Preference Learning for Object Ranking and Classification with Fixed-point Algorithm

Bénédicte Goujon[1]

Abstract.

In this paper, we propose an initial algorithm targetting the restricted object ranking, which consists in ranking new alternatives with an interpretable model built automatically from small learning datasets described with a limited number of monotone attributes. The objective of our algorithm is to provide a 2-additive Choquet integral to model learning datasets. Our approach is based on a fixed-point algorithm calling at each step a linear program used to find alternatively Möbius coefficients and utility values.

In order to evaluate our approach, we also present a classification version of this algorithm, adapted to treat classification benchmarks like CPU. A first evaluation is also presented and some weaknesses are identified for future improvements.

1 Introduction

Preference learning is a large domain with various types of problems, like instance ranking (i.e. learning from alternatives shared out among ordered classes in order to class new alternatives) or restricted object ranking (i.e. learning preferences of a particular user on few alternatives in order to evaluate and rank new alternatives).

In this paper, we target the restricted object ranking, which consists in ranking new alternatives with an interpretable model built automatically from small learning datasets described with a limited number of monotone attributes. Interpretability of the model is important in our approach since it allows to explain the results to decision makers. Monotonicity of the attributes is required.

We present two versions of an algorithm, based on the fixed-point algorithm, to learn from preferences a 2-additive Choquet integral model. The complexity of the task consists in learning both weights (with interactions) and utility functions. In this paper, we first present our preference learning context, with the description and characterisation of preference learning problems, and a state of the art focused on object ranking methods and methods targetting interpretable models with monotone attributes. Then we describe our approach, with the initial version of the algorithm targetting restricted object ranking and the classification version of the algorithm built in order to compare our results with existing works. Last chapter presents the evaluation of our classification algorithm on the CPU benchmark.

2 Preference learning context

2.1 Examples of problems to solve

In this section we present the first problems that we consider on the use of preference learning to build a decision-model in order to evaluate and/or rank new alternatives.

[1] Thales Research and Technology France, email: benedicte.goujon@thalesgroup.com

2.1.1 Thierry's choice problem

The first targeted problem is the Thierrys choice problem [3]. In this problem, Thierry has to choose a car among a set of 14 models. Cars are characterized with 5 criteria. Table 1 presents a subset of alternatives of this problem.

Name of cars	Crit1 Cost	Crit2 Accel	Crit3 Pick up	Crit4 Brakes	Crit5 Road-h
Fiat Tipo	18 342	30.7	37.2	2.33	3
Alfa 33	15 335	30.2	41.6	2	2.5
Nissan Sunny	16 973	29	34.9	2.66	2.5
Mazda 323	15 460	30.4	35.8	1.66	1.5

Table 1. Thierry's choice problem dataset.

Thierry is only able to express a preference order on 5 models. From this knowledge on his preference, the objective is to propose him the best car of the whole set of cars.

2.1.2 Second Problem

In a second problem, the objective is to evaluate any new alternative based on the evaluation provided by experts on few alternatives. Alternatives are described according to 5 criteria. Table 2 presents the values on criteria and global utility scores of learning alternatives.

	C1	C2	C3	C4	C5	Global utility
Alternative Ex3	-1	0	70	0	0	0.92
Alternative Ex4	2	1	12	1	1	0.85
Alternative Ex5	2	1	12	2	0	0.8
Alternative Ex6	2	0	6	3	0	0.6
Alternative Ex8	3	0	6	3	0	0.6
Alternative Ex9	2	1	24	2	2	0.68
Alternative Ex10	2	1	6	2	2	0.5

Table 2. The second problem's learning set of evaluated alternatives.

2.2 Various context of preference learning

In general preference learning, the objective is to learn a model that represents a learning data set (based on alternatives described with

several attributes) in order to identify or predict an evaluation or classification of new alternatives.

2.2.1 *Preference learning Approaches*

Several types of problems are identified in the general domain called preference learning:

- **Object ranking** In this kind of problem, inputs are ordered alternatives, characterized with some attributes. The objective is to order new instances. A method consists in evaluate new instances first, and then order them according to their score.
- **Instance ranking (= sorting)** In this kind of problem, inputs are instances, characterized with some attributes and associated to classes, classes being ordered. In this context, the objective is to class new instances. Examples of classes are: "very good car", "average car", "bad car".
- **Label ranking (= classification)** In this type of problem, inputs are instances characterized by attributes and associated to order on labels. The aim is to associate to each new instance an ordered set of labels. For instance, to class news, one would like to associate labels with a relevance order such as: sport 0.8, economy 0.5, international 0.4...

2.2.2 *Preference learning Characteristics*

Several characteristics can be used to specify preference learning contexts:

- **Preference learning problem** As explained above, different kinds of preference learning problems can be targeted: object ranking, instance ranking/sorting, label ranking/classification.
- **Preference model** Preference models can be divided into two main classes of models: "black box" models and interpretable models. For black box models, several parameters have to be learned to obtain the best results. In such approach, the monotonicity on each attribute is not always expected.
  For a subset of interpretable models, first main characteristics are the utility functions on each attribute, which can be more or less complex (using normalized values, searching thresholds values or utility functions based on non trivial piecewise linear functions). Second main characteristics are the weights (weights only or weights with interactions between attributes).
- **Learning method** To learn a model, different methods can be used: algorithms based on metaheuristics, using mixed integer program (MIP) or linear programming (LP), genetic algorithms...
- **Learning dataset** Learning datasets with different input characteristics can be used: small vs large sets of data, noisy vs noiseless data, complete vs incomplete data, coherent vs non coherent data, data with low vs high number of attributes, data with different kinds of attributes (numerical vs discrete values, monotone vs non-monotone functions).

2.2.3 *Restricted Object Ranking*

In our context, we use a MCDA (multi-criteria decision analysis) approach based on the 2-additive Choquet integral. Currently, the utility functions on each criterion and the weights of criteria on each aggregation are learned separatly with few adapted examples provided by experts. In the approach presented here, the objective is to learn directly both utility functions and weights from preferences expressed on complete alternatives provided by experts. In our usual context, learning datasets are coherent, since experts are asked to provide small set of preferences on alternatives (that could be expressed through set of evaluated alternatives).

According to the preference learning domain, our objective is to solve an object ranking problem (i.e. to rank new alternatives) with an interpretable model built automatically from small learning datasets (less than 50 alternatives) with a limited number of monotone attributes.

We propose to call such approach "Restricted Object Ranking", since the monotonicity of the attributes, the interpretability of the model and the small size of coherent learning data restrict the set of targetable object ranking problems.

2.3 State of the art

In this state of the art, we present approaches in preference learning domain dealing with some characteristics of our "restricted object ranking" approach: object ranking or interpretable models with monotone attributes.

2.3.1 *Object Ranking methods*

In 2011, Kamishima et al [10] presented a state of the art on the large Object Ranking Methods. For instance, in the two experiments presented to compare methods, 1000 alternatives are taken into account, with 300 and 500 orders in input. In this paper, the following schema illustrates the task of object ranking:

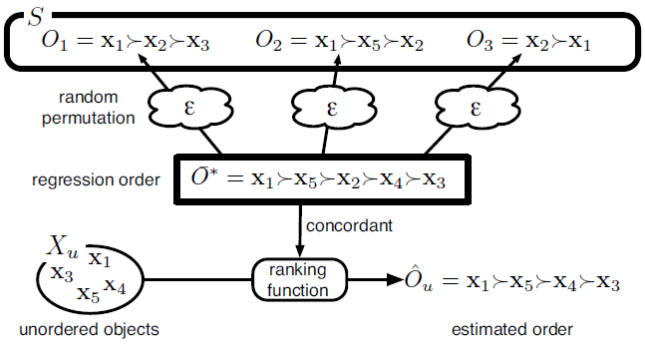

Figure 1. The object ranking task for [10]

In Fig. 1, O_1 represents a preference order on the subset of alternatives x_1, x_2, x_3. We can see on Fig. 1 that some orders are not coherent (O_1 vs O_3), but since large set of learning datasets are targetted, this inconsistency is compensated using the frequency of opposite preferences (most frequent preference is considered as valid one).

In this state of the art, several methods are detailed: Cohen's method [4], RankBoost [6] [6], two approaches based on SVM, etc. Those methods do not take into account the monotonicity of the utility values and the explicability, that are important points in our approach.

2.3.2 *Choquistic regression*

In [12], authors present the use of the Choquet integral to learn monotone nonlinear models. They propose a generalization of logistic

regression, called choquistic regression, to replace the linear function of predictor variables (commonly used in logistic regression to model the log odds of the positive class) by the Choquet integral.

$$\pi_c \stackrel{df}{=} \mathbf{P}(y = 1|x) = \Big(1 + \exp(-\gamma(C_\mu(f_x) - \beta))\Big)^{-1}. \quad (1)$$

where $C_\mu(f_x)$ is the Choquet integral (with respect to the measure μ) of the function $f_x : \{c_1, \ldots, c_m\} \to [0, 1]$ that maps each attribute c_i to a normalized value $x_i = f_x(c_i) \in [0, 1]$; $\beta, \gamma \in \mathbb{R}$ are constants. Their objective is to obtain a binary result: 1 if an alternative is in the positive class, and 0 if not. The model has several degrees of freedom: the fuzzy measure (or capacity) which determines the utility function, the utility threshold β and the scaling parameter γ which determine the discrete choice (yes/no decision) model. The goal of learning is to identify these parameters on the basis of the training data D. In this approach, each utility function is modeled with a linear function between the worst value (associated to 0) and the best one (associated to 1). This method of binary classification provides very good results on large datasets but does not target object ranking.

2.3.3 Learning a MR-Sort model for large datasets

In [11], a method is proposed to solve sorting problems using a majority rule sorting (MR-Sort) model. Such model is a particular case of the definition of non-compensatory models [1] [2], with profile thresholds on each criterion. In this work, a metaheuristic is proposed to learn the parameters of the MR-Sort model, divided in three components: 1- A heuristic which initializes a set of profiles; 2- A linear program learning the weights and the majority threshold of the model based on fixed profiles; 3- A heuristic adjusting the profiles to improve the quality of the model, while keeping the weights and majority threshold fixed. As in [12], this work gives importance to interpretability, accuracy and monotonicity of criteria, but the aim is the classification of new alternatives (results are proposed with 2 and 4 classes). In this work, no interaction between criteria is taken into account.

2.3.4 FlowSort

FlowSort is a version of the large set of tools called PROMETHEE [14].

PROMETHEE is an outranking method. It is based on the calculation of a preference function $P_i(a, b)$ for each criterion, using an intensity of preference for an alternative a over an alternative b. In this approach, the decision maker selects the shape of each preference function and specifies any parameters that are then needed (like thresholds). Then a global preference index $P(a, b)$ is calculated:

$$P(a, b) = \frac{\sum_{i=1}^{n} w_i \, P_i(a, b)}{\sum_{i=1}^{n} w_i}. \quad (2)$$

PROMETHEE I and II target the instance ranking. PROMETHEE I provides partial ranking while PROMETHEE II provides a total relation. FlowSort [13] is based on PROMETHEE I and II and provides interval sorting, i.e. it can associate two consecutive classes to alternatives. It is based on a genetic algorithm to find the best parameters (on each criterion: weight, indifference threshold, preference threshold and central profile for each category).

3 A Restricted Object Ranking Algorithm

3.1 Presentation of the problem

Main idea of this work is to propose an algorithm to automatically build a multicriteria model from the learning set of preferences, by learning weights and utility functions of a 2-additive Choquet integral [8].

$$H(x_1, \ldots, x_n) = \sum_{i \in N} m_i \, u_i(x_i) + \sum_{\{i,j\} \subseteq N} m_{i,j} \, u_i(x_i) \wedge u_j(x_j). \quad (3)$$

where u_i is the utility function on the criterion i (values in [0,1]) and m_i and $m_{i,j}$ are the Möbius coefficients: m_i is the weight of the criterion i (positive or null), $m_{i,j}$ is the interaction weight associated to the minimum between $u_i(x_i)$ and $u_j(x_j)$ (positive or negative). $H(x_1, \ldots, x_n)$ represents the global utility (value in [0, 1]) of the alternative $x = (x_1, \ldots, x_n)$, and x_i is the universe value of the alternative on the criterion i.

In the targeted problems, we know the x_i values for each alternative x of the learning set X. For some alternatives we also know the global utility value. Else, we have several preferences between $x, y \in X$:

$$x \prec y \iff H(x_1, \ldots, x_n) < H(y_1, \ldots, y_n) \quad (4)$$

We do not have a particular objective dealing with efficiency in term of time-consuming, but we do not want each run of our algorithm to last several hours.

3.2 Initial approach

A model as defined in (3) cannnot be solved directly with linear programming due to the fact that it contains products of variables (weights and utility function values). Similarly, it cannot be solved directly with constraint programming since variables (weights and utility function values) are continuous. Main idea of our approach consists in dividing the problem into two sub-problems that can be solved with linear programming:

- weight learning, with known utility values;
- utility values learning, with known weight.

A first algorithm has been proposed and developed, based on the use of the fixed point algorithm to alternatively treat the two sub-problems. Theoretical approach is detailed in [7].

Here are the steps of the algorithm:

1 Generation of an order on utility values
2 Attribution of initial values to utilities
3 Fixed point algorithm (sub-steps are called successively until a stop condition):

- **3.1** Search of values for the weights and interactions verifying the learning data, using the values on utilities calculated previously (linear programming (LP)).
- **3.2** Search of values for the utilities verifying the learning data, using the weights and interactions calculated previously (LP).

Step 1 generates a relevant global order on all utility values ($\{u_i(x_i), i \in N, x \in X\}$) and step 2 initializes those utility values, attributing values between 0 and 1 with respect to the global order.

In this version, each interaction between criteria values of each learning alternative is associated to the learning dataset is a variable in the LP of step 3.1. For instance, for all $\{i, j\} \in N$, x and $y \in X$, $m_{i,j}(x)$ and $m_{i,j}(y)$ are distinct variables. Similarly, each utility value of the learning set of alternatives is a variable in the LP of step 3.2. Last point means that considering 20 learning alternatives with distinct values on each criterion, each resulting utility function will be a piecewise linear function with 19 segments, which is very precise compared to [12] where each utility function is represented by a linear function.

Another specificity of this version consists in taking into account preferences between alternatives, since we have in input ordered alternatives. We decides to consider only the preferences between each alternative and the five closest following ones in order to have a restricted but relevant set of preferences.

Each LP has a binary variable associated to each preferencein order to penalize the unsatisfaction of each preference. The objective of each LP is to minimize the sum of all those binary variables.

A stop condition is encountered when, for instance, time limitation is reached or when new values are equal to previous ones.

Main limit of this algorithm is the number of binary variables on both steps that is linearly related to the number of preferences which is very high even with a quite small set of learning data. The high number of variables is also due to the management of interactions between criteria in step 3.1 and the management of multiple utility values in step 3.2.

For instance, with a learning dataset composed of 500 preferences between 105 alternatives described with 6 criteria, the linear program in step 3.2 has to handle more than 4000 variables: more than 600 for the utility values ($u_i(x_i)$), more than 3100 for the interaction utility values ($u_i(x_i) \wedge u_j(x_j)$) and 500 binary variables related to the number of preferences.

3.3 Classification version for comparison

One of our objectives is to compare our approach to existing ones. Since we did not identify benchmarks dedicated to the restricted object ranking task as defined previously, we have tried to evaluate our approach on benchmarks related to instance ranking (classification) with a limited number of alternatives.

This new objective brings us to provide a version of our algorithm more adapted to classification benchmarks since such problems have more alternatives to evaluate than in our initial objective. First evaluations with our previous algorithm show that it was not efficient due to the high number of variables and with a duration limit equals to 5 minutes.

The classification version of our algorithm is still based on the fixed point algorithm, but four improvements have been proposed to reduce the number of variables.

First, we have chosen to apply the fixed point algorithm to problems without interactions between criteria. But, in order to provide results taking potential interactions between criteria into account, a last step has been added to complete best results obtained with the fixed point algorithm by adding interactions between criteria.

Second, we have simplified utility functions in order to reduce the associated number of variables, like in [12], keeping only 3 values for each utility function regardless the number of learning alternatives. For instance, let's consider a dataset with 6 criteria and 50 distinct values on each criterion. With previous algorithm, each utility function contains 50 referential points and 300 variables (corresponding to utility values on all criteria) are used by the LP searching values for the utilities. With this simplification, only 18 variables are used in the LP searching values for the utilities, even with a dataset containing a larger number of distinct values on each criterion.

Third, to initialize utility values, we have decided to normalize the utility values rather than generate an order on utility values and attribute values between 0 and 1, since several tests show better results with this simple initialisation. For instance, with 2 criteria and 3 distinct utility values for each (both increasing utility functions), we previously could have the following attribution of values: $u_1(2) = 0.1$, $u_2(200) = 0.3$, $u_2(208) = 0.35$, $u_2(211) = 0.4$, $u_1(3) = 0.6$, $u_1(5) = 0.9$. With the normalization, we obtain the following values: $u_1(2) = u_2(200) = 0$, $u_1(3) = 0.33$, $u_2(208) = 0.73$, $u_2(211) = u_1(5) = 1$.

Fourthly, we have also changed the use of preferences by managing preferences expressed between an alternative and one or two threshold(s) (relevant for classification) rather than preferences between alternatives. The objective is to adapt the meaning of preferences to the classification task and to reduce the number of preferences with a small number of classes.

Here are the steps of this classification version of the algorithm:

Phase 1: no interaction (loop)

- **1.1** Normalization of subset of utility values
- **1.2** Fixed point algorithm (sub-steps are called successively until a stop value):
  - **1.2.1** Search of values for the weights (no interaction) verifying the learning data, using the values on utilities calculated previously (linear programming).
  - **1.2.2** Search of values for the utilities verifying the learning data, using the weights and interactions calculated previously (linear programming).

Phase 2: with interactions and all utility values (limited number of run)

- **2.1** Attribution of initial values to weights (from one best solution of the previous phase)
- **2.2** Fixed point like algorithm (sub-steps are called once):
- **2.2.1** Search of values for all the utilities (linear programming).
- **2.2.2** Search of values for the weights with interactions (linear programming).

The advantage of this algorithm is that it calls several times a problem with a smaller number of variables and constraints (managing less preferences) in step 1.2.1 and step 1.2.2, compared to initial version. For instance, with a learning dataset composed of 160 preferences between alternatives (105 learning alternatives) described with 6 criteria and one threshold (binary classification), the linear program in step 1.2.1 has to handle 166 variables and the linear program in step 1.2.2 has to handle 178 variables: 18 for the utility values ($u_i(x_i)$) and 160 for the preferences.

We can notice that the first step has similarities with the UTADIS method [5] [9], manipulating preferences between alternatives and thresholds and using a normalization of utility values. But, in UTADIS, only one PL is used to search marginal utility functions in order to obtain the global utility value by adding utilities.

4 Evaluation on CPU benchmark

4.1 Presentation of the CPU benchmark

To evaluate our approach and compare its results to the existing ones, we have chosen the CPU benchmark, provided in the UCI repository [2]. We have chosen this benchmark since it was easily available, it contains quite few alternatives and each utility value is clearly monotone (numerical values). Some other benchmarks contain too many alternatives (like CEV which contain 1728 alternatives), some are not available (i.e. benchmarks provided by WEKA, used by Tehrani et al [12]) or some have non-numerical values that cannot easily be transformed into monotone criteria (for instance breast-quad, from the BBC benchmark, with the following values: left-up, left-low, right-up, righ-low and central).

CPU benchmark contains 209 instances and 6 attributes. All the attributes are considered as monotone.

4.2 Evaluation of our approach on binary classification

The threshold between the two classes is the median value: 50. Table 3 presents state of the art results on the binary classification presented by Tehrani et al in [12] and by Sobrie in [11].

	Tehrani et al [12]		Sobrie [11]	
	CR	LR	META MR-Sort	UTADIS
20%	.0457 ± .0338	.0430 ± .0318	.0994 ± .0323	.0652 ± .0362
50%	.0156 ± .0135	.0400 ± .0106	.0675 ± .0237	.0230 ± .0238
80%	.0089 ± .0126	.0366 ± .0068	.0640 ± .0304	.0152 ± .0214

Table 3. State of the art results for binary classification on CPU (0/1 Loss).

In table 3, CR stands for Choquistic Regression, which is the method presented by Tehrani et al, and META MR-Sort corresponds to the method developed by Sobrie.

Table 4 presents the results of our classification algorithm (classif) and the results of the classical Logistic Regression (LR) method, implemented using the Logistic Regression algorithm provided in the machine learning Python's library called scikit-learn. As state of the art results presented in table 3, we have used 100 runs targetting the PRP value, with 3 percentage values of learning data: 20%, 50% and 80%. Each run has a random learning set.

	Our results	
	Classif	LR
20%	.153 ± .019	.169 ± .037
50%	.155 ± .040	.155 ± .025
80%	.126 ± .047	.151 ± .052

Table 4. Our results for binary classification on CPU (0/1 Loss).

We can observe that our results are very different (lower) from the state of the art's approaches ones. Comparing results provided by our classification algorithm and LR ones, we can notice that our classification algorithm is more efficient than our LR implementation with 20% and 80% and has the same efficiency with 50%.

We can notice a great difference between our results with LR and the results with the same method obtained by [12] that we cannot explain.

4.3 Evaluation of our approach on 4 classes classification

The thresholds between the four classes are the following (median values): 27, 50 and 113. Table 5 presents some results on the 4 classes classification presented by Sobrie in [11] and by Van Assche and De Smet in [13]. It compares those state of the art results with the result of the classification version of our algorithm ("classif"), using (as compared ones) 100 runs targetting the PRP value, with 3 percentage values of learning data: 20%, 50% and 80%. Each run has a random learning set. We also add results obtained with the Logistic Regression approach.

	Sobrie [11]		Van Assche, De Smet [13]	Our results	
	META MR-Sort	UTADIS	FlowSort	Classif	LR
20%	.2457 ± .0479	.1321 ± .0488	.120 ± .025	.367 ± .047	.417 ± .047
50%	.1961 ± .0354	.0660 ± .0266	.094 ± .023	.343 ± .052	.356 ± .040
80%	.2076 ± .0606	.0488 ± .0351		.377 ± .074	.341 ± .068

Table 5. Results for 4-classes classification on CPU (0/1 Loss).

As for the previous evaluation, we can observe that our results are not as good as those provided by other approaches. But, similarly, our classification algorithm is more efficient than our LR implementation with 20% and 50%. We can also specify that logically the method proposed in [13] obtains better results since the approach allows the attribution of multiple classes to several alternatives .

4.4 Precision limitations

In our approach, the idea is to find a precise model, with potentially complex monotone utility values and with weights and interactions between criteria, as modeled with the 2-additive Choquet integral in (3). Table 6 presents a comparison of the number of variables implied in different methods. In this table, n is the number of criteria, m is the number of distinct universe values in the learning dataset, n_p^1 is the number of preferences between alternatives of the learning dataset and n_p^2 is the number of preferences between alternatives of the learning datasets and thresholds between classes.

Let's note s_d the size of the learning dataset. n_p^1 can be very high ($(\frac{s_d}{2})^2$) while n_p^2 is smaller than $2s_d$. Those additional variables are binary variables in our Initial and Classification algorithms.

In order to obtain a precise model as targetted in our approach, we need to call a high enough number of times the fixed-point algorithm, in order to increase the chances to find good values on both utility values and weights. The difficulty is that a complex model has a high number of variables (as detailed in table 6) and corresponding LPs in the fixed-point algorithm have a calculation time increasing with the number of variables.

[2] http://archive.ics.uci.edu/ml

	Tehrani et al [12]	Sobrie [11]	Inital	Classif
Objective	Binary classif.	Classif.	Object ranking	Classif.
Interactions?	yes	no	yes	yes
Weight variables	$\frac{n(n+1)}{2}-1$	n	$\frac{n(n+1)}{2}-1$	n (1.2) $\frac{n(n+1)}{2}-1$(1.3)
Utility function forms	linear	thresholds	piecewise linear	piecewise linear
Utility variables	0	$3n$	m	$3n$
Additional variables	2 (γ and β)	no	n_p^1	n_p^2

Table 6. Number of variables in different approaches.

Table 7 presents the average number of calls of linear programs (steps 3.1 and 3.2 in initial version, steps 1.3.1 and 1.3.2 in classification version) according to the percentage of learning data, with runs having the maximal duration (5 minutes) with the CPU dataset. We have also specified the average number of associated preferences (that corresponds to n_p^1 for initial version and n_p^2 for classification version).

	Initial version		Classification version	
	Preferences	LP calls	Preferences	LP calls
20%	116	134	60	396
50%	305	91	159	129
80%	494	72	253	100

Table 7. Number of preferences and LP calls with our algorithms in 5 mn.

Table 7 shows a low number of calls of linear programs in each 5 minutes run of the initial algorithm. It is due to the high number of variables (with interactions) and of preferences. With the improvements brought to the classification version, the number of calls is higher thanks to a smaller set of variables (no interaction in main linear problems) and less preferences (with preferences between alternatives and classes thresholds rather than between couples of alternatives).

5 Conclusion

This paper investigates the restricted object ranking problem, which is an object ranking problem with following restrictions: monotonicity of the attributes, interpretability of the model and small size of coherent learning data. We have presented an initial version of our algorithm dedicated to restricted object ranking, based on the use of a fixed-point algorithm to alternatively search weight and utility values, targetting a 2-additive Choquet integral model. In order to compare our approach with existing ones, we had to use instance ranking (classification) benchmarks. We have then proposed a classification version of our algorithm which contains problems with less variables (no interaction, simplified utility functions and less preferences) called by the fixed-point algorithm. For instance, for a particular problem, the linear program searching values for the utilities contains more than 4000 variables (including 500 binary ones) in the initial algorithm, and only 178 variables (including 160 binary ones) in the classification algorithm.

First evaluations on the CPU benchmark (209 instances, 6 criteria) show better results of our classification algorithm compared to our implementation of the logistic regression (based on a Python's library version) on binary classification with 20 and 80% of learning data (equality with 50%), and on four classes classification with 20 and 50% of learning data. Compared to state-of-the-art results, our results are worse, but we have observed that results obtained by LR by Tehrani et al are very different to those obtained with our implementation of LR.

We have shown that one of the main problem of our algorithm is not only the number of variables but also the number of calls to linear programs. Even if a higher number of calls of LP with the classification version of our algorithm improves the results (i.e. the relevance of the model -weights and utility values- to represent the learning dataset), our classification approach still have results for both binary and 4-classes classification problems not as good as those obtained in state-of-the-art.

We will have to do more tests in order to understand the limits of our approach and how to improve it.

REFERENCES

[1] Denis Bouyssou and Thierry Marchant, 'An axiomatic approach to noncompensatory sorting methods in mcdm, i: The case of two categories', *European Journal of Operational Research*, **178**(1), 217–245, (2007).

[2] Denis Bouyssou and Thierry Marchant, 'An axiomatic approach to noncompensatory sorting methods in mcdm, ii: More than two categories', *European Journal of Operational Research*, **178**(1), 246–276, (2007).

[3] Denis Bouyssou, Thierry Marchant, Marc Pirlot, Patrice Perny, Alexis Tsoukiàs, and Philippe Vincke, *Comparing on the Basis of Several Attributes: The Example of Multiple Criteria Decision Analysis*, 91–152, Springer US, Boston, MA, 2000.

[4] William W Cohen, Robert E Schapire, and Yoram Singer, 'Learning to order things', in *Advances in Neural Information Processing Systems*, pp. 451–457, (1998).

[5] J.M. Devaud, G. Groussaud, and E. Jacquet-Lagreze, 'UTADIS: Une methode de construction de fonctions d'utilite additives rendant compte de jugements globaux', in *European working group on MCDA, Bochum , Germany*, (1980).

[6] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer, 'An efficient boosting algorithm for combining preferences', *Journal of machine learning research*, **4**(Nov), 933–969, (2003).

[7] Bénédicte Goujon and Christophe Labreuche, 'Holistic preference learning with the choquet integral', in *8th conference of the European Society for Fuzzy Logic and Technology (EUSFLAT-13)*. Atlantis Press, (2013).

[8] Michel Grabisch, 'The application of fuzzy integrals in multicriteria decision making', *European journal of operational research*, **89**(3), 445–456, (1996).

[9] E. Jacquet-Lagrèze and Y. Siskos, 'Assessing a set of additive utility functions for multicriteria decision making: the UTA method', *European Journal of Operational Research*, **10**, 151–164, (1982).

[10] Toshihiro Kamishima, Hideto Kazawa, and Shotaro Akaho, *A Survey and Empirical Comparison of Object Ranking Methods*, 181–201, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[11] Olivier Sobrie, *Learning preferences with multiple-criteria models*, Ph.D. dissertation, Université Paris-Saclay, 2016.

[12] Ali Fallah Tehrani, Weiwei Cheng, Krzysztof Dembczyński, and Eyke Hüllermeier, 'Learning monotone nonlinear models using the choquet integral', *Machine Learning*, **89**(1-2), 183–211, (2012).

[13] Dimitri Van Assche and Yves De Smet, 'Flowsort parameters elicitation based on categorization examples code-smg–technical report series', (2015).

[14] JP Vincke and Ph Brans, 'A preference ranking organization method. the promethee method for mcdm', *Management Science*, **31**(6), 647–656, (1985).