Learning GAI-decomposable Utility Models for Multiattribute Decision Making

Margot Herin, Patrice Perny, Nataliya Sokolovska

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.

Les 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 multiple kernel learning. 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 denote the DM's preference relation, which is assumed to be a weak order over .

Under mild assumptions [20], it can be shown that is representable by a utility, i.e., there exists a utility function $u :\mapsto$ such that $xy \Leftrightarrow u(x) \geq u(y)$ for all $x, y \in$. 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 .

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 $u_S : X_S \to, S \in \mathcal{F}$ such that:*

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

where x_S is the tuple formed by the x_j's, $j \in S$.

Note that, in the above definition, the condition $N = \bigcup_{S \in \mathcal{F}} S$ is not required because some components x_i in the model may appear to be unessential to explain the available preference information and could be removed to produce a more compact model.

Given a multiattribute utility function u defined on $\mathcal{X}$ there may exist multiple distinct GAI decompositions of this function. This raises a problem of identifiability [22], as shown in the following example:

Example 1 *Function $u(x_1, x_2, x_3, x_4) = (x_1 - x_2)^2 + 2x_1(x_2 + x_3) + x_4$ 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 $u_4(x_4) = x_4$ or rewritten as the sum of four smaller factors, e.g., $u'_1(x_1) = x_1^2$, $u'_2(x_2) = x_2^2$, $u'_{13}(x_1, x_3) = 2x_1x_3$ 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 $u'_1 + u'_2 + u'_{13} + u'_4$ is well-formed and refines the decomposition $u = u_{12} + u_{123} + u_4$ 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 $u_1^\alpha(x_1) + u_2^\alpha(x_2)$ with $u_1^\alpha(x_1) = (x_1 - \alpha)$ and $u_2^\alpha(x_2) = (x_2 + \alpha)$ for any constant α, 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:*

$$u(x) = f_\emptyset + \sum_{S \subseteq N \setminus \emptyset} f_S(x_S) \tag{1}$$

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 $f_\emptyset = \int u(x) dx_1 \ldots dx_n$ by integrating Equation 1 on $[0, 1]^n$. Then, by integrating the same equation over all variables except x_i we obtain $f_i(x_i) = \int u(x) \Pi_{k \neq i} dx_k - f_\emptyset$. Now, if we integrate Equation 1 on all variables except x_i and x_j we obtain $f_{ij}(x_i, x_j) = \int u(x) \Pi_{k \neq i,j} dx_k - f_i(x_i) - f_j(x_j) - f_\emptyset$. 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:

$$\begin{aligned} f_\emptyset &= \int_0^1 \int_0^1 (x_1 + x_2) dx_1 dx_2 = 1 \\ f_1(x_1) &= \int_0^1 (x_1 + x_2) dx_2 - f_\emptyset = x_1 - \frac{1}{2} \\ f_2(x_2) &= x_2 - \frac{1}{2} \\ f_{12}(x_1, x_2) &= x_1 + x_2 - f_1(x_1) - f_2(x_2) - f_\emptyset = 0 \end{aligned}$$

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

$$f_\emptyset = \frac{5}{3},\ f_1(x_1) = x_1 + x_1^2 - \frac{5}{6},\ f_2(x_2) = x_2^2 - \frac{1}{3} \tag{2}$$

$$f_3(x_3) = x_3 - \frac{1}{2},\ f_4(x_4) = x_4 - \frac{1}{2} \tag{3}$$

$$f_{13}(x_1, x_3) = 2x_1x_3 - x_1 - x_3 - \frac{1}{3} \tag{4}$$

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 ϕ 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 ϕ. This can be achieved by minimizing the L_2 norm of w along with the $\epsilon-$sensitive loss as follows:

$$\begin{aligned} (\mathcal{P}_{SVR}) \quad \min \quad & C \sum_{j \in \mathcal{E}} (\epsilon_j^+ + \epsilon_j^-) + \frac{1}{2} \|w\|_2^2 \\ & y^j - \langle w, \phi(x^j) \rangle - b \leq \epsilon + \epsilon_j^+, \quad j \in \mathcal{E} \\ & \langle w, \phi(x^j) \rangle + b - y^j \leq \epsilon + \epsilon_j^-, \quad j \in \mathcal{E} \\ & \epsilon_j^+, \epsilon_j^- \geq 0, \quad j \in \mathcal{E} \end{aligned}$$

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 $\mathcal{P}_{SVR}$, 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:

$$\begin{aligned} (\mathcal{D}_{SVR}) \quad \max \quad & -\frac{1}{2} \sum_{j,l \in \mathcal{E}} (\alpha_j^+ - \alpha_j^-)(\alpha_l^+ - \alpha_l^-) K(x^j, x^l) \\ & + \sum_{j \in \mathcal{E}} (\alpha_j^+ - \alpha_j^-) y^j - \epsilon \sum_{j \in \mathcal{E}} (\alpha_j^+ + \alpha_j^-) \\ & \sum_{j \in \mathcal{E}} \alpha_j^+ - \alpha_j^- = 0 \\ & \alpha_j^+, \alpha_j^- \in [0, C], \quad j \in \mathcal{E} \end{aligned}$$

where $K(x^j, x^l) = \langle \phi(x^j), \phi(x^l) \rangle$ and K is referred to as the kernel associated with ϕ. 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 ϕ of infinite dimension, it can be simply computed as follows: $K(x, x') = e^{-\|x-x'\|_2^2/2\sigma^2}$. More generally, any symmetric positive definite bi-variate function K can be used since they always represent a scalar product in some feature space $\mathcal{H}$ [25]. 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 [26, 27, 28]. In our case, we propose to use the particular framework of multiple kernel learning (MKL) [29] to derive a GAI-decomposition of the utility function u. In general, MKL methods replace the kernel K with a convex combination of kernels $\boldsymbol{K} = \sum_l d_l K_l$ (the weights d_l 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 $d_\emptyset$ 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 $\boldsymbol{K} = \sum_{S\subseteq N\setminus\emptyset} K_S$ coincides with the well-known ANOVA kernel [30, 31]. 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 $i \in S$ (See Definition 3). Also, the ANOVA kernel uses all the possible subsets of 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 d. 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 α_j^+, α_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 $\boldsymbol{K} = \sum_{S\subseteq N\setminus\emptyset} d_S K_S$ and the intercept $b = d_\emptyset$, we first make explicit the associated mapping function of $\boldsymbol{K}$. Let ϕ_S denote the mapping function attached to the kernel K_S for any $S \subseteq N \setminus \emptyset$, then $\boldsymbol{\phi} = (\sqrt{d_S}\phi_S)_{S\subseteq N\setminus\emptyset}$ is the mapping function associated to $\boldsymbol{K}$. Indeed, for any $x, x' \in \mathcal{X}$, we have:

$$\begin{aligned}\boldsymbol{K}(x, x') &= \sum_{S\subseteq N\setminus\emptyset} d_S K_S(x, x') = \sum_{S\subseteq N\setminus\emptyset} d_S\langle\phi_S(x), \phi_S(x')\rangle \\ &= \sum_{S\subseteq N\setminus\emptyset} \langle\sqrt{d_S}\phi_S(x), \sqrt{d_S}\phi_S(x')\rangle = \langle\boldsymbol{\phi}(x), \boldsymbol{\phi}(x')\rangle\end{aligned}$$

Thus, using the kernel $\boldsymbol{K}$ amounts to searching a utility function of the form $u(x) = \langle\boldsymbol{w}, \boldsymbol{\phi}(x)\rangle + d_\emptyset = \sum_{S\subseteq N\setminus\emptyset} \sqrt{d_S}\langle w_S, \phi_S(x)\rangle + d_\emptyset$ where $\boldsymbol{w} = (w_S)_{S\subseteq N\setminus\emptyset}$. 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:

$$(\mathcal{P}) \quad \min\ C\sum_{j\in\mathcal{E}}(\epsilon_j^+ + \epsilon_j^-) + \sum_{S\subseteq N\setminus\emptyset}\frac{1}{2}\|w_S\|_2^2 + \lambda\sum_{S\subseteq N\setminus\emptyset} d_S$$

$$y^j - \sum_{S\subseteq N\setminus\emptyset}\sqrt{d_S}\langle w_S, \phi_S(x^j)\rangle - d_\emptyset \leq \epsilon + \epsilon_j^+, j \in \mathcal{E} \tag{5}$$

$$\sum_{S\subseteq N\setminus\emptyset}\sqrt{d_S}\langle w_S, \phi_S(x^j)\rangle + d_\emptyset - y^j \leq \epsilon + \epsilon_j^-, j \in \mathcal{E} \tag{6}$$

$$\epsilon_j^+, \epsilon_j^- \geq 0, j \in \mathcal{E},\ \ d_S \geq 0,\ \ S \subseteq N \setminus \emptyset \tag{7}$$

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 $\alpha^+ = (\alpha_j^+)_{j\in\mathcal{E}}, \alpha^- = (\alpha_j^-)_{j\in\mathcal{E}}$ denote the positive dual variables respectively attached to the sets of constraints (5) and (6) and $\beta^+ = (\beta_j^+)_{j\in\mathcal{E}}, \beta^- = (\beta_j^-)_{j\in\mathcal{E}}, \mu = (\mu_S)_{S\subseteq N\setminus\emptyset}$ the positive dual variables respectively attached to the sign constraints on the primal variables $\epsilon^+ = (\epsilon_j^+)_{j\in\mathcal{E}}, \epsilon^- = (\epsilon_j^-)_{j\in\mathcal{E}}$ and d (constraints (7)). Let $\mathbf{1}$ denote the vector whose components are all equal to one. Then, the vectorial formulation of the Lagrangian function is $\mathcal{L} = C\langle\mathbf{1}, \epsilon^+ + \epsilon^-\rangle +$

$\frac{1}{2}\|\boldsymbol{w}\|_2^2+\lambda\langle\mathbf{1},d\rangle+\langle\alpha^+,-\epsilon\mathbf{1}-\epsilon^++Y-\tilde{\boldsymbol{\phi}}\boldsymbol{w}-d_\emptyset\mathbf{1}\rangle+\langle\alpha^-,-\epsilon\mathbf{1}-\epsilon^--Y+\tilde{\boldsymbol{\phi}}\boldsymbol{w}+d_\emptyset\mathbf{1}\rangle-\langle\beta^+,\epsilon^+\rangle-\langle\beta^-,\epsilon^-\rangle-\langle\mu,d\rangle$ with $\tilde{\boldsymbol{\phi}}=(\boldsymbol{\phi}(x^j))_{j\in\mathcal{E}}$ the matrix that vertically stacks the vectors $\boldsymbol{\phi}(x^j)$ and $Y=(y^j)_{j\in\mathcal{E}}$. Also, the dual problem, denoted $\mathcal{D}$, is the following problem:

$$(\mathcal{D})\quad \max_{(\alpha^+,\alpha^-,\beta^+,\beta^-,\mu)\in(\mathbb{R}^+)^t}\quad \min_{\epsilon^+,\epsilon^-,\boldsymbol{w},d,d_\emptyset}\mathcal{L}$$

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

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}}=\boldsymbol{w}-\tilde{\boldsymbol{\phi}}^\top(\alpha^+-\alpha^-)=0 \tag{8}$$

$$\frac{\partial\mathcal{L}}{\partial\epsilon^+}=C\mathbf{1}-\alpha^+-\beta^+=0,\ \ \frac{\partial\mathcal{L}}{\partial\epsilon^-}=C\mathbf{1}-\alpha^--\beta^-=0 \tag{9}$$

$$\frac{\partial\mathcal{L}}{\partial d_\emptyset}=\langle\mathbf{1},\alpha^+-\alpha^-\rangle=0 \tag{10}$$

Using Equations 8, 9 and 10, we obtain that for any d: $g(d)=\min_{\epsilon^+,\epsilon^-,\boldsymbol{w},d_\emptyset}\mathcal{L}=\langle\alpha^+-\alpha^-,Y\rangle-\epsilon\langle\mathbf{1},\alpha^++\alpha^-\rangle+\sum_{S\subseteq N\setminus\emptyset}d_S(\lambda-\mu_S-\frac{1}{2}(\alpha^+-\alpha^-)^\top\tilde{K}_S(\alpha^+-\alpha^-))$ with $\tilde{K}_S=(K_S(x^j,x^l))_{j,l\in\mathcal{E}}$ 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:

$$\frac{\partial g}{\partial d_S}=\lambda-\mu_S-\frac{1}{2}(\alpha^+-\alpha^-)^\top\tilde{K}_S(\alpha^+-\alpha^-)=0 \tag{11}$$

Then, using Equations 8, 9, 10 and 11, and the positivity of μ,β^+,β^-, the dual Problem $\mathcal{D}$ boils down to the following optimization problem:

$$\begin{aligned}(\mathcal{D})\quad &\max\langle\alpha^+-\alpha^-,Y\rangle-\epsilon\langle\mathbf{1},\alpha^++\alpha^-\rangle\\ &\lambda-\frac{1}{2}(\alpha^+-\alpha^-)^\top\tilde{K}_S(\alpha^+-\alpha^-)\geq 0, S\subseteq N\setminus\emptyset \qquad (12)\\ &\langle\alpha^+-\alpha^-,\mathbf{1}\rangle=0, \qquad (13)\\ &\alpha_j^+,\alpha_j^-\in[0,C],\ \ j\in\mathcal{E}\end{aligned}$$

Provided that the kernel basis $(K_S)_{S\subseteq N\setminus\emptyset}$ is constructed with a positive definite univariate kernel k, the matrices $\tilde{K}_S$ 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 $d_\emptyset$ 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:

$$\begin{aligned}u(x)&=\sum_{S\subseteq N\setminus\emptyset}d_S\sum_{j\in\mathcal{E}}(\alpha_j^+-\alpha_j^-)K_S(x_S^j,x_S)+d_\emptyset\\ &=\sum_{S\subseteq N\setminus\emptyset}u_S(x_S)+d_\emptyset\end{aligned}$$

with $u_S(x_s)=d_S\sum_{j\in\mathcal{E}}(\alpha_j^+-\alpha_j^-)K_S(x_S^j,x_S)$, 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 λ in problem $\mathcal{D}$.

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]:

$$k^0(x,y)=k(x,y)-\frac{\int_s k(s,y)ds\int_t k(x,t)dt}{\int_s\int_t k(s,t)dsdt} \tag{14}$$

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 $(K_S^0)_{S\subseteq N\setminus\emptyset}$, defined by:

$$K_S^0(x_s,y_s)=\prod_{i\in S}k^0(x_i,y_i)$$

For any $x, y \in [0,1]^n$, $S \subseteq N \setminus \emptyset$ and $i \in S$ we have:

$$\int K_S^0(x_S, y_S)dy_i = \prod_{j \in S \setminus \{i\}} k^0(x_j, y_j) \int k^0(x_i, y_i)dy_i = 0$$

Then, using the centered kernel basis $(K_S^0)_{S \subseteq N \setminus \emptyset}$, each sub-utility $u_S(x_S) = d_S \sum_{j \in \mathcal{E}} (\alpha_j^+ - \alpha_j^-) K_S^0(x_S^j, x_S)$ is guaranteed to have a zero integral with respect to x_i whatever $i \in S$. Thus, the obtained decomposition is an ANOVA decomposition.

Remark that, in addition to be positive definite, the univariate kernel k is now required to verify $\int k(s,x)ds < \infty$ for any $x \in \mathcal{X}$ and $\int \int k(s,t)dsdt < \infty$ so that k^0 is well defined (See Equation 14). Also, in practice, these integrals have to be approximated with numerical integration. To avoid unnecessary repeated computations, we use a discretized representation of function $x \mapsto \int k(s,x)ds$ that has been computed beforehand. Standard examples of univariate kernels that can be used here are the Gaussian RBF kernel $k(x,y) = e^{-(x-y)^2/2\sigma^2}$, the odd order B-spline kernel $k(x,y) = B_{2m+1}(x-y), m \in \mathbb{N}$ or the first order infinite spline kernel $k(x,y) = 1 + xy + \frac{1}{2}|x-y|\min(x,y)^2 + \frac{1}{3}\min(x,y)^3$ [25, 36].

As mentioned before, the advantage of the ANOVA decomposition is that it is uniquely defined for a given u and its factors can be learned from examples as explained above. However, although the use of penalization promotes the emergence of sparse ANOVA representations, the obtained representation can be further simplified by keeping only the maximal factors (i.e., factors having a maximal scope with respect to set inclusion), the others being included as subsets. More precisely, the non-maximal factors can be aggregated with the maximum factors containing them to reduce the number of factors without loosing the well-formed nature of the decomposition (see Definition 2). In the presence of two or more maximal factors with a non-empty intersection S, the factors whose scope is included in S could be indifferently aggregated to any maximal factor containing them or spread into all or part of them. Whatever the chosen option, it yields to a well-formed GAI decomposition of u that is possibly more compact than the learned ANOVA decomposition. Multiple GAI representations are therefore possible, sharing the same maximal factors as ANOVA decomposition.

For the sake of illustration, let us come back to Example 1 and consider the ANOVA decomposition given in Equations (2-4). By aggregating $f_1(x_1)$ and $f_3(x_3)$ to $f_{13}(x_1, x_3)$ and remarking that the sum of constant terms equals zero, function u can be rewritten, after simplification, as the sum of 3 terms: $u_2''(x_2) = x_2^2, u_4''(x_4) = x_4$ and $u_{13}''(x_1, x_3) = x_1^2 + 2x_1x_3$.

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 ϵ is set to 0.01 and the regularization hyper-parameters C and λ 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 integral forming an ANOVA decomposition of u. We generate a set $\mathcal{E}$ 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: $u(x) = \sum_{i=1}^n x_i + 1000 \prod_{i=1}^n x_i$ for $n = 6$. 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 $\mathcal{P}_{SVR}$, and using the ANOVA kernel of degree p: $K = \sum_{S \subseteq N, |S| \leq p} K_S^0$. As in the previous experiment, we generate, from the hidden utility function u, random training sets $\mathcal{E}$ 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

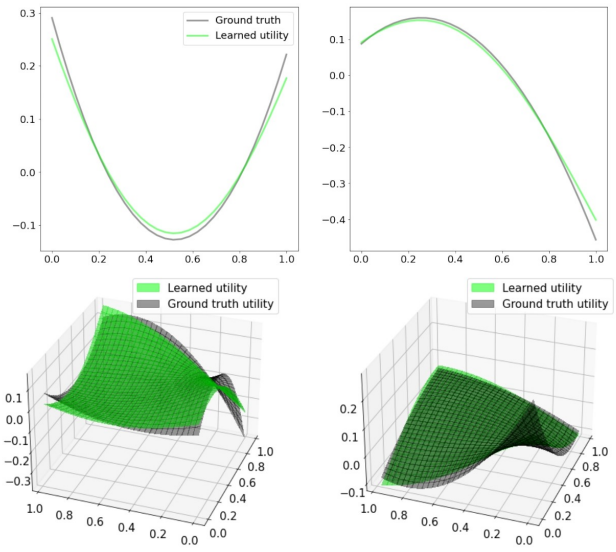

Figure 1: Learned and ground truth utility factors u_1(top left), u_2 (top right), u_{34} (bottom left) and u_{45} (bottom right).

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).

	SMKGAI	1-GAI	2-GAI	3-GAI	4-GAI
MAE	$\mathbf{0.0065 \pm 0.0016}$	0.0381 ± 0.0088	0.0456 ± 0.0106	0.0560 ± 0.0129	0.0543 ± 0.0121
Training time (sec.)	75.48 ± 6.78	$\mathbf{34.89 \pm 2.35}$	42.92 ± 3.35	58.01 ± 4.73	70.75 ± 7.06

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

Dataset	Linear Regression	2-add Choquet Integral	1-GAI	2-GAI	SMKGAI
ESL	$\mathbf{0.08198 \pm 0.00588}$	0.08488 ± 0.00498	$\mathbf{0.08266 \pm 0.00619}$	0.08426 ± 0.00725	0.08419 ± 0.00697
LEV	0.23540 ± 0.00551	0.25417 ± 0.01449	0.23163 ± 0.01295	0.17093 ± 0.01212	$\mathbf{0.12381 \pm 0.02662}$
ERA	0.24255 ± 0.00536	0.24256 ± 0.00698	0.20071 ± 0.01115	0.11817 ± 0.00932	$\mathbf{0.04749 \pm 0.00200}$
CPU	0.02795 ± 0.00357	0.01792 ± 0.00654	0.00888 ± 0.00209	$\mathbf{0.00796 \pm 0.00475}$	$\mathbf{0.00805 \pm 0.00455}$
MPG	0.06399 ± 0.00331	0.10113 ± 0.00557	0.05628 ± 0.01115	$\mathbf{0.05355 \pm 0.00656}$	$\mathbf{0.05212 \pm 0.00666}$
CITY	0.06282 ± 0.00861	0.06707 ± 0.00768	$\mathbf{0.04855 \pm 0.00916}$	0.05127 ± 0.00899	$\mathbf{0.05054 \pm 0.00893}$

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

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%.

Max size (true)	MAE	Max size	FDR
1	0.026 ± 0.012	1.0 ± 0.0	0.0 ± 0.0
2	0.036 ± 0.012	2.0 ± 0.4	0.011 ± 0.033
3	0.067 ± 0.013	3.0 ± 1.4	0.078 ± 0.115
4	0.085 ± 0.020	4.1 ± 1.0	0.198 ± 0.165
5	0.082 ± 0.015	4.2 ± 1.2	0.156 ± 0.124

Table 3: Model recovery assessment for growing interaction degree of the hidden models in average over 20 simulations.

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)[1], 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*[2] (CITY) dataset which contains overall evaluations of cities quality. The number of evaluations $|\mathcal{E}|$ and the number of attributes n of each dataset is given in Table 4.

Dataset	ESL	LEV	ERA	CPU	MPG	CITY
n	4	4	4	6	7	5
$\lvert\mathcal{E}\rvert$	488	1000	1000	209	392	216

Table 4: Datasets' number of attributes n and examples $\mathcal{E}$.

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 (ESL, LEV and ERA)

[2] www.kaggle.com/datasets/blitzr/movehub-city-rankings

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 sub terms $f_{S'}, S' \subseteq S$, 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 α_j^+, α_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

[1] Carmel Domshlak, Eyke Hüllermeier, Souhila Kaci, and Henri Prade. Preferences in ai: An overview. *Artificial Intelligence*, 175(7-8):1037–1052, 2011.

[2] Gabriella Pigozzi, Alexis Tsoukias, and Paolo Viappiani. Preferences in artificial intelligence. *Annals of Mathematics and Artificial Intelligence*, 77:361–401, 2016.

[3] Kathleen L Mosier and Linda J Skitka. Human decision makers and automated decision aids: Made for each other? In *Automation and human performance*, pages 201–220. CRC Press, 2018.

[4] Howard Raiffa. Decision analysis: introductory lectures on choices under uncertainty. 1968.

[5] P. C. Fishburn. *Utility Theory for Decision Making*. Wiley, 1970.

[6] Ralph L Keeney, Howard Raiffa, and Richard F Meyer. *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge university press, 1993.

[7] Bernard Roy. *Multicriteria methodology for decision aiding*, volume 12. Springer Science & Business Media, 1996.

[8] Peter P Wakker. *Prospect theory: For risk and ambiguity*. Cambridge university press, 2010.

[9] Christophe Labreuche. A general framework for explaining the results of a multi-attribute preference model. *Artificial Intelligence*, 175(7-8):1410–1448, 2011.

[10] Darius Braziunas and Craig Boutilier. Elicitation of factored utilities. *AI Magazine*, 29(4):79–79, 2008.

[11] D Krantz, R D Luce, P Suppes, and A Tversky. *Foundations of Measurement (Additive and Polynomial Representations)*, volume 1. Academic Press, 1971.

[12] Michel Grabisch and Christophe Labreuche. A decade of application of the choquet and sugeno integrals in multi-criteria decision aid. *Annals of Operations Research*, 175:247–286, 2010.

[13] F Bacchus and A Grove. Graphical models for preference and utility. In *UAI'95*, 1995.

[14] C Gonzales and P Perny. GAI networks for utility elicitation. In *KR'04*, pages 224–234, 2004.

[15] D Braziunas and C Boutilier. Local utility elicitation in GAI models. In *UAI'05*, 2005.

[16] Darius Braziunas and Craig Boutilier. Minimax regret based elicitation of generalized additive utilities. In *UAI'07*, pages 25–32, 2007.

[17] Ronen I Brafman and Yagil Engel. Directional decomposition of multiattribute utility functions. In *Algorithmic Decision Theory: First International Conference, ADT 2009*, pages 192–202. Springer, 2009.

[18] Damien Bigot, Hélene Fargier, Jérôme Mengin, and Bruno Zanuttini. Using and learning gai-decompositions for representing ordinal rankings. In *ECAI'2012 workshop on Preference Learning (PL 2012)*, pages 5–10. Fürnkranz Johannes Hüllermeier Eyke, 2012.

[19] Michel Grabisch, Christophe Labreuche, and Mustapha Ridaoui. Well-formed decompositions of generalized additive independence models. *Annals of Operations Research*, 312(2):827–852, 2022.

[20] G Debreu. Continuity properties of paretian utility. *Int. Econ. Review*, 5:285–293, 1964.

[21] Christophe Gonzales and Patrice Perny. Gai networks for decision making under certainty. In *IJCAI'05–Workshop on Advances in Preference Handling*, pages 100–105, 2005.

[22] Thomas J Rothenberg. Identification in parametric models. *Econometrica: Journal of the Econometric Society*, pages 577–591, 1971.

[23] Ilya M Sobol'. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and computers in simulation*, 55(1-3):271–280, 2001.

[24] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14:199–222, 2004.

[25] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[26] Carmel Domshlak and Thorsten Joachims. Unstructuring user preferences: Efficient non-parametric utility revelation. *arXiv preprint arXiv:1207.1390*, 2012.

[27] Sébastien Lahaie. Kernel methods for revealed preference analysis. In *ECAI*, pages 439–444, 2010.

[28] Olivier Chapelle and Zaid Harchaoui. A machine learning approach to conjoint analysis. *Advances in neural information processing systems*, 17, 2004.

[29] Francis R Bach, Gert RG Lanckriet, and Michael I Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, page 6, 2004.

[30] Mark Stitson, Alex Gammerman, Vladimir Vapnik, Volodya Vovk, Chris Watkins, and Jason Weston. Support vector regression with anova decomposition kernels. *Advances in kernel methods—Support vector learning*, pages 285–292, 1999.

[31] Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. 1998.

[32] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.

[33] Manik Varma and Debajyoti Ray. Learning the discriminative power-invariance trade-off. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[34] Steve R. Gunn and Jaz S. Kandola. Structural modelling with sparse kernels. *Machine learning*, 48:137–163, 2002.

[35] Nicolas Durrande, David Ginsbourger, Olivier Roustant, and Laurent Carraro. Anova kernels and rkhs of zero mean functions for model-based sensitivity analysis. *Journal of Multivariate Analysis*, 115:57–67, 2013.

[36] Rauf Izmailov, Vladimir Vapnik, and Akshay Vashist. Multidimensional splines with infinite number of knots as svm kernels. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2013.

[37] Francis Bach. Exploring large feature spaces with hierarchical multiple kernel learning. *Advances in neural information processing systems*, 21, 2008.

[38] Alain Rakotomamonjy, Francis Bach, Stéphane Canu, and Yves Grandvalet. More efficiency in multiple kernel learning. In *Proceedings of the 24th international conference on Machine learning*, pages 775–782, 2007.