
Movie	200	49.1±15.2	52.93±13.52	**69.50 ± 18.77**
	1000	55.06±4.51	54.48±4.7	**77.60 ± 16.93**

These examples suggest that analogy-based algorithms may surpass WSUM in some cases. However, the type of datasets for which it takes place is still to be determined.

4.3.4 Scientific impact

This research has been carried out in the context of the Master thesis of Toumather Nesibi and has been published in two international conferences (see [5, 6]).

4.4 Conclusion

The results presented in the previous section confirm the interest of considering analogical proportions for predicting preferences, which was the primary goal of this work since such an approach has been proposed only recently. We observed that analogical proportions yield a better accuracy as compared to a weighted sum model for certain datasets (TV, among the synthetic datasets and Food, as a real dataset). Determining for which datasets this tends to be the case requires further investigation.

Analogical proportions may be a tool of interest for creating artificial examples that are useful for enlarging a training set, see, e.g., [7]. It would be worth investigating to see if such enlarged datasets could benefit to analogy-based preference learning algorithms as well as to the ones based on weighted sum.

Chapter 5

Discussions and future works

5.1 Introduction

Comparing objects or situations and identifying in what respects they are identical (or similar) and in what respects they are different, is a basic type of operations at the core of many intelligent activities. In particular, proportions are a matter of comparison between pairs of objects or situations, where a comparison has already been done inside the pairs.

It is only in the last decade that *analogical* proportions, i.e., statements of the form a is to b as c is to d, where each item refers to a situation described by a vector of feature values, have been formalized first in terms of subsets of properties that hold true in a given situation [40, 63], and then in a logical manner [47]. Quite early, it was shown that a formal view of analogical proportions may be the basis of a new type of classifier that performs well on some difficult benchmarks [2, 45]. This was confirmed by other implementations directly based on a logical view of analogical proportions [8].

Besides, it was shown that analogical proportions belong to a larger family of so-called logical proportions that relate a 4-tuple of Boolean variables [50], where the 8 code-independent logical proportions are of particular interest since their truth status remain unchanged if a property is encoded positively or negatively. These 8 logical proportions divide into 4 *homogeneous* proportions, which include the analogical proportion and 3 related proportions, and 4 *heterogeneous* proportions [54]. An heterogeneous proportion expresses the idea that there is an intruder among the 4 truth values, which is forbidden to appear in a specific position.

This habilitation was devoted to study the merits of *homogeneous* (especially analogical) and *heterogeneous* proportions and their applications in two different domains in AI: Classification and Preference Learning.

The rest of this chapter is organized as follows: Section 5.2 summarizes the main

contributions related to classification and preference learning problems. A discussion regarding the proposed approaches as well as their limits are detailed in Section 5.3. Section 5.4, discusses the possible extensions and new applications of logical proportions in the future works.

5.2 Main contributions

5.2.1 First application of Logical Proportions: Classification

This section summarizes our first contribution in this habilitation devoted to the first application of logical proportions (homogeneous and heterogenous ones) to classification problems.

Analogy-based classification

In the first part of this habilitation, we have investigated the study of Analogical Proportions and their efficiency in the classification task. Analogical proportions have been, in particular, formalized in Boolean, nominal and numerical settings. In all cases if the analogical proportion holds, one of the fourth components of the proportion can be computed from the three others.

Based on this analogical inference, we have proposed diverse classification approaches in the first part of this habilitation. Analogical classifiers look for all triples of examples in the sample set that are in analogical proportion with the item to be classified on a maximal number of attributes and for which the corresponding analogical proportion equation on the class has a solution. To classify a new item, we specially emphasize an approach where the whole set of triples that can be built from the sample set is not considered. We just focus on a small part of the candidate triples. Namely, in order to restrict the scope of the search, we first look for examples that are as similar as possible to the new item to be classified. We then only consider the pairs of examples presenting the same dissimilarity as between the new item and one of its closest neighbors. Then the classification is made on the basis of an additive aggregation of the truth values corresponding to the pairs that can be analogically associated with the pairs made of the target item and one of its nearest neighbors. We then only deal with pairs leading to a solvable analogical equation for the class.

The proposed classification approach provides results as good as previous analogical classifiers with a lower average complexity, both in nominal and numerical cases.

Oddness-based classification

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.

In this second part of this habilitation, we first propose two viewpoints for estimating to what extent a new item, described in terms of binary-valued features, fits with a set of existing items. They are respectively based on an oddness index and an evenness index, which in spite of their names, are not exactly the opposite of each other. Both indicators, which refer to one feature, are built from heterogeneous logical proportions, and involve four items, the new item and three others. Heterogeneous proportions express that there is an intruder among four truth values, which is forbidden to appear in a specific position. Global oddness and evenness functions of an item with respect to a set are built from the corresponding indexes by taking all features into account, and then by considering all triples of items in the set.

Moreover the oddness function naturally extends to numerical features and to subsets of items of different sizes (pairs, triples, etc.). Simple classification procedures, based on these global functions, have been developed: a new item is assigned to the class that minimizes oddness or maximizes evenness.

The previous idea, dealing with pairs, have been then confirmed and refined in different ways: First, 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. Second, we 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.

Experimental results of oddness-based classifiers on a set of UCI benchmarks show that they are still competitive with regard to state of the art classifiers (k-N N, SVM) while having drastically decreased the complexity.

5.2.2 Second application of Analogical Proportions: Preference Learning

In the last part of this habilitation, our interest was to study the ability of Analogical Proportions as a tool for predicting user's preference. Given a set of preferences between items taken by pairs and described in terms of nominal or numerical attribute values, Preference Learning aims to predict the preference between the items of a new pair.

In this context, we have proposed and compared two basic preference learning approaches based on analogical proportions. The first one uses triples of pairs of items for which preferences are known and which make analogical proportions, altogether with the new pair. This provides a basis for predicting the preference associated with the fourth pair, also making sure that no contradictory trade-offs are created. Moreover, we also consider the option that one of the pairs in the triples is taken as a k-nearest neighbor of the new pair. The second approach exploits pairs of compared items one by one: for predicting the preference between two items, one looks for another pair of items for which the preference is known such that, attribute by attribute, the change between the elements of the first pair is the same as between the elements of the second pair. As discussed in Chapter 4, the two approaches agree with the postulates underlying weighted averages and more general multiple criteria aggregation models. The reported experiments, both on real data sets and on generated datasets suggest the effectiveness of the approaches.

5.3 Discussions

5.3.1 Logical Proportions-based Classification

On the basis of the investigation of AP-classifier or Oddness/Evenness-based classifiers, we may globally derive the following conclusions:

On the first hand, the AP-classifier achieves a reduced complexity when compared to previous analogical classifiers taking into account all triples, while maintaining an accuracy statistically better than the k-NN and in many case equivalent to SVM. Nevertheless, the execution time of the AP-classifier remains still high due to the exploration of triples of examples during the testing phase. It would remain to find classes of problems where AP-classifiers are of particular interest. It may be the case when we have relatively few data at hand. However, it is worth pointing out that the AP-classifier relies on ideas quite different from the other existing classifiers, while providing results of similar quality.

As explained in Chapter 2, a rather small number of pair-based predictors are used to classify. It remains to investigate if this is a general property and if it is possible to obtain accurate results by focusing only on a still more restricted number of predictors. Such an approach might be closer to a cognitive attitude where excellent human experts usually focus directly on the *few* relevant pieces of information for making prediction.

This first work is quite experimental and a complete theoretical investigation has still to be done. [34, 22] have started this investigation. Nevertheless, some questions remain unanswered: for instance, what kind of theoretical accuracy could be expected from analogical classifiers? How well analogical classifiers deal with noise? Is it possible to statically determine the relevant triples to be used for classification? Taking inspiration of a similar work done for k-NN [48, 37], we may try to provide answers to some of these

questions in the future.

On the second hand and apart from a formal investigation of the properties of oddness and evenness measures, their merits would need to be studied in greater detail in order, for instance, to more precisely assess the expected accuracy of oddness or evenness-based classifiers. Then, one might think to use them in conformal predictors [62, 67, 66], as first experimented in [12] with an evenness-based classifier. Indeed the oddness measure may be considered as a non conformity measure, while the evenness measure would be a conformity measure. Moreover, as datasets become bigger and bigger, the scalability of analogy-based and oddness-based classifiers is paramount which has to be further investigated in future works.

5.3.2 Analogical Proportions-based Preference Learning

The reported experiments, both on real and generated datasets suggest the effectiveness of Analogical Proportions for Preference Learning in a similar way as in classification. However, many remaining issues should be addressed; for example determining the characteristic of the datasets for which Preference Learning approaches perform better requires further investigation.

Moreover in the proposed Preference Learning approach, the experiments have been on training sets generated by means of weighted sums, which is a quite standard aggregation function, and we have obtained good results for rather small subsets of examples. Still it is known that the representation of multiple-criteria preferences may require more general settings such as Choquet integrals [32] where the condition for avoiding contradictory trade-offs is weaker. Adapting the proposed approach to guess preferences generated by such more general settings is a topic for further research.

5.4 Research in progress and future works

Analogical proportions have proven to be a valuable tool for diverse application domains in the last decade such as morphological linguistic analysis [64, 38], solving IQ tests such as Raven progressive matrices [58, 21] as well as in classification and Preference Learning tasks. It would be interesting to find new domains for which Analogical Proportions may be suitably applied.

5.4.1 Future research axe 1: Data expansion

Analogical proportions are not only a tool for classification, but more generally for building up a fourth item starting from three others, thanks to the equation solving process.

As we have seen, this fourth item could be entirely new. Thus, while classifiers like k-NN focus on the neighborhood of the target item, analogical classifiers go beyond this neighborhood, and rather than "copying" what emerges among close neighbors, "take inspiration" of relevant information possibly far from the immediate neighborhood.

Some previous works have considered, discussed and experimented the idea of an analogical proportion-based enlargement of a training set, based on triples of examples. In [3], the authors proposed an approach to generate synthetic data to tune a handwritten character classifier. Couceiro et al. [22] presented a way to extend a Boolean sample set for classification using the notion of "analogy preserving" functions that generate examples on the basis of triples of examples in the training set. The authors only tested their approach on Boolean data.

As an attempt to tackle such problem, we have recently developed a new idea of enlarging a training set using analogical proportions [7] as the work in [22], with two main differences: we only consider pairs of examples to predict new examples not in the training set by using *continuous* analogical proportions which contribute to reduce the complexity to be quadratic instead of cubic, and we test with ordered nominal datasets instead of Boolean ones.

In this last work [7], results obtained by classical machine learning methods such as k-NN on the enlarged training set generally improve those obtained by applying these methods to the original training sets. On the other hand, these results, obtained with a smaller level of complexity, are often not so far from those obtained by directly applying the analogical proportion-based classification method on the original training set [15]. It remains to extend this new method to enlarge a training set having numerical attribute values which is a work in progress.

As discussed above, analogical proportions seems to be a tool of interest for creating artificial examples that are useful for enlarging a training set. It would be worth investigating to see if such enlarged datasets could benefit to analogy-based preference learning algorithms as well as to the ones based on weighted sum. Especially in some domain for which the available data remains scarce.

5.4.2 Future research axe 2: Continuous Analogy-based classification

As noted in sub-section 5.3.1, the overall complexity of AP-classifier is still cubic due to the use of triples during the classification phase which requires more execution time in case of large datasets. This is what motivates us to develop a new family of classifiers still based on analogical proportions but in a different way.

Given a data set, described in terms of discrete, ordered attribute values, we are actually studying the efficiency of simpler classifier that searches for pairs instead of triples during

the classification phase by using $contineous$ analogical proportions which are statements of the form: a is b as b is to c, i.e: b can be seen as a midpoint of a and c. Let $V = \{v_1, \cdots, v_k\}$ be an ordered set of nominal values, then, v_i will be regarded as the midpoint of v_{i-j} and v_{i+j} with $j \geq 1$, provided that both v_{i-j} and v_{i+j} exist. For instance, if $V = \{1, \cdots, 5\}$, the analogical proportions $1 : 3 :: 3 : 5$ or $2 : 3 :: 3 : 4$ hold, while $2 : x :: x : 5 = 1$ has no solution. In this work, we aim to show that there is an alternative way for constructing analogical classifiers to handle discrete ordered data. The proposed approach deals with pairs rather than triples of examples since only items a and c are needed to predict b for example (which helps to significantly reduce the execution time).

This last idea has been recently studied and tested with success within the master thesis of Marouane Essid who proposed a set of new AP classifiers that led to encouraging results close to that of AP classifier [15] while reducing the complexity to be quadratic instead of cubic.

5.4.3 Future research axe 3: Classification versus Preference Learning: a discussion

It is also interesting to study the relationship between classification and Preference Learning problems. This may help to develop a new formalism that take into consideration the characteristics of both domains and generate a kind of generic approach able to tackle both problems similarly. The intuition behind this statement starts by noting that Preference Learning may be viewed as just a particular case of classification. In fact, it can easily be noticed that a preference relation, denoted $x : x^1 \succeq x^2$ s.t: $x^1 = (x_1^1, ..., x_n^1)$ and $x^2 = (x_1^2, ..., x_n^2)$, can be coded as a new instance x^* for classification whose attributes are simply the values of the two vectors x^1 and x^2 made in order and its class label is a Boolean value equal to 1 (resp. 0) in case the preference relation is $\succeq$ (resp. $\preceq$) namely: $x^* = \{x_1^1, ..., x_n^1, x_1^2, ..., x_n^2, 1\}$. It remains to check if the new transformed dataset satisfy the non-contradictory trade off property then evaluate the efficiency of any AP classifier to classify such new transformed dataset.

5.4.4 Future research axe 4: Analogical Proportions and Information retrieval

We have also recently investigated a preliminary study to apply analogy between queries and documents for Information Retrieval (IR). For this end, we proposed a new analogical inference to evaluate document relevance for a given query. Based on this inference process, a new matching model for Information Retrieval has been developed and discussed [4]. Experiments carried out on three IR Glasgow test collections highlight the effectiveness of the model if compared to the known efficient Okapi IR model. A deeper investigation on the merits of such matching model as well as its application on a variety

of test collections have still to be performed.

5.4.5 Future research axe 5: Analogical Proportions and Arabic Text Mining

Based on these preliminary results obtained in IR, we also aim to tackle new related-application domains and we have just started a new project on Arabic Text Mining using Analogical Proportions within a PHD proposal which is a work in progress.

Finally, one may think to extend such developed analogical models to support similar/close problems by adaptation. This process may be seen as a king of Transfer Learning in which it is required to save the main conclusions or knowledge while solving one problem and then apply it to solve a new but related problem.

Appendix A: Personal publications and submissions

1.1 Journal papers with impact factor

1. M. Bounhas, H. Prade, and G. Richard. Analogy-based classifiers for nominal or numerical data. Int. J. Approx. Reasoning, 91:36-55, 2017.
2. M. Bounhas, H. Prade, and G. Richard. Oddness/evenness-based classifiers for boolean or numerical data. Int. J. Approx. Reasoning, 82:81-100, 2017.
3. M. Bounhas, H. Prade, and G. Richard. Oddness-based classification:a new way of exploiting neighbors. Int. J. Int. Systems, 33(12):2379–2401, 2018.

1.2 International conferences

1. M. Bounhas, H. Prade, and G. Richard. Analogical classification: A new way to deal with examples. In Proc. 21st Europ. Conf. on Artificial Intelligence (ECAI'14), Aug. 18-22, Prague, volume 263 of Frontiers in Artificial Intelligence and Applications, pages 135-140. IOS Press, 2014.
2. M. Bounhas, H. Prade, and G. Richard. Analogical classification: A rule-based view. In Proc. 15th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'14), Montpellier, July 15-19, Part II, pages 485-495, 2014.
3. M. Bounhas, H. Prade, and G. Richard. Analogical classification: Handling numerical data. In U. Straccia and A. Calì, editors, Proc. 8th Int. Conf. on Scalable Uncertainty Management (SUM'14), volume 8720 of LNCS, pages 66-79. Springer, 2014.
4. Myriam Bounhas, Henri Prade and Gilles Richard, "La classification analogique : Une nouvelle manière d'exploiter des exemples". In Actes IAF 2014 - Huitièmes Journées de l'Intelligence Artificielle Fondamentale, Angers, France, 11-13 juin 2014. Online : $http://jiaf2014.univ-angers.fr/actes/Bounhas_IAF_2014.pdf$

5. M. Bounhas, H. Prade, and G. Richard. Evenness-based reasoning with logical proportions applied to classification. In Proc. 9th Int. Conf. on Scalable Uncertainty Management (SUM'15), volume 9310 of LNCS, pages 139-154. Springer, 2015.

6. M. Bounhas, H. Prade, and G. Richard. A new view of conformity and its application to classification. In L. C. van der Gaag, editor, Proc. 13th Eur. Conf. on Symb. and Quantit. Appr. to Reasoning with Uncertainty (ECSQARU'15), Compigne, France, Jul. 15-17, pages 221-232, LNAI. Springer, 2015.

7. M. Bounhas, H. Prade, and G. Richard. Oddness-based classifiers for boolean or numerical data. In Proc. 38th Annual German Conference on AI (KI'15), Advances in Artificial Intelligence, Dresden, Sept. 21-25, volume 9324 of LNCS, pages 32-44. Springer, 2015.

8. M. Bounhas, H. Prade, and G. Richard. Not being at odds with a class: A new way of exploiting neighbors for classification. In G. A. Kaminka, M. Fox, P. Bouquet, E. Hüllermeier, V. Dignum, F. Dignum, and F. van Harmelen, editors, ECAI 2016 - 22nd Europ. Conf. on Artificial Intelligence, 29 Aug.-2 Sept., The Hague, volume 285 of Frontiers in Artificial Intelligence and Applications, pages 1662-1663. IOS Press, 2016.

9. M. Bounhas, M. Pirlot, and H. Prade. Predicting preferences by means of analogical proportions. In M. T. Cox, P. Funk, and S. Begum, editors, Proc. 26th Int. Conf. on Case-Based Reasoning (ICCBR'18), Stockholm, July 9-12, volume 11156 of LNCS, pages 515-531. Springer, 2018.

10. M. Bounhas and B. Elayeb. Analogy-based matching model for domain specific information retrieval. In Proceedings of the 11th International Conference on Agents and Artificial Intelligence, ICAART 2019, Volume 2, Prague, Czech Republic, February 19-21, pages 496–505, 2019.

11. M. Bounhas, M. Pirlot, H. Prade, and O. Sobrie. Comparison of analogy-based methods for predicting preferences. In N. BenAmor and M. Theobald, editors, Proc. 13th Int. Conf. on Scalable Uncertainty Management (SUM'19), Compiègne, Dec. 16-18, pages 339–354, LNCS. Springer, 2019.

12. M. Bounhas and H. Prade. An analogical interpolation method for enlarging a training dataset. In N. BenAmor and M. Theobald, editors, Proc. 13th Int. Conf. on Scalable Uncertainty Management (SUM'19), Compiègne, Dec. 16-18, pages 136–152, LNCS. Springer, 2019.

Bibliography

[1] A. Aamodt and E. Plaza. Case-based reasoning; foundational issues, methodological variations, and system approaches. *AICom, 7 (1)*, pages 39–59, 1994.

[2] S. Bayoudh, L. Miclet, and A. Delhay. Learning by analogy: A classification rule for binary and nominal data. *Proc. Inter. Joint Conf. on Artificial Intelligence IJCAI07*, pages 678–683, 2007.

[3] S. Bayoudh, H. Mouchère, L. Miclet, and E. Anquetil. Learning a classifier with very few examples: Analogy based and knowledge based generation of new examples for character recognition. In*Proc. 18th Eur. Conf. on Mach. Lear. (ECML'07), 527-534*. Springer, 2007.

[4] M. Bounhas and B. Elayeb. Analogy-based matching model for domain-specific information retrieval. In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence, ICAART, Volume 2, Prague, Czech Republic, February 19-21*, pages 496–505, 2019.

[5] M. Bounhas, M. Pirlot, and H. Prade. Predicting preferences by means of analogical proportions. In M. T. Cox, P. Funk, and S. Begum, editors, *Proc. 26th Int. Conf. on Case-Based Reasoning (ICCBR'18), Stockholm, July 9-12*, volume 11156 of *LNCS*, pages 515–531. Springer, 2018.

[6] M. Bounhas, M. Pirlot, H. Prade, and O. Sobrie. Comparison of analogy-based methods for predicting preferences. In N. BenAmor and M. Theobald, editors, *Proc. 13th Int. Conf. on Scalable Uncertainty Management (SUM'19), Compiègne, Dec. 16-18*, LNCS, pages 339–354. Springer, 2019.

[7] M. Bounhas and H. Prade. An analogical interpolation method for enlarging a training dataset. In N. BenAmor and M. Theobald, editors, *Proc. 13th Int. Conf. on Scalable Uncertainty Management (SUM'19), Compiègne, Dec. 16-18*, LNCS, pages 136–152. Springer, 2019.

[8] M. Bounhas, H. Prade, and G. Richard. Analogical classification: A new way to deal with examples. In *Proc. 21st Europ. Conf. on Artificial Intelligence (ECAI'14), Aug. 18-22, Prague*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 135–140. IOS Press, 2014.

[9] M. Bounhas, H. Prade, and G. Richard. Analogical classification: A rule-based view. In *Proc. 15th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'14), Montpellier, July 15-19, Part II*, pages 485–495, 2014.

[10] M. Bounhas, H. Prade, and G. Richard. Analogical classification: Handling numerical data. In U. Straccia and A. Calì, editors, *Proc. 8th Int. Conf. on Scalable Uncertainty Management (SUM'14)*, volume 8720 of *LNCS*, pages 66–79. Springer, 2014.

[11] M. Bounhas, H. Prade, and G. Richard. Evenness-based reasoning with logical proportions applied to classification. In *Proc. 9th Int. Conf. on Scalable Uncertainty Management (SUM'15)*, volume 9310 of *LNCS*, pages 139–154. Springer, 2015.

[12] M. Bounhas, H. Prade, and G. Richard. A new view of conformity and its application to classification. In L. C. van der Gaag, editor, *Proc. 13th Eur. Conf. on Symb. and Quantit. Appr. to Reasoning with Uncertainty (ECSQARU'15), CompiÃ¨gne, France, Jul. 15-17*, LNAI, pages 221–232. Springer, 2015.

[13] M. Bounhas, H. Prade, and G. Richard. Oddness-based classifiers for boolean or numerical data. In *Proc. 38th Annual German Conference on AI (KI'15), Advances in Artificial Intelligence, Dresden, Sept. 21-25*, volume 9324 of *LNCS*, pages 32–44. Springer, 2015.

[14] M. Bounhas, H. Prade, and G. Richard. Not being at odds with a class: A new way of exploiting neighbors for classification. In G. A. Kaminka, M. Fox, P. Bouquet, E. Hüllermeier, V. Dignum, F. Dignum, and F. van Harmelen, editors, *ECAI 2016 - 22nd Europ. Conf. on Artificial Intelligence, 29 Aug.-2 Sept., The Hague*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 1662–1663. IOS Press, 2016.

[15] M. Bounhas, H. Prade, and G. Richard. Analogy-based classifiers for nominal or numerical data. *Int. J. Approx. Reasoning*, 91:36–55, 2017.

[16] M. Bounhas, H. Prade, and G. Richard. Oddness/evenness-based classifiers for boolean or numerical data. *Int. J. Approx. Reasoning*, 82:81–100, 2017.

[17] Myriam Bounhas, Henri Prade, and Gilles Richard. Oddness-based classification: A new way of exploiting neighbors. *Int. J. Intell. Syst.*, 33(12):2379–2401, 2018.

[18] W.J. Conover. *Practical Nonparametric Statistics*. John Wiley, New York, 1971.

[19] D. Coomans and D.L. Massart. "alternative k-nearest neighbour rules in supervised pattern recognition : Part 1. k-nearest neighbour classification by using alternative voting rules". In *Analytica Chimica Acta*, pages 15–27, 1982.

[20] W. Correa, H. Prade, and G. Richard. Trying to understand how analogical classifiers work. In E. Hüllermeier et al., editor, *Proc. 6th Inter. Conf. on Scalable Uncertainty Management (SUM'12), Marburg, Germany*, LNAI 7520, pages 582–589. Springer Verlag, 2012.

[21] W. Correa, H. Prade, and G. Richard. When intelligence is just a matter of copying. In *Proc. 20th Eur. Conf. on Artificial Intelligence, Montpellier, Aug. 27-31*, pages 276–281. IOS Press, 2012.

[22] M. Couceiro, N. Hug, H. Prade, and G. Richard. Analogy-preserving functions: A way to extend Boolean samples. *Proc. of 26th Inter. Joint Conf. on Artificial Intelligence IJCAI17, Melbourne*, pages 1575–1581, 2017.

[23] T. R. Davies and Russell S. J. A logical approach to reasoning by analogy. In J. P. McDermott, editor, *Proc. of the 10th International Joint Conference on Artificial Intelligence (IJCAI'87), Milan*, pages 264–270. Morgan Kaufmann, 1987.

[24] C. Domshlak, E. Hüllermeier, S. Kaci, and H. Prade. Preferences in AI: an overview. *Artif. Intell.*, 175(7-8):1037–1052, 2011.

[25] D. Dubois, H. Prade, and G. Richard. Multiple-valued extensions of analogical proportions. *Fuzzy Sets and Systems*, 292:193–202, 2016.

[26] M. A. Fahandar and E. Hüllermeier. Learning to rank based on analogical aeasoning. In *Proc. 32th Nat Conf. on Artificial Intelligence (AAAI'18), New Orleans, Feb. 2-7*, 2018.

[27] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[28] J. Fürnkranz and E. Hüllermeier, editors. *Preference Learning*. Springer, 2010.

[29] J. Fürnkranz, E. Hüllermeier, C. Rudin, R. Slowinski, and S. Sanner. Preference learning (dagstuhl seminar 14101). *Dagstuhl Reports*, 4(3):1–27, 2014.

[30] J. D. Gergonne. Application de la méthode des moindres quarrés à l'interpolation des suites. *Annales de Math. Pures et Appl,*, 6:242–252, 1815.

[31] J. D. Gergonne. Théorie de la règle de trois. *Annales de Math. Pures et Appl,*, 7:117–122, 1816.

[32] M. Grabisch and Ch. Labreuche. A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid. *Annals of Oper. Res.*, 175:247–286, 2010.

[33] M. Hesse. On defining analogy. *Proceedings of the Aristotelian Society*, 60:79–100, 1959.

[34] N. Hug, H. Prade, G. Richard, and M. Serrurier. Analogical classifiers: A theoretical perspective. In *Proc. 22nd Europ. Conf. on Artificial Intelligence (ECAI'16), The Hague*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2016.

[35] E. Hüllermeier and J. Fürnkranz. Editorial: Preference learning and ranking. *Machine Learning*, 93(2-3):185–189, 2013.

[36] Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled Elbassioni, and Vladimir Gurvich. Generating all vertices of a polyhedron is hard. *Discrete & Computational Geometry*, 39(1):174–190, Mar 2008.

[37] P. Langley and W. Iba. Average-case analysis of a nearest neighbor algorithm. In *Proc. 13th Int. Joint Conf. on Artificial Intelligence. Chambéry, France, August 28 - September 3*, pages 889–894. Morgan Kaufmann, 1993.

[38] J. F. Lavallée and P. Langlais. Moranapho: un système multilingue d'analyse morphologique basé sur l'analogie formelle. *TAL*, 52(2):17–44, 2011.

[39] Y. Lepage. Analogy and formal languages. *Electr. Notes Theor. Comput. Sci.*, 53, 2001.

[40] Yves Lepage. Analogy and formal languages. In *Proc. FG/MOL 2001*, pages 373–378, 2001. (see also http://www.slt.atr.co.jp/ lepage/pdf/dhdryl.pdf.gz).

[41] E. Melis and M. Veloso. Analogy in problem solving. In *Handbook of Practical Reasoning: Comput. and Theor. Aspects*. OUP, 1998.

[42] J. Mertz and P.M. Murphy. Uci repository of machine learning databases. *Available at: ftp://ftp.ics.uci.edu/pub/machine-learning-databases*, 2000.

[43] L. Miclet, N. Barbot, and B. Jeudy. Analogical proportions in a lattice of sets of alignments built on the common subwords in a finite language. In H. Prade and G. Richard, editors, *Computational Approaches to Analogical Reasoning - Current Trends*, page : in this volume. Springer, 2013.

[44] L. Miclet, N. Barbot, and H. Prade. From analogical proportions in lattices to proportional analogies in formal concepts. In *Proc. 21st Europ. Conf. on Artificial Intelligence (ECAI'14), 18-22 Aug., Prague*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 627–632. IOS Press, 2014.

[45] L. Miclet, S. Bayoudh, and A. Delhay. Analogical dissimilarity: definition, algorithms and two experiments in machine learning. *JAIR, 32*, pages 793–824, 2008.

[46] L. Miclet and A. Delhay. Relation d'analogie et distance sur un alphabet défini par des traits. Technical Report 1632, IRISA, July 2004.

[47] L. Miclet and H. Prade. Handling analogical proportions in classical logic and fuzzy logics settings. In *Proc. 10th Eur. Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'09),Verona*, pages 638–650. Springer, LNCS 5590, 2009.

[48] S. Okamoto and N. Yugami. Theoretical analysis of the nearest neighbor classifier in noisy domains. In *Proc. 13th International Conference on Machine Learning*, pages 355–363. Morgan Kaufmann, 1996.

[49] M. Pirlot, H. Prade, and G. Richard. Completing preferences by means of analogical proportions. In V. Torra, Y. Narukawa, G. Navarro-Arribas, and C. Yañez, editors, *Proc. 13th Int. Conf. on Modeling Decisions for Artificial Intelligence (MDAI'16), Sant Julià de Lòria, Andorra, Sept. 19-21*, volume 9880 of *LNAI*, pages 135–147. Springer, 2016.

[50] H. Prade and G. Richard. Reasoning with logical proportions. In *Proc. 12th Int. Conf. on Principles of Knowledge Representation and Reasoning, KR 2010, Toronto, May 9-13, 2010 (F. Z. Lin, U. Sattler, M. Truszczynski, eds.)*, pages 545–555. AAAI Press, 2010.

[51] H. Prade and G. Richard. Homogeneous logical proportions: Their uniqueness and their role in similarity-based prediction. In G. Brewka, T. Eiter, and S. A. McIlraith, editors, *Proc. 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'12), Roma, June 10-14*, pages 402–412. AAAI Press, 2012.

[52] H. Prade and G. Richard. Analogical proportions and multiple-valued logics. In L. C. van der Gaag, editor, *Proc. 12th Eur. Conf. on Symb. and Quantit. Appr. to Reasoning with Uncertainty (ECSQARU'13), Utrecht, Jul. 7-10*, LNAI 7958, pages 497–509. Springer, 2013.

[53] H. Prade and G. Richard. From analogical proportion to logical proportions. *Logica Universalis*, 7(4):441–505, 2013.

[54] H. Prade and G. Richard. Homogenous and heterogeneous logical proportions. *IfCoLog J. of Logics and their Applications*, 1(1):1–51, 2014.

[55] H. Prade and G. Richard. On different ways to be (dis)similar to elements in a set. Boolean analysis and graded extension. In J. P. Carvalho, M.-J. Lesot, U. Kaymak, S. M. Vieira, B. Bouchon-Meunier, and R. R. Yager, editors, *Proc. 16th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'16), Part II, Eindhoven, June 20-24*, volume 611 of *Comm. in Comp. and Inf. Sci.*, pages 605–618. Springer, 2016.

[56] H. Prade and G. Richard. Multiple-valued logic interpretations of analogical, reverse analogical, and paralogical proportions. In *Proc. 40th IEEE Int. Symp. on Multiple-Valued Logic (ISMVL'10)*, pages 258–263, Barcelona, 2010.

[57] H. Prade, G. Richard, and B. Yao. Enforcing regularity by means of analogy-related proportions-a new approach to classification. *International Journal of Computer Information Systems and Industrial Management Applications*, 4:648–658, 2012.

[58] M. Ragni and S. Neubert. Analyzing raven's intelligence test: Cognitive model, demand, and complexity. In *Computational Approaches to Analogical Reasoning: Current Trends*, volume 548 of *Studies in Computational Intelligence*, pages 351–370. Springer, 2014.

[59] N. Rescher. *Many-valued Logic*. McGraw-Hill Inc., New York, 1969.

[60] D. E. Rumelhart and A. A. Abrahamson. A model for analogical reasoning. *Cognitive Psychol.*, 5:1–28, 2005.

[61] E. Sander. *L'Analogie, du Naïf au Créatif : Analogie et Catégorisation*. L'Harmattan, 2000.

[62] C. Saunders, A. Gammerman, and V. Vovk. Computationally efficient transductive machines. In H. Arimura, S. Jain, and A. Sharma, editors, *Proc. of Algorithmic Learning Theory, 11th International Conference, ALT 2000, Sydney, Australia, December 11-13*, volume 1968 of *LNCS*, pages 325–333. Springer, 2000.

[63] N. Stroppa and F. Yvon. Analogical learning and formal proportions: Definitions and methodological issues. Technical report, ENST, June 2005.

[64] N. Stroppa and F. Yvon. Du quatrième de proportion comme principe inductif : une proposition et son application à l'apprentissage de la morphologie. *Traitement Automatique des Langues*, 47(2):1–27, 2006.

[65] A. Tversky. Intransitivity of preferences. *Psychological Review*, 76:31–48, 1969.

[66] V. Vovk. Cross-conformal predictors. *Annals of Maths and Artificial Intelligence*, pages 9–28, 2015.

[67] V. Vovk, A. Gammerman, and C. Saunders. Machine-learning applications of algorithmic randomness. *Int. Conf. on Machine Learning*, pages 444–453, 1999.

[68] Chih wei Hsu, Chih chung Chang, and Chih jen Lin. A practical guide to support vector classification, 2010.

[69] F. Yvon and N. Stroppa. Formal models of analogical proportions. Technical report, Ecole Nationale Supérieure des Télécommunications, no D008, 2006.

[70] F. Yvon, N. Stroppa, A. Delhay, and L. Miclet. Solving analogical equations on words. Technical report, Ecole Nationale Supérieure des Télécommunications, 2004.