Learning GAI-decomposable Utility Models for Multiattribute Decision Making
Margot Herin, Patrice Perny, Nataliya Sokolovska

To cite this version:
Margot Herin, Patrice Perny, Nataliya Sokolovska. Learning GAI-decomposable Utility Models for Multiattribute Decision Making. The 38th Annual AAAI Conference on Artificial Intelligence (AAAI 2024), Feb 2024, Vancouver, Canada. hal-04424705

HAL Id: hal-04424705
https://hal.science/hal-04424705
Submitted on 22 Feb 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
LEARNING GAI-DECOMPOSABLE UTILITY MODELS FOR MULTIATTRIBUTE DECISION MAKING

Margot Herin, Patrice Perny
LIP6
Sorbonne University,
Paris
{margot.herin, patrice.perny}@lip6.fr

Nataliya Sokolovska
LCQB
Sorbonne University,
Paris
nataliya.sokolovska@sorbonne-universite.fr

ABSTRACT

We propose an approach to learn a multiattribute utility function to model, explain or predict the value system of a Decision Maker. The main challenge of the modeling task is to describe human values and preferences in the presence of interacting attributes while keeping the utility function as simple as possible. We focus on the generalized additive decomposable utility model which allows interactions between attributes while preserving some additive decomposability of the evaluation model. We present a learning approach able to identify the factors of interacting attributes and to learn the utility functions defined on these factors. This approach relies on the determination of a sparse representation of the ANOVA decomposition of the multiattribute utility function using multiple kernel learning. It applies to both continuous and discrete attributes. Numerical tests are performed to demonstrate the practical efficiency of the learning approach.

1 Introduction

An increasing number of AI systems is used to describe, explain or predict human behaviors in evaluation or decision making tasks, but also to help one or more individuals to make a relevant choice in the light of their preferences [1][2]. Intelligent decision systems can also be used to give machines the ability to make sophisticated decisions in complex environments in an autonomous but controlled way [3]. These systems are often grounded on decision theory which proposes preference models that verify normative properties guaranteeing the internal consistency of the value system modeled and the resulting decisions [4][5][6][7][8].

One of the current research challenges in this area is to produce models that are sufficiently expressive to account for a diversity of possible behaviors, but at the same time sufficiently simple to remain interpretable and allow the explanation of evaluations or decisions [9]. There is also a need of automated mechanisms for preference assessment and tailoring general models to specific users [10]. With this in mind, we focus in this paper on learning utility functions to efficiently represent values and preferences on multiattribute objects. For this purpose, the most widely used preference model is the so-called additive utility model [11]. It assumes a special kind of independence among attributes called “mutual preferential independence” which ensures that the overall utility of multiattribute objects can be decomposed as the sum of mono-attribute utility functions. Such decomposability significantly simplifies the construction of the model that can be achieved componentwise. However, in practice, preferential independence is often violated, when interactions among attributes are present, and the simple additive utility model does not fit. This is the reason why, in this paper, we tackle the problem of learning a more general multiattribute utility function in the presence of interacting attributes, with the aim of keeping the model as simple and decomposable as possible.

Let us consider a toy example involving a set of cars described using 3 attributes, namely car make, body color and seat color. Let $X_1 = \{\text{Ferrari, Lamborghini, Porsche}\}$ be the car make domain, $X_2 = \{\text{Blue, Red, Yellow}\}$ be the body color domain, and $X_3 = \{\text{Black, White}\}$ the seat color domain. Assume that an individual (named the Decision Maker hereafter, DM for short) prefers a red body color for a Ferrari but a blue one for a Porsche. Then one cannot build a mono-attribute utility function on body color because the preferences on body colors depend on the car make. This is an example of interaction between two attributes. A similar problem appears when, in addition, the DM prefers a
black seat color for a red car and a white seat color for a blue car. This makes it impossible to define a mono-attribute utility function on the seat color. Hence the DM’s preferences over \( X_1 \times X_2 \times X_3 \) cannot be described by an additive utility function of the form \( u(x_1, x_2, x_3) = u_1(x_1) + u_2(x_2) + u_3(x_3) \) due to presence of interactions among attributes \( \{1, 2\} \) on the one hand, and among attributes \( \{2, 3\} \) on the other. A less decomposable utility function can be used instead, of the form: \( u(x_1, x_2, x_3) = u_{1,2}(x_1, x_2) + u_{2,3}(x_2, x_3) \). Function \( u_{1,2} \) measures the attractiveness of the pair (car make, body color) for the DM while function \( u_{2,3} \) measures car and seats color matching.

We can see that some additive decomposability still remains in this decomposition but utility factors now include several interacting attributes and the factors may partially overlap (here \( x_2 \) appears in both factors). Such a decomposition over overlapping factors is called a GAI decomposition where GAI stands for Generalized Additive Independence [5]. It includes additive and multilinear decompositions as special cases [6], and also multiattribute utility models based on Choquet integrals [12], but it is much more flexible as it does not make any assumption on the kind of interactions between attributes. For this reason, it is widely used for preference modeling and decision support and was the subject of various studies in the AI community [13, 14, 15, 16, 17, 18].

The construction of a GAI utility model from preference information (overall evaluations or pairwise comparisons) remains a challenge. It requires the determination of the relevant factors to be used in the decomposition (groups of interacting attributes) as well as the determination of sub-utility functions on these factors. Some contributions focus on the elicitation of these sub-utility functions, assuming the decomposition of the utility into factors is known [14, 15]. Some other tackle the problem of learning the decomposition. For example, a procedure to learn the GAI model (structure + utility tables) has been proposed [18] in the case of boolean attributes and interactions limited to subsets of bounded size (typically 2 elements). More recently, a procedure to determine a well-formed decomposition of monotonic GAI models was proposed [19] wherein the interactions are limited to pairs of attributes.

However, until now, the learning of general GAI models with no prior assumption on the size of the interacting groups of attributes is still understudied. All the above-mentioned contributions either assume that the structure of the GAI decomposition is known or that it is limited to interactions involving very few attributes. Moreover, most of them only consider the case of finite attribute domains. In this paper, we propose a more general procedure to learn a GAI utility model, kept as simple as possible, with no prior restriction on the size of interactions, and that applies to both continuous and discrete attribute domains.

The paper is organized as follows. In a first section we recall some background on multiattribute utility models and GAI decompositions. Then, in the second section, we present an approach to construct a GAI decomposition from data using multiattribute evaluation vectors. Finally, a third section provides the results of numerical tests on synthetic and real data.

## 2 Background and Notations

Let us consider a finite set \( N = \{1, \ldots, n\} \) of attributes used to describe a set \( X \) of alternatives in a decision problem. Every alternative \( x \) in \( X \) is characterized by a multiattribute evaluation vector \( x = (x_1, \ldots, x_n) \) where \( x_i \) is the evaluation of \( x \) with respect to attribute \( i \), for any \( i \in N \). Let \( X_i \) be the attribute domain of \( i \), i.e., the set of possible values of alternatives on attribute \( i \). This set can be continuous or discrete. The multiattribute description space is denoted \( = X_1 \times \ldots \times X_n \) and \( X \) can be characterized as a subset of \( \). Let \( S \) denote the DM’s preference relation, which is assumed to be a weak order over \( X \).

Under mild assumptions [20], it can be shown that \( S \) is representable by a utility, i.e., there exists a utility function \( u : \rightarrow \mathbb{R} \) such that \( x y \Leftrightarrow u(x) \geq u(y) \) for all \( x, y \in X \). As preferences are specific to each individual, utilities must be elicited using sequences of preference queries or learned using databases of preference examples. Another use of multiattribute utility functions can be to measure the intrinsic value of any alternative for the DM. In this case, the multiattribute utility function can be learned from a labeled data set wherein overall utility values are assigned by the DM to elements of \( X \).

DM’s preferences usually have an underlying structure induced by independence among attributes that substantially decreases the complexity of the model. This simplifies its interpretation and its use for optimization tasks [21]. The simplest case is obtained when values or preferences over \( X_1 \times \ldots \times X_n \) are represented by an additive utility. When DM’s preferences are more complex and may include interaction among attributes, a more elaborate model is needed, as illustrated in the introduction. This motivates the introduction of GAI decompositions:

**Definition 1** Let \( \mathcal{F} \) be a collection of subsets of \( N = \{1, \ldots, n\} \). A utility function \( u(\cdot) \) is GAI-decomposable w.r.t. \( \mathcal{F} \) if there exist functions \( u_S : X_S \rightarrow \mathbb{R} \) such that:

\[
u(x_1, \ldots, x_n) = \sum_{S \in \mathcal{F}} u_S(x_S), \text{ for all } (x_1, \ldots, x_n) \in \mathcal{X},\]

where \( x_S \) is the tuple formed by the \( x_j \)’s, \( j \in S \).
The process can be continued similarly to identify the factors of higher arity. For instance, the ANOVA decomposition

Then, by integrating the same equation over all variables except $u$ variance of $u$

identifiability without any ambiguity. We indeed have $f$ random variables uniformly distributed in $[0, 1]$. In this definition, the multiattribute space $X$ in the form:

An ANOVA decomposition of a function $u(x_1, x_2, x_3, x_4)$ could be seen as the sum of the three following factors: $u_{12}(x_1, x_2) = (x_1 - x_2)^2$, $u_{1,2,3}(x_1, x_2, x_3) = 2x_1(x_2 + x_3)$ and $u_4(x_4)$ or rewritten as the sum of four smaller factors, e.g., $u_1'(x_1) = x_1^2$, $u_2'(x_2) = x_2^2$, $u_3'(x_1, x_3) = 2x_1x_3$ and $u_4'(x_4) = x_4$. The latter decomposition is simpler because it includes factors of smaller arity that are subsets of the factors used in the former decomposition.

In order to further specify what would be a suitable representation of utilities and preferences we might consider well-formed decompositions as proposed in [19].

Definition 2 A GAI decomposition is well-formed if each term $u_S$ appearing in the decomposition satisfies the following conditions:

- each variable in $S$ is active, i.e., the derivative of $u_S$ w.r.t. this variable is not identically 0,
- $u_S$ cannot be further decomposed into terms involving a proper subset of variables

Coming back to Example 1, the decomposition of $u$ into $u_1' + u_2' + u_4'$ is well-formed and refines the decomposition $u = u_{12} + u_{1,2,3} + u_4$ that is therefore not well-formed.

Unfortunately, there may exist multiple well-formed decompositions of the same GAI utility function due to possible utility transfers from a factor to another. This is well illustrated by the following example:

Example 2 $u(x_1, x_2) = x_1 + x_2$ can be rewritten as $u_1''(x_1) + u_2''(x_2)$ with $u_1''(x_1) = (x_1 - \alpha)$ and $u_2''(x_2) = (x_2 + \alpha)$ for any constant $\alpha$, thus providing an infinity of possible decompositions of the same utility function.

In order to avoid such utility transfers and obtain a uniquely defined decomposition of the utility function, we shall consider the ANOVA decomposition as suggested in [23].

Definition 3 An ANOVA decomposition of a function $u(x_1, \ldots, x_n)$ defined on $[0, 1]^n$ and integrable on this domain is a representation of $u$ in the form:

$$u(x) = f_0 + \sum_{S \subseteq N \setminus \emptyset} f_S(x_S)$$

(1)

where $f_S$ are factors such that $\int_0^1 f_S dx_i = 0$ for all non-empty $S$ in $N$ and all $i \in S$.

In this definition, the multiattribute space $X$ is identified to $[0, 1]^n$ for simplicity, as in [23]. This is not restrictive since the attribute domains $X_i$, $i \in N$ can be numerically encoded and normalized. Thus, in the following, the integrals are always computed between 0 and 1.

The name ANOVA comes from ANalysis Of VAriance. Indeed, if $u$ is square integrable and the attributes $x_i$ are random variables uniformly distributed in $[0, 1]$, the ANOVA representation of $u$ allows deriving a decomposition of the variance of $u(x)$.

It is important to note that the ANOVA decomposition of a utility function is uniquely defined, which guarantees its identifiability without any ambiguity. We indeed have $f_0 = \int u(x)dx_1 \ldots dx_n$ by integrating Equation 1 on $[0, 1]^n$. Then, by integrating the same equation over all variables except $x_i$ we obtain $f_i(x_i) = \int u(x)dx_{\bar{i}}$, where $dx_{\bar{i}}$ denotes the integral with respect to all variables except $x_i$. Now, if we integrate Equation 1 on all variables except $x_i$ and $x_j$ we obtain $f_{ij}(x_i, x_j) = \int u(x)dx_{\bar{i}, j}$, where $dx_{\bar{i}, j}$ denotes the integral with respect to all variables except $x_i$ and $x_j$. The process can be continued similarly to identify the factors of higher arity. For instance, the ANOVA decomposition...
of the utility function introduced in Example 2 is:

\[ f_0 = \int_0^1 \int_0^1 (x_1 + x_2)dx_1dx_2 = 1 \]

\[ f_1(x_1) = \int_0^1 (x_1 + x_2)dx_2 - f_0 = x_1 - \frac{1}{2} \]

\[ f_2(x_2) = x_2 - \frac{1}{2} \]

\[ f_{12}(x_1, x_2) = x_1 + x_2 - f_1(x_1) - f_2(x_2) - f_0 = 0 \]

Now, if we consider the model given in Example 1, the same process leads to the following ANOVA decomposition:

\[ f_0 = \frac{5}{3}, \quad f_1(x_1) = x_1 + x_1^2 - \frac{5}{6}, \quad f_2(x_2) = x_2^2 - \frac{1}{3} \]

(2)

\[ f_3(x_3) = x_3 - \frac{3}{7}, \quad f_4(x_4) = x_4 - \frac{1}{2} \]

(3)

\[ f_{13}(x_1, x_3) = 2x_1x_3 - x_1 - x_3 + \frac{3}{7} \]

(4)

In the next section, we propose a method for learning a sparse GAI decomposition of a utility function based on the identification of its ANOVA decomposition. To this end, we use the framework of multiple kernel learning (MKL) that we introduce hereafter.

### 3 A MKL Algorithm for Learning a GAI Decomposition

We want to learn utility function \( u \) from a set of examples \( \{(x^j, y^j), j \in \mathcal{E}\} \) where \( x^j \in \mathcal{X} \) is an alternative and \( y^j \in \mathbb{R} \) denotes its overall evaluation. An efficient approach to learn a multivariate function from regression examples is to use a Support Vector Regression (SVR) algorithm [24]. This algorithm consists in searching \( u \) as a linear model in some high-dimensional space \( \mathcal{H} \), attached to a mapping function \( \phi : \mathbb{R}^n \to \mathcal{H} \) such that \( u(x) = \langle w, \phi(x) \rangle + b \). Function \( \phi \) makes it possible to go beyond the linear model and to fit more complex utility shapes. The learning is performed by minimizing both the error on the regression examples and the oscillation of the utility function (to maximize flatness) in the space generated by \( \phi \). This can be achieved by minimizing the \( L_2 \) norm of \( w \) along with the \( \epsilon \)-sensitive loss as follows:

\[
(P_{SVR}) \quad \text{min } C \sum_{j \in \mathcal{E}} (\epsilon_j^+ + \epsilon_j^-) + \frac{1}{2} \|w\|^2_2
\]

\[
y^j - \langle w, \phi(x^j) \rangle - b \leq \epsilon + \epsilon_j^+, \quad j \in \mathcal{E}
\]

\[
\langle w, \phi(x^j) \rangle + b - y^j \leq \epsilon + \epsilon_j^-, \quad j \in \mathcal{E}
\]

\[
\epsilon_j^+, \epsilon_j^- \geq 0, \quad j \in \mathcal{E}
\]

where \( C > 0 \) controls the tradeoff between flatness and data fitting and \( \epsilon > 0 \) is a tolerance threshold.

The efficiency of this method is due to the compact dual formulation of \( P_{SVR} \), the size of which only depends on the number of regression examples \( |\mathcal{E}| \). It can be obtained using Lagrangian duality and reads as follows:

\[
(D_{SVR}) \quad \text{max } - \sum_{j,i \in \mathcal{E}} (\alpha_j^+ - \alpha_j^-)(\alpha_i^+ - \alpha_i^-)K(x^j, x^i)
\]

\[
+ \sum_{j \in \mathcal{E}} (\alpha_j^+ - \alpha_j^-)y^j - \epsilon \sum_{j \in \mathcal{E}} (\alpha_j^+ + \alpha_j^-)
\]

\[
\sum_{j \in \mathcal{E}} \alpha_j^+ - \alpha_j^- = 0
\]

\[
\alpha_j^+, \alpha_j^- \in [0, C], \quad j \in \mathcal{E}
\]

where \( K(x^j, x^i) = \langle \phi(x^j), \phi(x^i) \rangle \) and \( K \) is referred to as the kernel associated with \( \phi \). This approach relies on polynomial computations of the values \( K(x, x') \) that do not require the computation of the high dimensional vectors.
\( \phi(x) \). For instance, a widely used kernel is the Gaussian radial basis function (RBF) kernel. While involving a projection function \( \phi \) of infinite dimension, it can be simply computed as follows: \( K(x, x') = e^{-\|x-x'\|^2/2\sigma^2} \). More generally, any symmetric positive definite bi-variate function \( K \) can be used since they always represent a scalar product in some feature space \( \mathcal{H} \). Finally, using the KKT conditions, the coefficients \( w \) and the intercept \( b \) can be recovered from the dual optimal variables, and the learned utility function is: \( u(x) = \sum_{j \in \mathcal{E}} (\alpha_j^+ - \alpha_j^-) K(x^j, x) + b \).

Such kernel-based machine learning methods have proved to be effective for utility elicitation from preference statements. In our case, we propose to use the particular framework of multiple kernel learning (MKL) to derive a GAI-decomposition of the utility function \( u \). In general, MKL methods replace the kernel \( K \) with a convex combination of kernels \( K = \sum i d_i K_i \) (the weights \( d_i \) are positive and preserve the property of definite positiveness), in order to either combine multiple kernels with different properties or combine different sources of information. Here, we consider a basis of kernel functions \( (K_S)_{S \subseteq N} \) where for any \( S \subseteq N \setminus \emptyset \), \( K_S \) is a kernel attached to the sub-utility \( u_S \) depending on, and only on, the attributes in \( S \). By convention, \( K_{\emptyset} \) is a constant function equal to one. Let \( d = (d_S)_{S \subseteq N \setminus \emptyset} \) denote the kernel weights and \( d_0 \) the intercept of the model. Since the product of positive definite kernels is also a positive definite kernel, this basis can be constructed using a univariate kernel \( k : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \) and defining \( K_S(x, x') \) as the product \( \prod_{j \in S} k(x_j, x'_j) \) for any \( x, x' \in \mathcal{X} \) and any \( S \subseteq N \setminus \emptyset \).

Note that for \( d = (1, \ldots, 1) \), the kernel \( K = \sum_{S \subseteq N \setminus \emptyset} K_S \) coincides with the well-known ANOVA kernel. However, if the name is related to the ANOVA’s core idea of decomposing a function in terms depending on subsets of variables, this kernel does not provide an ANOVA decomposition since nothing guarantees that \( \int u_S dx_i = 0 \), for any \( S \subseteq N \setminus \emptyset \) and \( i \in S \). Also, the ANOVA kernel uses all the possible subsets of \( \mathbb{N} \) (or all until a given size \( p \)) and does not perform a selection of the most important subsets. In order to select the most useful coalitions and provide a simple ANOVA decomposition, we need to learn a sparse representation of the weight vector \( v \). The objective of selecting only a few kernels in the kernel basis is at the core of MKL and the need for a simultaneous optimization over the weights \( d \) and the dual variables \( \alpha_j^+, \alpha_j^- \) has led to the development of many optimization algorithms (see [32] for an overview). Here, we use \( L_1 \) regularization to obtain sparsity in \( d \), as in [33, 34].

In order to implement the multiple kernel learning approach on problem \( \mathcal{P}_{SVR} \) with the kernel \( K = \sum_{S \subseteq N \setminus \emptyset} d_S K_S \) and the intercept \( b = d_0 \), we first make explicit the associated mapping function of \( K \). Let \( \phi_d \) denote the mapping function associated to the kernel \( K_S \) for any \( S \subseteq N \setminus \emptyset \), then \( \phi = (\sqrt{d_S} \phi_S)_{S \subseteq N \setminus \emptyset} \) is the mapping function associated to \( K \). Indeed, for any \( x, x' \in \mathcal{X} \), we have:

\[
K(x, x') = \sum_{S \subseteq N \setminus \emptyset} d_S K_S(x, x') = \sum_{S \subseteq N \setminus \emptyset} d_S \langle \phi_S(x), \phi_S(x') \rangle = \sum_{S \subseteq N \setminus \emptyset} \langle \sqrt{d_S} \phi_S(x), \sqrt{d_S} \phi_S(x') \rangle = \langle \phi(x), \phi(x') \rangle
\]

Thus, using the kernel \( K \) amounts to searching a utility function of the form \( u(x) = \langle w, \phi(x) \rangle + d_0 = \sum_{S \subseteq N \setminus \emptyset} \sqrt{d_S} \langle w_S, \phi_S(x) \rangle + d_0 \) where \( w = (w_S)_{S \subseteq N \setminus \emptyset} \). Finally, we introduce an \( L_1 \)-regularization term on weight vector \( d \) in the initial problem \( \mathcal{P}_{SVR} \). Since the weights are positive, the \( L_1 \)-penalty is simply the sum of the weights. Then, the obtained approximation problem, denoted \( \mathcal{P} \), is the following:

\[
(P) \quad \min C \sum_{j \in \mathcal{E}} (\epsilon_j^+ + \epsilon_j^-) + \sum_{S \subseteq N \setminus \emptyset} \frac{1}{2} \|w_S\|^2 + \lambda \sum_{S \subseteq N \setminus \emptyset} d_S
\]

\[
y^j - \sum_{S \subseteq N \setminus \emptyset} \sqrt{d_S} \langle w_S, \phi_S(x^j) \rangle - d_0 \leq \epsilon - \epsilon_j^+, j \in \mathcal{E}
\]

\[
\sum_{S \subseteq N \setminus \emptyset} \sqrt{d_S} \langle w_S, \phi_S(x^j) \rangle + d_0 - y^j \leq \epsilon + \epsilon_j^-, j \in \mathcal{E}
\]

\[
\epsilon_j^+, \epsilon_j^- \geq 0, j \in \mathcal{E}, \quad d_S \geq 0, \quad S \subseteq N \setminus \emptyset
\]

where \( \lambda > 0 \) is a regularization hyper-parameter that controls the level of sparsity we want to enforce on \( d \). Similarly to the SVR algorithm, one can apply Lagrangian duality to recover a dual formulation of this problem. Let \( \alpha_j^+ = (\alpha_j^+)_{j \in \mathcal{E}}, \alpha_j^- = (\alpha_j^-)_{j \in \mathcal{E}} \) denote the positive dual variables respectively attached to the sets of constraints (5) and (6) and \( \beta_j^+ = (\beta_j^+)_{j \in \mathcal{E}}, \beta_j^- = (\beta_j^-)_{j \in \mathcal{E}} \) the positive dual variables respectively attached to the sign constraints on the primal variables \( \epsilon_j^+ = (\epsilon_j^+)_{j \in \mathcal{E}}, \epsilon_j^- = (\epsilon_j^-)_{j \in \mathcal{E}} \) and \( d \) (constraints (7)). Let \( I \) denote the vector whose components are all equal to one. Then, the vectorial formulation of the Lagrangian function is \( \mathcal{L} = C(1, \epsilon^+ + \epsilon^-) + \).
\[ \frac{1}{2} \| w \|^2 + \lambda (1, d) + \langle \alpha^+, -\epsilon_1 - \epsilon^+ + Y - \tilde{\phi}w - d_0 1 \rangle + \langle \alpha^-, -\epsilon_1 - \epsilon^- - Y + \tilde{\phi}w + d_0 1 \rangle - \langle \beta^+, \epsilon^+ \rangle - \langle \beta^-, \epsilon^- \rangle - \langle \mu, d \rangle \]

with \( \tilde{\phi} = (\phi(x^i))_{i \in \mathcal{E}} \) the matrix that vertically stacks the vectors \( \phi(x^i) \) and \( Y = (y^j)_{j \in \mathcal{E}} \). Also, the dual problem, denoted \( D \), is the following problem:

\[
\max_{(\alpha^+, \alpha^-, \beta^+, \beta^-, \mu) \in (\mathbb{R}^+)^5} \min_{\epsilon^+, \epsilon^-, w, d_0} \mathcal{L}
\]

where \( t = 4|\mathcal{E}| + 2^n - 1 \) is the number of dual variables. Due to the square roots \( \sqrt{L} \) present in the mapping function \( \phi \), \( \mathcal{L} \) is not differentiable w.r.t. \( d \). However, since \( \mathcal{L} \) is differentiable w.r.t \( \epsilon^+, \epsilon^- \), \( w \) and \( d_0 \), we have the following necessary optimality conditions:

\[
\frac{\partial \mathcal{L}}{\partial w} = w - \tilde{\phi}(\alpha^+ - \alpha^-) = 0 \tag{8}
\]
\[
\frac{\partial \mathcal{L}}{\partial \epsilon^+} = C1 - \alpha^+ - \beta^+ = 0, \quad \frac{\partial \mathcal{L}}{\partial \epsilon^-} = C1 - \alpha^- - \beta^- = 0 \tag{9}
\]
\[
\frac{\partial \mathcal{L}}{\partial d_0} = (1, \alpha^+ - \alpha^-) = 0 \tag{10}
\]

Using Equations 8, 9, and 10 we obtain that for any \( d \): \( g(d) = \min_{\epsilon^+, \epsilon^-} \mathcal{L} = \langle \alpha^+ - \alpha^-, Y \rangle - \epsilon \langle 1, \alpha^+ + \alpha^- \rangle + \sum_{S \subseteq N \setminus \emptyset} d_s (\lambda - \mu - \frac{1}{2} (\alpha^+ - \alpha^-) K_S (\alpha^+ - \alpha^-)) \) with \( K_S = (K_S (x^j, x^i))_{j, i \in \mathcal{E}} \) for any \( S \subseteq N \setminus \emptyset \). Function \( g \) is differentiable w.r.t \( d \), and thus for any \( S \subseteq N \setminus \emptyset \), we can derive the last necessary conditions:

\[
\frac{\partial g}{\partial d_s} = \lambda - \mu s - \frac{1}{2} (\alpha^+ - \alpha^-) K_S (\alpha^+ - \alpha^-) = 0 \tag{11}
\]

Then, using Equations 8, 9, 10, and 11 and the positivity of \( \mu, \beta^+, \beta^- \), the dual Problem \( D \) boils down to the following optimization problem:

\[
\max_{(\alpha^+, \alpha^-, \beta^+, \beta^-, \mu) \in (\mathbb{R}^+)^5} \min_{\epsilon^+, \epsilon^-} \mathcal{L}
\]

Provided that the kernel basis \( (K_S)_{S \subseteq N \setminus \emptyset} \) is constructed with a positive definite univariate kernel \( k \), the matrices \( K_S \) are positive semi-definite, and the dual problem is convex. Then it is a convex quadratically constrained optimization problem and thus can be solved in polynomial time with interior points methods using standard solvers. We can recover the optimal weight values \( d_s \) for any \( S \subseteq N \setminus \emptyset \) and the intercept \( d_0 \) by respectively accessing the dual values of the set of constraints 12 and 13. Finally, using Equation 8 we recover the learned utility function as follows:

\[
u(x) = \sum_{S \subseteq N \setminus \emptyset} d_s \sum_{j \in \mathcal{E}} (\alpha^+_j - \alpha^-_j) K_S (x^j, x^i) + d_0
\]

with \( u_S (x_s) = d_s \sum_{j \in \mathcal{E}} (\alpha^+_j - \alpha^-_j) K_S (x^j, x^i) \), the sub-utility attached to the group of attributes \( S \). Then the presence of a sub-utility \( u_S \) in the decomposition of the utility function \( u \) is equivalent to a non-null weight \( d_s \). We thus recover simple decompositions by increasing the \( L_1 \)-penalty hyper-parameter \( \lambda \) in problem \( D \).

### 3.1 Retrieving an ANOVA Decomposition

Finally, we guarantee the identification of an ANOVA decomposition by constraining the model so that \( \int u_S dx_i = 0 \) holds for all sub-utility factors \( u_S, S \subseteq N \setminus \emptyset \) and all \( i \in S \). This is achieved by constructing an univariate kernel \( k^0(x, y) \) the integral of which (w.r.t \( x \) or \( y \)) equals zero, as proposed in [35]:

\[
k^0(x, y) = k(x, y) - \frac{\int k(s, y) ds \int k(x, t) dt}{\int s \int k(s, t) ds dt}
\]

Then for any \( x \in \mathbb{R}, \int k^0(x, s) ds = 0 \). Also, under mild hypothesis on kernel \( k \), it can be shown that the attached kernel \( k^0 \) is still positive definite [35]. Then, we can construct the associated basis of kernels \( (K^0_S)_{S \subseteq N \setminus \emptyset} \), defined by:

\[
K^0_S (x_s, y_s) = \prod_{i \in S} k^0(x_i, y_i)
\]
We first show the result of the learning on synthetic data generated using a 6-dimensional model \( u(x) = u_1(x_1) + u_2(x_2) + u_{34}(x_3, x_4) + u_{45}(x_4, x_5) \). The factors \( u_1 \) and \( u_2 \) are taken as quadratic spline functions and \( u_{34} \) and \( u_{45} \) are bivariate tensor products of quadratic splines. To generate the factors we use a basis of B-splines functions of zero order, \( \sigma = 1 \) is set to 0.01 and the regularization hyper-parameters \( C \) and \( \lambda \) are selected by cross-validation using a number of folds equal to 3. All tests are conducted on a 2.8 GHz Intel Core i7 processor with 16GB RAM and we used the mathematical programming Gurobi solver (version 9.1.2).

## 4 Numerical Experiments

This section presents the results of numerical tests performed on synthetic and real-world preference data. We implement our method, called SMKGAI for Sparse Multiple Kernel GAI, with the Gaussian RBF kernel using \( \sigma = 1 \). The tolerance threshold \( \epsilon \) is set to 0.01 and the regularization hyper-parameters \( C \) and \( \lambda \) are selected by cross-validation using a number of folds equal to 3. All tests are conducted on a 2.8 GHz Intel Core i7 processor with 16GB RAM and we used the mathematical programming Gurobi solver (version 9.1.2).

### 4.1 Synthetic Data

We first show the result of the learning on synthetic data generated using a 6-dimensional model \( u(x) = u_1(x_1) + u_2(x_2) + u_{34}(x_3, x_4) + u_{45}(x_4, x_5) \). The factors \( u_1 \) and \( u_2 \) are taken as quadratic spline functions and \( u_{34} \) and \( u_{45} \) are bivariate tensor products of quadratic splines. To generate the factors we use a basis of B-splines functions of zero order forming an ANOVA decomposition of \( u \). We generate a set \( E \) of overall evaluations of size 70 from the hidden utility function \( u \), with a random uniform draw of alternatives \( x^j \in [0, 1]^n \). The data is then perturbated with a centered Gaussian noise with standard error \( \sigma = 0.05 \). In Figure 1 we represent the ground truth utility factors in grey and the learned utility factors in green.

Secondly, we conduct an experiment using a model with a high degree of interaction: \( u(x) = \sum_{i=1}^n x_i + 1000 \prod_{i=1}^n x_i \). In order to assess the benefit of allowing high interactions in the learning of a GAI decomposition, we compare SMKGAI with \( p \)-additive GAI utilities that do not use \( L_1 \) regularization to select the most useful factors but that include factors of size at most \( p \) for \( p \in \{1, 2, 3, 4\} \). The case \( p = 1 \) corresponds to the learning of an additive utility. This is done using the SVR algorithm, i.e., by solving the dual formulation of the initial problem \( P_{\text{SVR}} \), and using the ANOVA kernel of degree \( p \): \( K = \sum_{S \subseteq N, |S| \leq p} K^0_S \). As in the previous experiment, we generate, from the hidden utility function \( u \), random training sets \( E \) of noisy overall evaluations of size 70. In Table 1 we compare the generalizing performances of SMKGAI and the one obtained with dense \( p \)-additive GAI models (\( p \)-GAI) over 20
simulations. The generalized performances are measured as the mean absolute errors (MAE), i.e. the average absolute differences between the normalized ground truth utility and the predicted utility over test sets of size 150. We also provide the computing times (sec.) for all the methods. We observe that SMKGAI, by capturing the interaction of size \( n \), nearly divides by 10 the MAE compared to the \( p \)-additive models. This is achieved for a time that is only multiplied by 2 compared to the additive model (1-GAI).

Finally, we perform an experiment on synthetic data generated with more general models for \( n = 10 \). The models are randomly generated as sums of 10 tensor products of quadratic splines. In order to increase the complexity of the hidden models, the maximal size of the factors (max. size) is increased from 1 (additive utility) to 5. We perform 20 simulations and each time, we generate a set \( \mathcal{E} \) of overall evaluations of size 140 perturbated with a Gaussian centered noise of standard error \( \sigma = 0.05 \). In Table 3 is represented the MAE on test sets of size 150 along with the maximal size of the learned factors and the False Discovery Rate (FDR), which is computed as the percentage of selected factors in the learned ANOVA decomposition that are not included in any of the factors of the hidden function. We consider that a factor \( S \subseteq N \) is selected as soon as the attached weight \( d_S \) is higher than 0.01. As expected, we observe that the MAE increases as the interaction degree (max size) of the hidden model increases. However, our learning approach is able to capture these interactions since the maximal size of the learned factors increases similarly to the ground truth, with a percentage of false inclusion (FDR) in the model that stays below 20%.

### Table 1: Comparisons of the generalizing performances (MAE) and training times over 20 simulations for the product model.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SMKGAI</th>
<th>1-GAI</th>
<th>2-GAI</th>
<th>3-GAI</th>
<th>4-GAI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training time (sec.)</td>
<td>75.48 ± 6.78</td>
<td>34.89 ± 2.35</td>
<td>42.92 ± 3.35</td>
<td>58.01 ± 4.73</td>
<td>70.75 ± 7.06</td>
</tr>
<tr>
<td>MAE</td>
<td>0.0065 ± 0.0016</td>
<td>0.0381 ± 0.0088</td>
<td>0.0456 ± 0.0106</td>
<td>0.0560 ± 0.0129</td>
<td>0.0543 ± 0.0121</td>
</tr>
</tbody>
</table>

### Table 2: Mean absolute error (MAE) averaged over 20 random splits for SMKGAI and baseline methods.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Linear Regression</th>
<th>2-add Choquet Integral</th>
<th>1-GAI</th>
<th>2-GAI</th>
<th>SMKGAI</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESL</td>
<td>0.0819 ± 0.00585</td>
<td>0.08266 ± 0.00619</td>
<td>0.08426 ± 0.00725</td>
<td>0.08419 ± 0.00697</td>
<td></td>
</tr>
<tr>
<td>LEV</td>
<td>0.23540 ± 0.00554</td>
<td>0.25147 ± 0.01449</td>
<td>0.23163 ± 0.01295</td>
<td>0.17093 ± 0.01212</td>
<td>0.12381 ± 0.02662</td>
</tr>
<tr>
<td>ERA</td>
<td>0.24255 ± 0.00536</td>
<td>0.24256 ± 0.00698</td>
<td>0.20071 ± 0.01115</td>
<td>0.11817 ± 0.00932</td>
<td>0.04749 ± 0.00200</td>
</tr>
<tr>
<td>CPU</td>
<td>0.02795 ± 0.00357</td>
<td>0.01792 ± 0.00654</td>
<td>0.00888 ± 0.00209</td>
<td>0.00796 ± 0.00475</td>
<td>0.00805 ± 0.00455</td>
</tr>
<tr>
<td>MPG</td>
<td>0.06399 ± 0.00331</td>
<td>0.10113 ± 0.00557</td>
<td>0.05628 ± 0.01115</td>
<td>0.05355 ± 0.00656</td>
<td>0.05212 ± 0.00666</td>
</tr>
<tr>
<td>CITY</td>
<td>0.06282 ± 0.00861</td>
<td>0.06707 ± 0.00768</td>
<td>0.04855 ± 0.00916</td>
<td>0.05127 ± 0.00899</td>
<td>0.05054 ± 0.00893</td>
</tr>
</tbody>
</table>

Figure 1: Learned and ground truth utility factors \( u_1 \) (top left), \( u_2 \) (top right), \( u_{34} \) (bottom left) and \( u_{45} \) (bottom right).
4.2 Real-world Datasets

In this subsection, we test our method on real preference datasets. We use standard multi-criteria decision-making benchmarks containing overall evaluations of alternatives described by continuous or discrete attributes. We use Employee Selection (ESL) which contains profiles and overall psychological evaluations of job candidates, Lecture Evaluation (LEV), containing examples of anonymous lecturer evaluations and Employee Rejection/Acceptance (ERA) which contains the judgment of a decision-maker w.r.t candidate profiles. Then from the UCI repository, we use CPU and Car MPG (MPG) which respectively contain the performances of CPU and the fuel consumption of cars, along with attributes describing the objects. Finally, we use the Movehub city ranking (CITY) dataset which contains overall evaluations of cities quality. The number of evaluations $|E|$ and the number of attributes $n$ of each dataset is given in Table 4.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ESL</th>
<th>LEV</th>
<th>ERA</th>
<th>CPU</th>
<th>MPG</th>
<th>CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>$</td>
<td>E</td>
<td>$</td>
<td>488</td>
<td>1000</td>
<td>1000</td>
<td>209</td>
</tr>
</tbody>
</table>

Table 4: Datasets’ number of attributes $n$ and examples $|E|$.

We compare SMKGAI to standard baselines from preference modeling such as the linear regression, the 2-additive Choquet Integral [12], and $p$-additive GAI ($p$-GAI) for $p = 1$ and $p = 2$. The attribute values are normalized using a linear max-min normalization. Each dataset is split to produce a training set containing 80% of the examples and a test set with the 20% left. For 20 random splits, we compute the MAE obtained on the test set for each method and present the averaged results in Table 2. For each dataset, the best result is displayed in bold and if there is another performance close to this result, it is also displayed in bold. We can see that SMKGAI is attached to the best average MAE or is very close to the optimal result, except on the dataset ESL where the linear regression and the additive utility (1-GAI) provide the best results. In particular, for the datasets LEV and ERA, SMKGAI outperforms the baseline methods, showing the presence of interactions between more than two attributes in the data. Also, for the datasets CPU, MPG and CITY, it seems that SMKGAI is able to adapt its complexity to the underlying data since it provides results similar to the additive utility (1-GAI) or 2-additive GAI (2-GAI) depending on the case.

5 Conclusion

We have presented a multiple kernel learning approach that constructs a sparse GAI model from evaluation examples to describe and explain the value system of a decision maker. The core of the approach relies on the determination of a sparse ANOVA decomposition of utilities obtained thanks to the use of zero integral kernels. This decomposition can be simplified afterward to produce a more compact and still well-formed GAI decomposable utility that well fits the available preference information. The advantage of the proposed approach is to be able to capture general interactions among continuous or discrete attributes without prior restrictions on the size of interacting factors. It makes it possible to fit model complexity to the available preference information. The regularization used in the objective function ensures that model complexity is kept as low as possible, given the descriptive constraints imposed by preference data. As far as we know, this is the first learning method able to learn both the structure of the GAI decomposition (by identification of the factors that really matter), and the utility functions defined on these factors, that can handle continuous attributes and that does not use prior restrictions on the cardinality of the interactions. Also, note that the same approach can be easily implemented to extract the utility function from pairwise preference/indifference examples.

---

1. [www.openml.org](http://www.openml.org) (ESL, LEV and ERA)
In order to go further, some directions are worth investigating. In particular, for an interaction term \( f_S \) present in the ANOVA decomposition, it is likely that the subterms \( f_{S'} \) for \( S' \subseteq S \) also appear in the ANOVA decomposition. Thus, it could be of interest to implement a grouped hierarchical regularization [37] that would simultaneously cancel \( f_S \) and its subterms \( f_{S'} \), \( S' \subseteq S \), as suggested in [35]. Another direction is to enhance the scalability of the method w.r.t the number of attributes, since the number of possible factors grows exponentially. One path could be to bound from above the size of possible interacting factors. Another possible path is to use an iterative optimization procedure based on projected gradient descent that successively optimizes over the variables \( \alpha_j^+, \alpha_j^- \) and the weights \( d \), as proposed in [38].

Acknowledgments

This work is supported by the ANR project ANR-20-CE23-0018 THEMIS of the French National Research Agency.

References


