Learning with tree-based tensor formats : Application to uncertainty quantification in vibroacoustics

Erwan Grelier

THESE DE DOCTORAT DE

L'ÉCOLE CENTRALE DE NANTES
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Mathématiques et leurs interactions*

Par

Erwan GRELIER

Learning with tree-based tensor formats

Application to uncertainty quantification in vibroacoustics

Thèse présentée et soutenue à Nantes, le 19/12/2019
Unité de recherche : Laboratoire de Mathématiques Jean Leray

Rapporteurs avant soutenance :

Reinhold Schneider	Professeur, Institut für Mathematik, Technische Universität Berlin
Clémentine Prieur	Professeure des universités, Université Grenoble Alpes

Composition du Jury :

Président :	Rémi Gribonval	Directeur de recherche, Institut national de recherche en sciences et technologies du numérique, Ecole normale supérieure de Lyon
Examinateurs :	Reinhold Schneider	Professeur, Institut für Mathematik, Technische Universität Berlin
	Clémentine Prieur	Professeure des universités, Université Grenoble Alpes
	Stéphane Gaïffas	Professeur des universités, Université Paris Diderot
Dir. de thèse :	Anthony Nouy	Professeur des universités, Ecole Centrale de Nantes
Co-dir. de thèse :	Mathilde Chevreuil	Maître de conférences, Université de Nantes
Invités		
	Cédric Leblond	Docteur, Naval Group
	Régis Lebrun	Docteur, AIRBUS SRT/VPE

École Centrale de Nantes

Doctoral Thesis

Learning with tree-based tensor formats

—

Application to uncertainty quantification in vibroacoustics

Author:
Erwan Grelier

Supervisors:
Prof. Anthony Nouy
Dr. Mathilde Chevreuil

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

Acknowledgements

This thesis is part of the Eval-π project, in the context of the Joint Laboratory of Marine Technology between Naval Group, Centrale Nantes and Université de Nantes.

Contents

Introduction (version française)

Le développement rapide des modèles informatiques et les performances croissantes des ressources de calcul ont mené à l'apparition d'un nombre croissant d'études numériques. De plus, cela a permis à certaines industries de remplacer des campagnes expérimentales coûteuses par des simulations numériques (avec toutes les questions que cela soulève, voir par exemple [1]). Ces modèles, qui décrivent la réalité, dépendent souvent de nombreux paramètres d'entrée (nous dirons qu'ils sont en grande dimension) et sont coûteux à évaluer (en termes de mémoire et/ou de puissance de calcul).

Habituellement, les paramètres d'entrée de ces modèles ne sont pas déterministes : ces incertitudes peuvent avoir de nombreuses origines, comme par exemple des écarts dus aux processus de fabrication ou une variabilité due au manque de connaissances de la physique du problème considéré. On peut donc souhaiter quantifier ces incertitudes sur les paramètres d'entrée, ou les propager dans le modèle ; en d'autres termes, faire de la quantification d'incertitudes [2].

La quantification d'incertitudes nécessite classiquement l'évaluation du modèle pour de nombreuses réalisations des paramètres d'entrée aléatoires afin d'obtenir des statistiques significatives, comme des probabilités d'événements rares, des quantiles, *etc.* Cela implique l'utilisation de méthodes robustes étant capables d'apprendre avec précision un modèle de substitution—une approximation—pouvant être évaluée de manière efficace.

En particulier, l'industrie maritime repose grandement sur l'utilisation de modèles informatiques complexes pour décrire le comportement de structures (voir par exemple la *review* [3]), et a donc un intérêt à développer des méthodes de réduction de modèle afin de faire de la quantification d'incertitudes (voir par exemple [4, 5, 6]). Naval Group, un grand groupe industriel français spécialisé dans le naval de défense, s'est associé à Centrale Nantes et l'université de Nantes pour former le *Joint Laboratory of Marine Technology* (JLMT), qui a pour objectif « d'accélérer les développements technologiques du Groupe dans trois domaines clés : la fabrication additive, l'hydrodynamique navale et la simulation numérique multi-physique. »[1] Cette thèse fait partie du projet Eval-π, dans le contexte du JLMT, et a pour objectif d'apporter des solutions dans le dernier domaine clé.

Dans cette thèse, nous considérons le problème de l'approximation d'une fonction en utilisant un échantillon d'apprentissage constitué de données. Cela inclut l'apprentissage supervisé qui a pour but d'approximer la relation entre une variable aléatoire de sortie Y et des variables aléatoires d'entrée $X = (X_1, \ldots, X_d)$, en utilisant des échantillons $\{(x_k, y_k)\}_{k=1}^n$ de

[1] Récupéré sur `www.naval-group.com/`. Accédé le 5 septembre 2019.

(X, Y) (voir par exemple [7]), et l'apprentissage non supervisé, où l'on cherche à approximer la distribution de probabilité d'un vecteur aléatoire d'entrée $X = (X_1, \ldots, X_d)$ en utilisant un échantillon $\{x_k\}_{k=1}^n$ de X. Nous supposons que les échantillons sont donnés, indépendants et identiquement distribués, qui est un cadre classique en apprentissage statistique [8, 9].

L'approximation recherchée est typiquement obtenue par minimisation d'une fonctionnelle de risque

$$\mathcal{R}(g) = \mathbb{E}(\gamma(g, Z)),$$

avec $\mathbb{E}(\cdot)$ l'espérance mathématique, $Z = (X, Y)$ pour l'apprentissage supervisé et $Z = X$ pour l'apprentissage non supervisé, et γ une fonction de contraste qui est telle que $\gamma(g, z)$ mesure l'erreur due à l'utilisation de l'approximation g pour un échantillon z de Z. La fonction f minimisant le risque est appelée fonction cible (ou oracle).

En pratique, étant donné un échantillon d'apprentissage $\{z_k\}_{k=1}^n$ de Z, une approximation g_M^n de la fonction cible est obtenue par minimisation du risque empirique

$$\mathcal{R}_n(g) = \frac{1}{n} \sum_{k=1}^{n} \gamma(g, z_k)$$

sur un ensemble de fonctions M, appelé classe (ou ensemble d'hypothèse). Lorsque la dimension d de X est grande ou que la taille de l'échantillon d'apprentissage n est petite, les classes de fonctions dans lesquelles l'approximation est recherchée doivent exploiter des structures de faible dimension de la fonction cible f à approximer. Des classes de fonctions typiques pour aborder les problèmes d'approximation en grande dimension incluent :

- des expansions $\sum_{\lambda \in \Lambda} c_\lambda \psi_\lambda(x)$ sur un ensemble de fonctions $\{\psi_\lambda\}_{\lambda \in \Lambda}$, éventuellement choisies dans un dictionnaire de fonctions (dans lequel une approximation parcimonieuse est recherchée, voir la *review* [10], et [11] pour les polynômes et [12, 13] pour les ondelettes) ;
- des modèles additifs $g_1(x_1) + \cdots + g_d(x_d)$, ou plus généralement des modèles avec des interactions de faible ordre $\sum_{\alpha \in T} g_\alpha(x_\alpha)$, avec T une collection de petits sous-ensembles α de $\{1, \ldots, d\}$ et g_α des fonctions des groupes de variables x_α ;
- des modèles multiplicatifs $g_1(x_1) \cdots g_d(x_d)$ ou des sommes de modèles multiplicatifs $\sum_{i=1}^r g_1^i(x_1) \cdots g_d^i(x_d)$;
- des modèles multiplicatifs généralisés[2] $\prod_{\alpha \in T} g_\alpha(x_\alpha)$, avec T une collection de sous-ensembles α de $\{1, \ldots, d\}$ et g_α des fonctions des groupes de variables x_α ;
- des modèles *ridge* $v(Ax)$ [15], où A est une application linéaire de $\mathbb{R}^d$ vers $\mathbb{R}^m$, et où v appartient à une classe de fonctions de m variables ;
- des modèles *projection pursuit*[3] $g_1(w_1^\intercal x) + \cdots + g_m(w_m^\intercal x)$, où $w_i \in \mathbb{R}^d$, $i = 1, \ldots, m$,

[2] Ceux-ci incluent les réseaux bayésiens ou plus généralement les modèles graphiques pour l'apprentissage non supervisé, voir [14].

[3] Ceux-ci correspondent à un modèle *ridge* avec une fonction g choisie dans la classe des modèles additifs. Pour le cas particulier de la régression, voir [16].

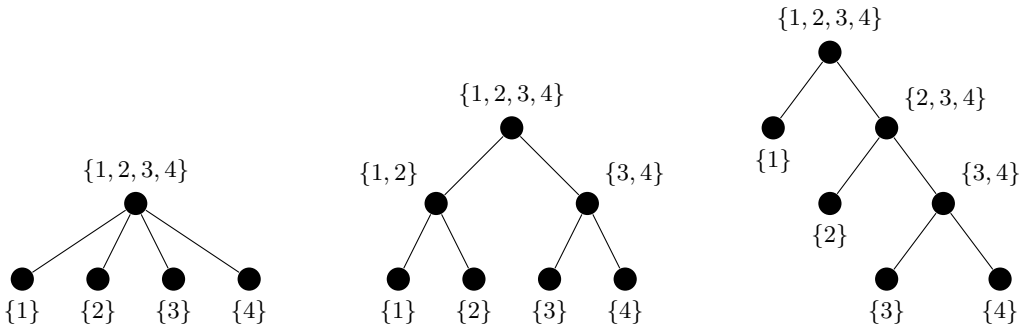
un cas particulier étant les réseaux de neurones à une couche cachée $c_1\sigma(w_1^\intercal x + b_1) + \cdots + c_m\sigma(w_m^\intercal x + b_m)$, avec σ une fonction d'activation ;

- des compositions de fonctions plus générales $v \circ h_1 \circ \cdots \circ h_L(x)$, où les fonctions h_i, $i = 1, \ldots, L$, sont à valeurs vectorielles et dont les composantes sont choisies dans des classes de fonctions classiques. Les réseaux de neurones profonds en sont un cas particulier, pour lesquels les h_i sont des fonctions *ridge* de la forme $h_i(t) = \sigma(A_i t + b_i)$, $i = 1, \ldots, L$ (avec les matrices A_i éventuellement parcimonieuses, comme pour les réseaux à convolution ou récurrents).

Dans cette thèse, nous considérons des classes de fonctions structurées par rangs, largement utilisées en traitement des données, du signal et des images, ainsi qu'en analyse numérique [17, 18]. Nous nous focalisons sur des fonctions qui peuvent s'écrire

$$g(x) = \sum_{i_\alpha=1}^{r_\alpha} g_{i_\alpha}^\alpha(x_\alpha) g_{i_{\alpha^c}}^\alpha(x_{\alpha^c}) \quad \forall \alpha \in T,$$

avec T une collection de sous-ensembles de $D = \{1, \ldots, d\}$, et $\alpha^c = D \setminus \alpha$ le complémentaire de $\alpha \in D$. Nous appelons α-rang le plus petit entier r_α tel que la fonction g admet la représentation ci-dessus. En particulier, nous considérons des collections T qui sont des arbres de dimension, quelques exemples étant montrés sur la Figure 1. Nous appelons le tuple $r = (r_\alpha)_{\alpha \in T}$ le rang d'arbre.



(a) Un arbre trivial. (b) Un arbre binaire équilibré. (c) Un arbre binaire linéaire.

Figure 1: Exemples d'arbres de dimension de $D = \{1, \ldots, 4\}$.

La classe de fonctions correspondante est l'ensemble des fonctions au format de tenseurs basés sur des arbres [19], noté $\mathcal{T}_r^T(\mathcal{H})$, où $\mathcal{H} = \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_d$ est un espacé produit de fonctions (*e.g.* $L_\mu^2(\mathbb{R}^d)$ avec μ une mesure de probabilité produit). C'est un cas particulier des réseaux de tenseurs [20]. Elle inclut le format Tucker pour un arbre trivial (Figure 1a), le format de tenseurs hiérarchiques [21] pour un arbre binaire équilibré (Figure 1b), et le format *tensor-train* [22] pour un arbre linéaire (Figure 1c).

Considérer pour T des arbres de dimension donne de bonnes propriétés topologiques et géométriques à l'ensemble $\mathcal{T}_r^T(\mathcal{H})$ [23, 19, 24, 25, 26, 27, 28], et ses éléments admettent des représentations explicites et numériquement stables. De plus, la complexité des fonctions

au format de tenseurs basés sur des arbres est linéaire en la dimension d et polynomiale en les rangs, rendant cette classe de fonctions adaptée à l'apprentissage en grande dimension.

L'ensemble $\mathcal{T}_r^T(\mathcal{H})$ peut être interprété comme une classe de fonctions qui sont des compositions de fonctions multilinéaires, la structure des compositions étant donnée par l'arbre. Cela mène à une interprétation des formats de tenseurs basés sur des arbres comme un cas particulier de réseaux de neurones profonds avec une connectivité parcimonieuse et sans *parameter sharing*. La structure parcimonieuse du réseau est donnée par l'arbre de dimension du tenseur, sa profondeur par la profondeur de l'arbre, et sa largeur à un niveau l par la somme des α-rangs associés aux nœuds α de niveau l dans l'arbre. Les fonctions d'activation dans de tels réseaux ne sont pas celles habituellement utilisées en apprentissage profond (sigmoïde, tangente hyperbolique, *rectifier linear unit*, *etc.*) mais des fonctions multilinéaires. Dans ce contexte, les tenseurs au format canonique, qui sont associés à un arbre trivial (Figure 1a) où le tenseur de la racine est diagonal, correspondent à des réseaux *fully connected* à une couche cachée, de largeur le rang canonique. Des tenseurs basés sur des arbres linéaires (Figure 1c) correspondent à des réseaux récurrents, avec une profondeur égale à la dimension d et une largeur au niveau l égale à la somme des α-rangs associés aux nœuds de niveau l dans l'arbre [29]. Finalement, des tenseurs basés sur des arbres équilibrés (Figure 1b) correspondent à des réseaux à convolution, avec $\log_2(d)$ couches cachées—la profondeur de l'arbre—et une largeur au niveau l égale à la somme des α-rangs associés aux nœuds de niveau l dans l'arbre [30].

L'apprentissage dans des classes de fonctions non-linéaires telles que les réseaux de neurones profonds impliquent de nombreuses difficultés [31, Chapitre 8], parmi elles, un grand nombre de paramètres à apprendre et l'existence de minima locaux du risque à minimiser. L'algorithme du gradient stochastique est largement utilisé pour résoudre le problème de minimisation du risque empirique : cette méthode, en calculant le gradient grâce à un sous-échantillon choisi aléatoirement, donne de bons résultats même avec des réseaux très complexes nécessitant un grand échantillon d'apprentissage.

Un des objectifs principaux de cette thèse est de fournir des algorithmes d'apprentissage robustes avec des classes de fonctions au format de tenseurs basés sur des arbres, en exploitant fortement notre connaissance de ce format.

Nous proposons tout d'abord des algorithmes stables d'apprentissage dans $\mathcal{T}_r^T(\mathcal{H})$ en utilisant la multilinéarité de la paramétrisation pour transformer le problème non-linéaire de minimisation du risque empirique en une série de problèmes d'apprentissage linéaires. Des conditions d'orthogonalité de la représentation (obtenues grâce à la décomposition en valeurs singulières d'ordre supérieur [32, Section 11.4.2] combinée à l'utilisation de bases orthonormés de $\mathcal{H}_\nu$, $\nu = 1, \ldots, d$) donnent de bonnes propriétés aux problèmes à résoudre pour construire une approximation dans $\mathcal{T}_r^T(\mathcal{H})$ dans un contexte de minimisation au sens des moindres carrés. Cela permet l'utilisation de méthodes d'optimisation classiques ainsi que d'obtenir des estimateurs du risque de type *fast cross-validation*, utiles à la fois pour la validation et pour la sélection de modèle. Les algorithmes proposés sont capables d'exploiter la

parcimonie dans les paramètres, ce qui peut être utile pour réduire encore plus la complexité des représentations considérées.

Une propriété intéressante est que toute fonction dans un espace produit de dimension finie $\mathcal{H}$ peut être représentée dans $\mathcal{T}_r^T(\mathcal{H})$ pour n'importe quel arbre T, à condition de choisir des rangs r assez élevés. Cependant, la complexité de la représentation d'une fonction au format de tenseurs basés sur les arbres (son nombre de paramètres) dépendant de manière polynomiale des rangs, il est désirable que ceux-ci soient les plus petits possible. Nous proposons des algorithmes adaptatifs ayant pour but d'obtenir une bonne convergence de l'erreur par rapport aux rangs, en augmentant de manière séquentielle uniquement un sous-ensemble d'α-rangs r_α associés aux plus hautes erreurs de troncation. Dans le contexte de la minimisation au sens des moindres carrés, ces erreurs sont estimées en utilisant la décomposition en valeurs singulières d'ordre supérieur de fonctions au format de tenseurs basés sur des arbres.

Même si n'importe quelle fonction de $\mathcal{H}$ peut être représentée dans $\mathcal{T}_r^T(\mathcal{H})$, choisir un arbre de dimension adapté est en pratique d'une importance capitale. En effet, les rangs nécessaires pour l'approximation d'une fonction à un niveau d'erreur donné peuvent fortement dépendre du choix de l'arbre de dimension. Cela peut avoir, une fois de plus, un fort impact sur la complexité de l'approximation, et peut être un réel problème en grande dimension. Cette problématique est liée au choix d'une structure parcimonieuse particulière pour les réseaux de neurones profonds (par exemples les réseaux à convolution ou récurrents, avec un ordre des variables dépendant de l'application). Trouver l'arbre de dimension optimal (au sens de la complexité à une précision donnée) est un problème combinatoire, insoluble en pratique. Nous proposons un algorithme stochastique qui explore l'ensemble des arbres de dimension à arité donnée (le nombre maximal d'enfants que n'importe quel nœud de l'arbre possède) en appliquant des changements de l'arbre choisis aléatoirement (en suivant une règle heuristique) et retournant celui donnant la plus faible complexité. Pour faire le lien avec les réseaux de neurones profonds, cette procédure permet la modification de la structure du réseau, permettant par exemple la transition d'un réseau récurrent à un réseau à convolution. Cet algorithme d'adaptation d'arbre peut être utilisé seul, par exemple pour réaliser de la compression de tenseurs.

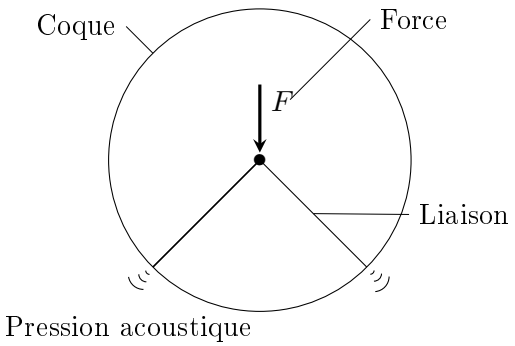
Ces algorithmes d'adaptation des rangs et de l'arbre de dimension sont inclus dans un algorithme adaptatif global d'apprentissage avec des classes de fonctions au format de tenseurs basés sur des arbres. Ce travail a fait l'objet de deux articles pour l'apprentissage avec des classes de fonctions au format de tenseurs basés sur des arbres, pour l'apprentissage supervisé dans [33], et pour l'estimation de densité dans [34].

Certaines fonctions ne peuvent montrer de structure de faible rang qu'après un changement de variables adapté. Dans ce travail, nous proposons une généralisation des formats précédents en considérant des approximations de la forme $g = v \circ h$, avec h une application de $\mathbb{R}^d$ dans $\mathbb{R}^m$ et v fonction à m variables au format de tenseurs basés sur des arbres. Cela correspond à un format de tenseurs basés sur des arbres après un changement de variables obtenu

par l'application à valeurs vectorielles $h(x) = (h_1(x), \ldots, h_m(x))$. Avec $h = \mathrm{id}$ (l'identité de $\mathbb{R}^d$ dans $\mathbb{R}^d$), nous retrouvons le format de tenseurs basés sur des arbres standard. Avec h une application linéaire, cela correspond à une approximation *ridge* $g(x) = v(Ax)$, avec A une matrice et v une fonction au format de tenseurs basés sur des arbres. Avec v un modèle additif et h linéaire, $v \circ h$ correspond à un modèle *projection pursuit* (voir [16]). Un modèle additif étant représenté au format de tenseurs basés sur des arbres avec des α-rangs bornés par 2, le format ici proposé est donc capable de représenter un modèle *projection pursuit* avec une complexité similaire Nous proposons ici des algorithmes adaptatifs pour la construction de telles approximations, avec une dimension m croissante. Ces algorithmes sont inspirés des algorithmes gloutons pour la *projection pursuit regression* [16].

Les algorithmes proposés peuvent être appliqués à l'approximation de fonctions univariées. L'idée est la suivante : considérons, sans perte de généralité, une fonction univariée F définie sur $[0, 1[$. Un élément $x \in [0, 1[$ peut être identifié avec un tuple $(i_1, \ldots, i_d, y)$, tel que $x = t_{b,d}(i_1, \ldots, i_d, y) = \sum_{k=1}^{d} i_k b^{-k} + b^{-d} y$, avec $i_k \in I_b = \{0, \ldots, b-1\}$, $k = 1, \ldots, d$, et $y = b^d x - \lfloor b^d x \rfloor \in [0, 1[$. Cela donne une identification entre une fonction univariée $F(t_{b,d}(i_1, \ldots, i_d, y))$ et une fonction multivariée $f(i_1, \ldots, i_d, y)$ définie sur $\{0, \ldots, b-1\}^d \times [0, 1[$. Les algorithmes proposés peuvent ainsi être appliqués à l'apprentissage de f au format de tenseurs basés sur des arbres. Cette identification est appelée tensorisation (ou quantisation quand $b = 2$ [35]).

Dans le contexte du projet Eval-π et du *Joint Laboratory of Marine Technology*, le problème à aborder est celui de la quantification d'incertitude de la fonction de réponse en fréquence de la puissance acoustique rayonnée produite par une structure immergée en vibration. La puissance acoustique rayonnée est une quantité scalaire qui mesure le bruit que produit une structure donnée. Cette quantité est déduite d'une modélisation fluide-structure éléments finis fournie par Naval Group, qui décrit la réponse vibratoire d'un carlingage lié à un tronçon de coque immergé (représenté sur la Figure 2b), auquel une force est appliquée (voir la Figure 2a). Pour plus d'informations sur l'interaction fluide-structure et le couplage éléments finis, voir [6].



(a) Représentation d'une source sonore.

(b) Géométrie du cas d'étude (source Naval Group).

Figure 2: Cas d'étude : carlingage lié à un tronçon de coque immergé. Une force oscillante est appliquée au carlingage et son énergie est transmise à l'eau *via* la coque.

En réalité, de nombreuses incertitudes sont présentes : les propriétés mécaniques des matériaux, les propriétés de l'eau, l'amplitude, la position et l'orientation de la force, *etc.* (voir l'exemple d'étude des incertitudes dans ce domaine [36]). Dans cette étude, nous considérons uniquement comme aléatoires les propriétés mécaniques des matériaux constituant le véhicule. Notre objectif est de proposer une méthodologie pour construire une approximation de la fonction de réponse en fréquence de la puissance acoustique rayonnée sur une large bande de fréquences, comme fonction de paramètres d'entrée incertains, ainsi que de deux quantités issues de celle-ci : sa fonction de réponse en fréquence intégrée sur des bandes de tiers d'octave et sa courbe enveloppe, chacune répondant à des objectifs différents d'un point de vue industriel. La méthodologie proposée fait intervenir une procédure spécifique au problème qui aligne certains pics de résonance pour toutes les valeurs des paramètres d'entrée, et utilise les algorithmes proposés précédemment pour l'apprentissage avec des classes de fonctions au format de tenseurs basés sur des arbres. Nous tirons aussi avantage de la tensorisation des fonctions univariées pour représenter la fonction de réponse en fréquence en bande fine de la puissance acoustique rayonnée au format de tenseurs à structure d'arbre.

Tous les algorithmes proposés dans cette thèse ont été implémentés dans la *toolbox open-source ApproximationToolbox*, qui sera rendue publique prochainement.

Le plan de cette thèse est comme suit.

Le Chapitre 1 présente les classes de fonctions au format de tenseurs basés sur des arbres, leurs liens avec d'autres classes de fonctions, différentes représentations utiles ainsi qu'un algorithme de troncation (s'appuyant sur la décomposition en valeurs singulières d'ordre supérieur) et un algorithme innovant d'adaptation de l'arbre de dimension.

Dans le Chapitre 2, nous développons des algorithmes d'apprentissage avec des classes de fonctions au format de tenseurs basés sur des arbres, avec adaptation de l'arbre de dimension et des rangs, en utilisant un algorithme heuristique basé sur l'estimation des erreurs de troncation pour ce dernier. Ces algorithmes sont décrits dans un cadre d'apprentissage général, et spécifiés pour deux applications typiques : la régression au sens des moindres carrés, un type d'apprentissage supervisé, et l'estimation de densité au sens des moindres carrés, un type d'apprentissage non supervisé. Nous montrons les performances de ces algorithmes avec plusieurs expérimentations numériques.

Le Chapitre 3 est dévoué au développement d'algorithmes d'apprentissage combinant formats de tenseurs basés sur des arbres et changements de variables. Ces algorithmes fournissent une séquence d'approximations avec dimension effective croissante, et font intervenir la résolution de problèmes d'apprentissage linéaires et non-linéaires. Les performances de ces algorithmes sont présentées sur des cas tests montrant qu'ils sont capables de trouver, lorsqu'ils existent, des changements de variables capables de réduire la dimension effective et de découvrir des structures de faible rang dans la fonction à approximer.

Dans le Chapitre 4, nous appliquons les algorithmes proposés au problème proposé par Naval Group : la quantification d'incertitudes de la réponse en fréquence de la puissance acoustique

rayonnée d'une structure immergée. Des quantités d'intérêt spécifiques à ce problème sont déduites de la puissance acoustique rayonnée, et une procédure d'alignement des pics de résonance est développée, pour proposer des méthodologies efficaces qui pourraient être utilisées dans un contexte industriel.

Le Chapitre exploratoire 5 décrit la tensorisation de fonctions univariées, introduite ci-dessus, ainsi que quelques exemples d'approximation d'une fonction univariée au format de tenseurs basés sur des arbres.

L'Annexe A présente différents types de représentation de distributions de probabilités en utilisant les formats de tenseurs basés sur des arbres, fournit quelques résultats sur les relations entre les rangs de ces représentations et donne des exemples de représentations de modèles probabilistes standards au format de tenseurs basés sur des arbres.

Enfin, dans l'Annexe B, nous décrivons comment obtenir des bases de polynômes orthonormés par rapport à une mesure de probabilité donnée, qui est soit connue soit estimée depuis un échantillon.

Introduction

The fast development of computational models and the growing performances of hardware resources led to an increased number of numerical studies. Moreover, it allowed some industries to replace expensive experimental work with computer simulations (with all the questions this raises, see for example [1]). These models, describing reality, often involve many input parameters (we will say that they are high-dimensional) and are costly to evaluate (in terms of memory usage and/or computational power).

Usually, the input parameters of these models are not deterministic: these so-called uncertainties can have many origins, for instance discrepancies due to manufacturing processes or variability induced by a lack of knowledge of the underlying physics of a problem. One may then wish to quantify these input parameters uncertainties, or to propagate them through the model; in other words, to perform uncertainty quantification [2].

Uncertainty quantification typically requires the evaluation of the model for many realizations of the random input parameters in order to obtain meaningful statistics, such as probabilities of rare events, quantiles, *etc.* It then calls for robust methods able to learn an accurate surrogate model—an approximation—that can be efficiently evaluated.

Notably, the maritime industry heavily relies on the use of complex numerical models to describe the behavior of marine structures (see for instance the review [3]), and then has an interest in developing model reduction methods to perform uncertainty quantification (see for example [4, 5, 6]). Naval Group, a major French industrial group specialized in naval defense, partnered with Centrale Nantes and Université de Nantes to form the Joint Laboratory of Marine Technology (JLMT), which aims at "accelerating the group's technological developments in three key areas: additive manufacturing, naval hydrodynamics and multi-physical numerical simulation."[4] This thesis is part of the Eval-π project, in the context of the JLMT, and aims at bringing solutions to the latter key area.

In this thesis, we consider the problem of approximating a function based on a training set of data. This includes supervised learning which aims at approximating the relation between an output random variable Y and input random variables $X = (X_1, \ldots, X_d)$ based on samples $\{(x_k, y_k)\}_{k=1}^n$ of (X, Y) (see for instance [7]), and unsupervised learning where one seeks to approximate the probability distribution of the input random vector $X = (X_1, \ldots, X_d)$ based on samples $\{x_k\}_{k=1}^n$ of X. We assume that the samples are given, independent and identically distributed, which is a classical setting in statistical learning [8, 9].

[4] Retrieved from `www.naval-group.com/`. Accessed on September 5th, 2019.

The sought approximation is typically obtained by minimizing a risk functional

$$\mathcal{R}(g) = \mathbb{E}(\gamma(g, Z)),$$

with $\mathbb{E}(\cdot)$ the mathematical expectation, $Z = (X, Y)$ for supervised learning and $Z = X$ for unsupervised learning, and γ a contrast function such that $\gamma(g, z)$ measures the error due to the use of the approximation g for a sample z of Z. The function f that minimizes the risk is called the target (or oracle) function.

In practice, given a training set of samples $\{z_k\}_{k=1}^n$ of Z, an approximation g_M^n of the target function is obtained by minimizing the empirical risk

$$\mathcal{R}_n(g) = \frac{1}{n}\sum_{k=1}^{n} \gamma(g, z_k)$$

over a set of functions M, called a model class (or hypothesis set). When the dimension d of X is high or when the sample size n is small, the model classes in which the approximation is sought must exploit low-dimensional structures of the target function f to approximate. Typical model classes for tackling high-dimensional approximation problems include:

- expansions $\sum_{\lambda\in\Lambda} c_\lambda \psi_\lambda(x)$ on a set of functions $\{\psi_\lambda\}_{\lambda\in\Lambda}$, possibly chosen from a dictionary of functions (in which a sparse approximation is sought, see the review [10], and [11] for polynomials and [12, 13] for wavelets);
- additive models $g_1(x_1) + \cdots + g_d(x_d)$, or more general models with low-order interactions $\sum_{\alpha\in T} g_\alpha(x_\alpha)$, with T a collection of small subsets α of $\{1, \ldots, d\}$ and g_α functions of the groups of variables x_α;
- multiplicative models $g_1(x_1)\cdots g_d(x_d)$ or a sum of such models $\sum_{i=1}^r g_1^i(x_1)\cdots g_d^i(x_d)$;
- generalized multiplicative models[5] $\prod_{\alpha\in T} g_\alpha(x_\alpha)$, with T a collection of subsets α of $\{1, \ldots, d\}$ and g_α functions of the groups of variables x_α;
- ridge models $v(Ax)$ [15], where A is a linear map from $\mathbb{R}^d$ to $\mathbb{R}^m$, and where v belongs to a model class of functions of m variables;
- projection pursuit models[6] $g_1(w_1^\intercal x) + \cdots + g_m(w_m^\intercal x)$, where $w_i \in \mathbb{R}^d$, $i = 1, \ldots, m$, a particular case being neural networks with one hidden layer $c_1\sigma(w_1^\intercal x + b_1) + \cdots + c_m\sigma(w_m^\intercal x + b_m)$, with σ the activation function;
- more general compositions of functions $v \circ h_1 \circ \cdots \circ h_L(x)$, where the functions h_i, $i = 1, \ldots, L$, are vector-valued functions whose components are taken in some standard model classes. A particular case is deep neural networks for which the h_i are ridge functions of the form $h_i(t) = \sigma(A_i t + b_i)$, $i = 1, \ldots, L$ (with A_i possibly sparse, such as for convolutional or recurrent networks).

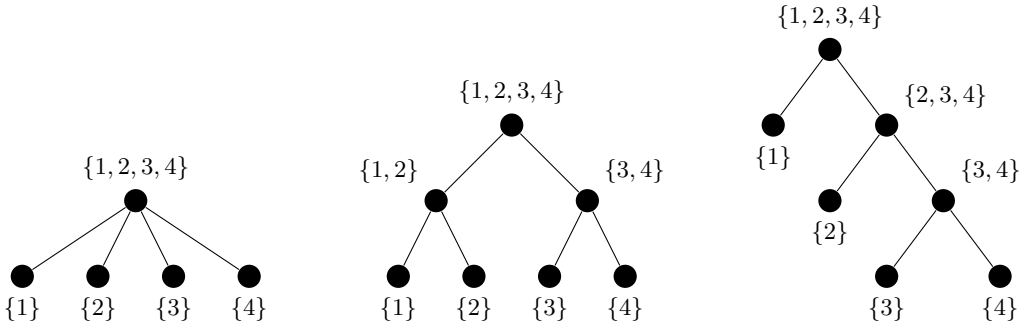
[5] These include bayesian networks or more general graphical models for unsupervised learning, see [14].

[6] These correspond to a ridge model with a function g taken in the class of additive models. For the particular case of regression, see [16].

In this thesis, we consider model classes of rank-structured functions, widely used in data analysis, signal and image processing, and numerical analysis [17, 18]. We focus on functions that can be written

$$g(x) = \sum_{i_\alpha=1}^{r_\alpha} g^\alpha_{i_\alpha}(x_\alpha) g^\alpha_{i_{\alpha^c}}(x_{\alpha^c}) \quad \forall \alpha \in T,$$

with T a collection of subsets of $D = \{1, \ldots, d\}$, and $\alpha^c = D \setminus \alpha$ the complementary subset of $\alpha \in D$. We call α-rank the smallest integer r_α such that the function g admits the above representation. In particular, we consider collections T that are dimension trees, some examples being displayed in Figure 3. We then call the tuple $r = (r_\alpha)_{\alpha \in T}$ the tree-based rank.



(a) A trivial tree. (b) A balanced binary tree. (c) A linear binary tree.

Figure 3: Examples of dimension trees over $D = \{1, \ldots, 4\}$.

.

The corresponding model class is the set of functions in tree-based tensor format [19], denoted by $\mathcal{T}_r^T(\mathcal{H})$, where $\mathcal{H} = \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_d$ is some tensor space of functions (*e.g.* $L^2_\mu(\mathbb{R}^d)$ with μ a product probability measure). It is a particular class of tensor networks [20]. It includes the Tucker format for a trivial tree (Figure 3a), the hierarchical tensor format [21] for a balanced binary tree (Figure 3b), and the tensor-train format [22] for a linear tree (Figure 3c).

Considering dimension trees for T gives nice topological and geometrical properties to the model class $\mathcal{T}_r^T(\mathcal{H})$ [23, 19, 24, 25, 26, 27, 28], and its elements admit explicit and numerically stable representations. Furthermore, the complexity of functions in tree-based tensor format is linear in the dimension d and polynomial in the ranks, making this model class suitable for high-dimensional approximation.

The model class $\mathcal{T}_r^T(\mathcal{H})$ can be interpreted as a class of functions that are compositions of multilinear functions, the structure of compositions being given by the tree. This yields an interpretation of tree-based tensor formats as a particular class of deep neural networks with a sparse connectivity and without parameter sharing. The sparse structure of the network is given by the dimension tree of the tensor, its depth by the depth of the tree, and its width at level l by the sum of the α-ranks associated with the nodes α of the tree at level l. The activation functions involved in such networks are not the ones usually used in deep learning (logistic sigmoid, hyperbolic tangent, rectifier linear unit, *etc.*) but multilinear functions.

In this context, canonical tensors, which are associated with a trivial tree (Figure 3a) where the root tensor is diagonal, correspond to fully connected networks with one hidden layer (shallow networks) of width the canonical rank. Tree-based tensors with linear trees (Figure 3c) correspond to recurrent networks, with a depth equal to the dimension d and a width at level l equal to the sum of the α-ranks associated with the nodes at level l of the tree [29]. Finally, tree-based tensors with balanced trees (Figure 3b) correspond to convolutional networks, with $\log_2(d)$ hidden layers—the depth of the tree—and a width at level l equal to the sum of the α-ranks associated with the nodes at level l of the tree [30].

Learning with nonlinear model classes such as deep neural networks involves many difficulties [31, Chapter 8], among them, a high number of parameters to learn and the existence of local minima of the risk function to minimize. Stochastic gradient descent is widely used to solve the empirical risk minimization problem: this method, by computing the gradient using a randomly selected subset of the training sample, has proved to be effective even with highly complex networks requiring a large training set.

One of the main objectives of this thesis is to provide robust algorithms for learning with model classes of functions in tree-based tensor format, by extensively exploiting our knowledge of this format.

We first propose stable learning algorithms in $\mathcal{T}_r^T(\mathcal{H})$ by using the multilinearity of the parametrization to recast the nonlinear empirical risk minimization problem into a series of learning problems with linear model classes. Orthogonality conditions of the representation (obtained using higher-order singular value decompositions [32, Section 11.4.2] combined with the use of orthonormal bases of $\mathcal{H}_\nu$, $\nu = 1, \ldots, d$) give good properties to the problems to be solved to learn an approximation in $\mathcal{T}_r^T(\mathcal{H})$ in a least-squares setting. This enables us to use classical optimization methods and to derive fast cross-validation estimators of the risk, useful both for validation and for model selection. The proposed algorithms are able to exploit sparsity in the parameters, which can be useful to further reduce the complexity of the considered representations.

An attractive property is that any function in a finite dimensional tensor space $\mathcal{H}$ can be represented in $\mathcal{T}_r^T(\mathcal{H})$ for any tree T, provided that the ranks r are chosen high enough. However, the complexity of the representation of a function in tree-based tensor format (its number of parameters) depending polynomially on the ranks, it is desirable that they remain as small as possible. We propose an adaptive algorithm that aims at obtaining a good convergence of the error with respect to the ranks, by sequentially increasing only a subset of α-ranks r_α associated with the highest truncation errors. In the least-squares case, these errors are estimated using the higher-order singular value decompositions of functions in tree-based tensor format.

Although any function in $\mathcal{H}$ can be represented in the format $\mathcal{T}_r^T(\mathcal{H})$, choosing a suitable dimension tree is in practice of utmost importance. Indeed, the ranks required for approximating a function with a specified error may strongly depend on the choice of the dimension

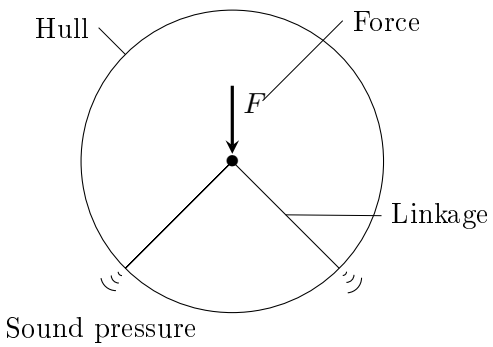
tree. This may have, once again, a significant impact on the complexity of the approximation, and can be a real issue for high-dimensional learning problems. This matter has to be related to the choice of a particular sparse architecture for deep neural networks (*e.g.* convolutional or recurrent networks, with an ordering of the variables which is application-dependent). Finding the optimal dimension tree (in the sense of the complexity at a given accuracy) is a combinatorial problem, intractable in practice. We propose a stochastic algorithm that explores the set of all dimension trees of a given arity (the maximal number of children any node of a tree has) by applying randomly drawn changes in the tree (following a heuristic rule) and returning the one yielding the smallest complexity. Remembering the links between tree-based tensors and deep neural networks, this procedure enables the modification of the structure of the network, enabling for example the transition from a recurrent to a convolutional network. This tree adaptation algorithm is a standalone that can also be used, for instance, to perform compression of tensors.

These tree-based rank and dimension tree adaptation algorithms are embedded in a global adaptive learning algorithm with model classes of functions in tree-based tensor format. This work is the subject of two articles for the learning with model classes of functions in tree-based tensor format, for supervised learning in [33], and for density estimation in [34].

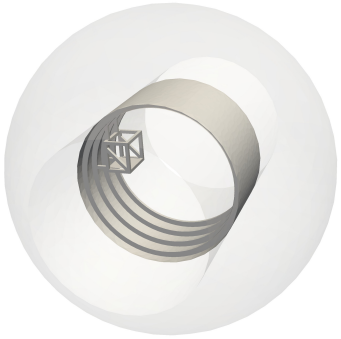
Some functions might only exhibit a low-rank structure after a suitable change of variables. In this work, we propose a generalization of the previous format by considering approximations of the form $g = v \circ h$, with h a mapping from $\mathbb{R}^d$ to $\mathbb{R}^m$ and v a m-dimensional function in tree-based tensor format. This corresponds to a tree-based tensor format after a change of variables induced by the vector-valued map $h(x) = (h_1(x), \ldots, h_m(x))$. With $h = \mathrm{id}$ (the identity from $\mathbb{R}^d$ to $\mathbb{R}^d$), we retrieve the standard tree-based tensor format. With h a linear map, this corresponds to a ridge approximation $g(x) = v(Ax)$, with A a matrix and v a function in tree-based tensor format. With v an additive model and h linear, $v \circ h$ corresponds to a projection pursuit model (see [16]). An additive model being represented in tree-based tensor format with α-ranks bounded by 2, the proposed format is then able to represent a projection pursuit model with a similar complexity. Here, we propose adaptive algorithms for the construction of such approximations, with increasing dimension m. These algorithms are inspired by greedy algorithms for projection pursuit regression [16].

The proposed algorithms can be applied to the approximation of univariate functions. The idea is as follows: let us consider, without loss of generality, a univariate function F that takes values in the interval $[0, 1[$. An element $x \in [0, 1[$ can be identified with a tuple $(i_1, \ldots, i_d, y)$, such that $x = t_{b,d}(i_1, \ldots, i_d, y) = \sum_{k=1}^{d} i_k b^{-k} + b^{-d} y$, with $i_k \in I_b = \{0, \ldots, b-1\}$, $k = 1, \ldots, d$, and $y = b^d x - \lfloor b^d x \rfloor \in [0, 1[$. This gives an identification between the univariate function $F(t_{b,d}(i_1, \ldots, i_d, y))$ and the multivariate function $f(i_1, \ldots, i_d, y)$ defined on $\{0, \ldots, b-1\}^d \times [0, 1[$. The proposed algorithms can then be applied to the learning of f in tree-based tensor format. This identification is called tensorization (or quantization when $b = 2$ [35]).

In the context of the Eval-π project and the Joint Laboratory of Marine Technology, the problem to tackle is the uncertainty quantification of the frequency response function of the sound power level produced by a vibrating underwater structure. The sound power level is a scalar quantity measuring, broadly speaking, how much sound a given structure produces. This quantity is derived from a fluid-structure finite element model provided by Naval Group, which describes the vibration response of a keelson attached to a section of a hull, submerged in water (represented in Figure 4b), and to which a force is applied (see Figure 4a). For more information on fluid-structure interaction and finite element coupling, see [6].



(a) Representation of a noise source.



(b) Geometry of the study case (source Naval Group).

Figure 4: Study case: a keelson attached to a section of a hull submerged in water. An oscillating force is applied to the keelson and its energy is transmitted to the water through the hull.

In reality, many uncertainties are present: mechanical properties of the materials, properties of the water, magnitude, position and orientation of the force, *etc.* (see an example of uncertainties study in this domain [36]). In this study, we only consider as random the mechanical properties of the materials constituting the structure. We aim to provide a methodology to compute an approximation of the frequency response function of the sound power level on a large frequency domain as a function of the uncertain input parameters, as well as of two quantities derived from it: its one-third octave band frequency response function and its envelope curve, each serving different purposes from an industrial point of view. The proposed methodology involves a problem-specific procedure that aligns the signed resonance peaks for all values of the input parameters, and uses the proposed algorithms introduced above for learning with model classes of functions in tree-based tensor format. We also take advantage of the tensorization of univariate functions to represent the narrow-band frequency response function of the sound power level in tree-based tensor format.

All the algorithms proposed in this thesis were implemented in the in-house open-source toolbox ApproximationToolbox, that will be made publicly available soon.

The outline of this thesis is as follows.

Chapter 1 presents the model classes of functions in tree-based tensor format, their links with other model classes, different useful representations as well as an algorithm to perform truncation (based on higher-order singular value decompositions) and a novel algorithm to perform dimension tree adaptation.

In Chapter 2, we develop learning algorithms with model classes of functions in tree-based tensor format, with the adaptation of both the dimension tree and the tree-based rank, using a heuristic algorithm based on an estimation of the truncation errors for the latter. The algorithms are described in a general learning framework, and specified for two typical settings: least-squares regression for supervised learning, and least-squares density estimation, which is a type of unsupervised learning. We demonstrate the performances of the proposed algorithms with several numerical experiments.

Chapter 3 is devoted to the development of learning algorithms that combine tree-based tensor formats and changes of variables. These algorithms return a sequence of approximations with increasing effective dimension and involve solving both linear and nonlinear optimization problems. The performances of these algorithms are presented on test cases showing that they can find, when they exist, changes of variables able to reduce the effective dimension and discover low-rank structures in the function to approximate.

In Chapter 4, we apply the proposed algorithms to the problem proposed by Naval Group: the uncertainty quantification of the frequency response function of the sound power level of an underwater structure. Problem-specific quantities of interest are derived from the sound power level, and a signed resonance peaks alignment procedure is developed, to propose efficient methodologies that might be used in an industrial framework.

The exploratory Chapter 5 describes the tensorization of univariate functions, introduced above, as well as some examples of approximations of a univariate function in tree-based tensor format.

Appendix A presents different types of representation of probability distributions using tree-based formats, provides some results on the relations between the ranks of these representations and gives some examples of representation in tree-based tensor format of standard probabilistic models.

Lastly, in Appendix B, we describe how to obtain polynomial bases orthonormal with respect to some given probability measure, which is either known or estimated from data.

Chapter 1

Tree-based tensor formats

Contents

1.1 Introduction

This chapter is devoted to the presentation of model classes of rank-structured functions, with a focus on the tree-based tensor formats. After an introduction to tensor spaces of multivariate functions in Section 1.2, we present in Sections 1.3, 1.4 and 1.5 the tree-based tensor formats and different representations useful for learning, described in Chapter 2.

Then, we remind in Section 1.6 the handy notion of higher order singular value decomposition of tensors, used to perform truncation, and we present in Section 1.7 how to introduce sparsity in the parameters of the tree-based tensors. Finally, we propose in Section 1.8 a novel algorithm to perform dimension tree adaptation, in order to reduce the storage complexity of the representation in tree-based tensor format of a given function.

This chapter is partly based on two articles [33, 34] for the learning with model classes of functions in tree-based tensor format.

1.2 Tensor spaces of multivariate functions

Let $(X_1, \ldots, X_d)$ be a set of independent random variables, where X_ν is with values in $\mathcal{X}_\nu$ and with probability law μ_ν, $1 \leq \nu \leq d$. Typically, $\mathcal{X}_\nu$ is a subset of $\mathbb{R}$ but the case where $\mathcal{X}_\nu \subset \mathbb{R}^{d_\nu}$, $d_\nu > 1$, can be considered as well. We denote by $X = (X_1, \ldots, X_d)$ the random variable with values in $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$ and with probability law $\mu = \mu_1 \otimes \cdots \otimes \mu_d$, and by $\mathbb{E}(\cdot)$ the mathematical expectation.

Let $\mathcal{H}_\nu$ be a Hilbert space of functions defined on $\mathcal{X}_\nu$, $1 \leq \nu \leq d$, equipped with the inner product $(\cdot, \cdot)_\nu$ and associated norm $\|\cdot\|_\nu$. The elementary tensor product $f^1 \otimes \cdots \otimes f^d$ of functions $f^\nu \in \mathcal{H}_\nu$, $1 \leq \nu \leq d$, is identified with a function defined on $\mathcal{X}$ such that for $x = (x_1, \ldots, x_d) \in \mathcal{X}$, $(f^1 \otimes \cdots \otimes f^d)(x_1, \ldots, x_d) = f^1(x_1) \cdots f^d(x_d)$. The algebraic tensor space $\mathcal{H} = \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_d$ is then identified with the set of functions f which can be written as a finite linear combination of elementary tensors. The tensor space $\mathcal{H}$ is equipped with the canonical inner product $(\cdot, \cdot)$, first defined for elementary tensors by $(f^1 \otimes \cdots \otimes f^d, g^1 \otimes \cdots \otimes g^d) = (f^1, g^1)_1 \cdots (f^d, g^d)_d$, and then extended by linearity to the whole space $\mathcal{H}$. We denote by $\|\cdot\|$ the canonical norm associated with $(\cdot, \cdot)$.

Remark 1.2.1. *If the $\mathcal{H}_\nu$ are infinite dimensional spaces, the tensor space $\mathcal{H}$ is a pre-Hilbert space. A tensor Hilbert space $\mathcal{H}_{\|\cdot\|}$ is obtained by the completion of $\mathcal{H}$ (in the topology induced by the norm $\|\cdot\|$). In particular, $L^2_\mu(\mathcal{X})$ can be identified with the completion of the algebraic tensor space $L^2_{\mu_1}(\mathcal{X}_1) \otimes \cdots \otimes L^2_{\mu_d}(\mathcal{X}_d)$.*

Hereafter, we consider that $\mathcal{H}_\nu$ is a finite dimensional subspace of $L^2_{\mu_\nu}(\mathcal{X}_\nu)$, equipped with the norm $\|f^\nu\|^2_\nu = \mathbb{E}(f^\nu(X_\nu)^2)$. Then, $\mathcal{H}$ is a subspace of $L^2_\mu(\mathcal{X})$, equipped with the canonical norm $\|f\|^2 = \mathbb{E}(f(X)^2)$. Let $\{\phi^\nu_i : i \in I^\nu\}$ be an orthonormal basis of $\mathcal{H}_\nu$, and $N_\nu = \dim(\mathcal{H}_\nu) = \#I^\nu$. For a multi-index $i = (i_1, \ldots, i_d) \in I^1 \times \cdots \times I^d := I$, we let $\phi_i = \phi^1_{i_1} \otimes \cdots \otimes \phi^d_{i_d}$. The set of functions $\{\phi_i : i \in I\}$ constitutes an orthonormal basis of $\mathcal{H}$. A function $f \in \mathcal{H}$ can be written $f(x) = \sum_{i \in I} f_i \phi_i(x)$, where the set of coefficients $(f_i)_{i \in I} \in \mathbb{R}^I$ is identified with a tensor $\boldsymbol{f} \in \mathbb{R}^{I^1} \otimes \cdots \otimes \mathbb{R}^{I^d}$, and

$$\|f\|^2 = \sum_{i \in I} f_i^2 = \sum_{i_1 \in I^1} \cdots \sum_{i_d \in I^d} f^2_{i_1, \ldots, i_d}$$

coincides with the canonical norm of $\boldsymbol{f}$, also denoted by $\|\boldsymbol{f}\|$. Introducing

$$\boldsymbol{\Phi}^\nu(x_\nu) = (\phi_i^\nu(x_\nu))_{i\in I^\nu} \in \mathbb{R}^{I^\nu}$$

and

$$\boldsymbol{\Phi}(x) = \boldsymbol{\Phi}^1(x_1) \otimes \cdots \otimes \boldsymbol{\Phi}^d(x_d) \in \mathbb{R}^{I^1} \otimes \cdots \otimes \mathbb{R}^{I^d},$$

we have $f(x) = \langle \boldsymbol{\Phi}(x), \boldsymbol{f}\rangle$, where $\langle\cdot,\cdot\rangle$ is the canonical inner product on $\mathbb{R}^{I^1} \otimes \cdots \otimes \mathbb{R}^{I^d}$. The linear map

$$F\colon \boldsymbol{f} \mapsto \langle \boldsymbol{\Phi}(\cdot), \boldsymbol{f}\rangle$$

defines a linear isometry from $\mathbb{R}^{I^1} \otimes \cdots \otimes \mathbb{R}^{I^d}$ to $\mathcal{H}$, such that $\|F(\boldsymbol{f})\| = \|\boldsymbol{f}\|$.

Remark 1.2.2 (Discrete set $\mathcal{X}$)**.** *If $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$ is a finite or countable set and $\mu = \sum_{x\in\mathcal{X}} \delta_x = \mu_1 \otimes \cdots \otimes \mu_d$ with $\mu_\nu = \sum_{x_\nu\in\mathcal{X}_\nu} \delta_{x_\nu}$, then the space $\mathcal{H}_\nu = L^2_{\mu_\nu}(\mathcal{X}_\nu)$ is identified with $\ell^2(\mathcal{X}_\nu)$. If $\mathcal{X}_\nu = \{x_\nu^{i_\nu} : i_\nu \in I^\nu\}$, the canonical basis $\phi^\nu_{i_\nu}(x_\nu^{j_\nu}) = \mathbf{1}_{i_\nu=j_\nu}$ (which is equal to 1 if $i_\nu = j_\nu$ and 0 otherwise) is orthonormal in $\mathcal{H}_\nu$. A function $f(x) = \sum_{i\in I} f_i\phi_i(x)$ is then isometrically identified with the set of coefficients $f_{i_1,\ldots,i_d} = f(x_1^{i_1}, \ldots, x_d^{i_d})$.*

The canonical rank of a tensor $f \in \mathcal{H}$ is the minimal integer r such that f can be written in the form

$$f(x_1, \ldots, x_d) = \sum_{i=1}^{r} f_i^1(x_1) \cdots f_i^d(x_d),$$

for some $f_i^\nu \in \mathcal{H}_\nu$ and $r \in \mathbb{N}$. The set of tensors in $\mathcal{H}$ with canonical rank bounded by r is denoted by $\mathcal{R}_r(\mathcal{H})$. An approximation in $\mathcal{R}_r(\mathcal{H})$ is called an approximation in canonical tensor format. For an order-two tensor ($d = 2$), the canonical rank coincides with the classical and unique notion of rank. For higher-order tensors ($d \geq 3$), different notions of rank can be introduced.

1.3 Tree-based ranks and corresponding tree-based formats

For a non-empty subset α in $\{1, \ldots, d\} := D$ and its complementary subset $\alpha^c = D \setminus \alpha$, a tensor $f \in \mathcal{H}$ can be identified with an element $\mathcal{M}_\alpha(f)$ of the space of order-two tensors $\mathcal{H}_\alpha \otimes \mathcal{H}_{\alpha^c}$, where $\mathcal{H}_\alpha = \bigotimes_{\nu\in\alpha} \mathcal{H}_\nu$. This is equivalent to identifying $f(x)$ with a bivariate function of the complementary groups of variables $x_\alpha = (x_\nu)_{\nu\in\alpha}$ and $x_{\alpha^c} = (x_\nu)_{\nu\in\alpha^c}$ in x. The operator $\mathcal{M}_\alpha$ is called the α-matricization operator. The α-rank of f, denoted by $\operatorname{rank}_\alpha(f)$, is the dimension of the minimal subspace $U_\alpha^{\min}(f)$, which is the smallest subspace of functions of the variables x_α such that $\mathcal{M}_\alpha(f) \in U_\alpha^{\min}(f) \otimes \mathcal{H}_{\alpha^c}$. By convention, $U_D^{\min}(f) = \operatorname{span}\{f\}$ and $\operatorname{rank}_D(f) = 1$ if $f \neq 0$ and 0 if $f = 0$. If $\operatorname{rank}_\alpha(f) = r_\alpha$, then f admits the representation

$$f(x) = \sum_{i=1}^{r_\alpha} f_i^\alpha(x_\alpha) f_i^{\alpha^c}(x_{\alpha^c}),$$

for some functions $f_i^\alpha \in \mathcal{H}_\alpha$ and $f_i^{\alpha^c} \in \mathcal{H}_{\alpha^c}$, and $U_\alpha^{\min}(f) = \operatorname{span}\{f_i^\alpha\}_{i=1}^{r_\alpha}$.

From the definition of the α-rank, we deduce the following properties.

Proposition 1.3.1. *If $\alpha = \bigcup_{i\in I} \beta_i$ with $\{\beta_i : i \in I\}$ a collection of disjoint subsets of D, then for any function f,*

$$\operatorname{rank}_\alpha(f) \leq \prod_{i\in I} \operatorname{rank}_{\beta_i}(f).$$

Proposition 1.3.2. *For two functions f and g, and for any $\alpha \subset D$,*

- $\operatorname{rank}_\alpha(f+g) \leq \operatorname{rank}_\alpha(f) + \operatorname{rank}_\alpha(g)$,
- $\operatorname{rank}_\alpha(fg) \leq \operatorname{rank}_\alpha(f)\operatorname{rank}_\alpha(g)$.

Example 1.3.3.

- $f(x) = f^1(x_1)\cdots f^d(x_d)$ *can be written* $f(x) = f^\alpha(x_\alpha)f^{\alpha^c}(x_{\alpha^c})$, *with* $f^\alpha(x_\alpha) = \prod_{\nu\in\alpha} f^\nu(x_\nu)$. *Therefore* $\operatorname{rank}_\alpha(f) \leq 1$ *for all* $\alpha \subset D$.
- $f(x) = \sum_{k=1}^r f_k^1(x_1)\cdots f_k^d(x_d)$ *can be written* $\sum_{k=1}^r f_k^\alpha(x_\alpha)f_k^{\alpha^c}(x_{\alpha^c})$ *with* $f_k^\alpha(x_\alpha) = \prod_{\nu\in\alpha} f_k^\nu(x_\nu)$. *Therefore,* $\operatorname{rank}_\alpha(f) \leq r$ *for all* $\alpha \subset D$.
- $f(x) = f^1(x_1) + \cdots + f^d(x_d)$ *can be written* $f(x) = f^\alpha(x_\alpha) + f^{\alpha^c}(x_{\alpha^c})$, *with* $f^\alpha(x_\alpha) = \sum_{\nu\in\alpha} f^\nu(x_\nu)$. *Therefore,* $\operatorname{rank}_\alpha(f) \leq 2$ *for all* $\alpha \subset D$.
- $f(x) = \prod_{\alpha\in A} f^\alpha(x_\alpha)$ *with A a collection of disjoint subsets is such that* $\operatorname{rank}_\alpha(f) = 1$ *for all* $\alpha \in A$, *and* $\operatorname{rank}_\gamma(f) \leq \prod_{\alpha\in A,\alpha\cap\gamma\neq\emptyset} \operatorname{rank}_{\alpha\cap\gamma}(f^\alpha)$ *for all* γ.

The set of tensors f in $\mathcal{H}$ with α-rank bounded by r_α is denoted by

$$\mathcal{T}_{r_\alpha}^{\{\alpha\}}(\mathcal{H}) = \{f \in \mathcal{H} : \operatorname{rank}_\alpha(f) \leq r_\alpha\}.$$

For a collection T of non-empty subsets of D, we define the T-rank of f as the tuple $\operatorname{rank}_T(f) = \{\operatorname{rank}_\alpha(f) : \alpha \in T\}$. Then, we define the set of tensors $\mathcal{T}_r^T(\mathcal{H})$ with T-rank bounded by $r = (r_\alpha)_{\alpha\in T}$ by

$$\mathcal{T}_r^T(\mathcal{H}) = \{f \in \mathcal{H} : \operatorname{rank}_T(f) \leq r\} = \bigcap_{\alpha\in T} \mathcal{T}_r^{\{\alpha\}}(\mathcal{H}).$$

A dimension partition tree T is a tree such that (i) all nodes $\alpha \in T$ are non-empty subsets of D, (ii) D is the root of T, (iii) every node $\alpha \in T$ with $\#\alpha \geq 2$ has at least two children and the set of children of α, denoted by $S(\alpha)$, is a non-trivial partition of α, and (iv) every node α with $\#\alpha = 1$ has no child and is called a leaf (see for example Figure 1.1).

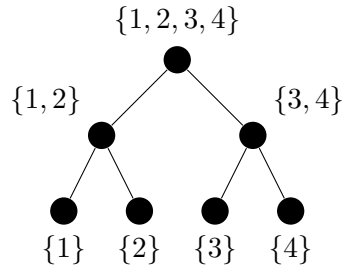


Figure 1.1: Example of dimension partition tree over $D = \{1, \ldots, 4\}$.

.

When T is a dimension partition tree, $\mathcal{T}_r^T(\mathcal{H})$ is the set of tensors with tree-based rank bounded by r, and an approximation in $\mathcal{T}_r^T(\mathcal{H})$ is called an approximation in tree-based (or hierarchical) tensor format [21, 26]. A tree-based rank r is said admissible if the set $\mathcal{T}_{=r}^T(\mathcal{H}) = \{f : \text{rank}_\alpha(f) = r_\alpha, \alpha \in T\}$ of tensors with T-rank r is non empty. Necessary conditions of admissibility can be found in [26, Section 2.3]. In particular, r_D has to be less than or equal to 1 for $\mathcal{T}_{=r}^T$ to be non empty, and $\mathcal{T}_{=r}^T$ is reduced to $\{0\}$ if $r_D = 0$.

For a dimension partition tree T and a node $\alpha \in T$, we denote by $P(\alpha)$ and $A(\alpha)$ the parent and ascendants of α, respectively. The level of a node α is denoted by $\text{level}(\alpha)$. The levels are defined such that $\text{level}(D) = 0$ and $\text{level}(\beta) = \text{level}(\alpha) + 1$ for $\beta \in S(\alpha)$. We let $\text{depth}(T) = \max_{\alpha \in T} \text{level}(\alpha)$ be the depth of T, and $\mathcal{L}(T)$ be the set of leaves of T, which are such that $S(\alpha) = \emptyset$ for all $\alpha \in \mathcal{L}(T)$.

1.3.1 Representation of tensors in tree-based format

Let $f \in \mathcal{T}_r^T(\mathcal{H})$ having a tree-based rank $r = (r_\alpha)_{\alpha \in T}$. By definition of the minimal subspaces, we have that $f \in \bigotimes_{\alpha \in S(D)} U_\alpha^{\min}(f)$, and $U_\alpha^{\min}(f) \subset \bigotimes_{\beta \in S(\alpha)} U_\beta^{\min}(f)$ for any $\alpha \in T \setminus \mathcal{L}(T)$ (see [23]). For any $\alpha \in T$, let $\{f_{k_\alpha}^\alpha\}_{k_\alpha=1}^{r_\alpha}$ be a basis of the minimal subspace $U_\alpha^{\min}(f)$. For each $\alpha \in T \setminus \mathcal{L}(T)$, we let $I^\alpha = \bigtimes_{\beta \in S(\alpha)} \{1, \dots, r_\beta\}$ and

$$\phi_{i_\alpha}^\alpha(x_\alpha) = \prod_{\beta \in S(\alpha)} f_{k_\beta}^\beta(x_\beta), \quad i_\alpha = (k_\beta)_{\beta \in S(\alpha)} \in I^\alpha, \tag{1.1}$$

be a basis of $\bigotimes_{\beta \in S(\alpha)} U_\beta^{\min}(f) \subset \mathcal{H}_\alpha$. Therefore, f admits the following representation:

$$f(x) = \sum_{i_D \in I^D} C_{i_D,1}^D \phi_{i_D}^D(x)$$

where $C^D \in \mathbb{R}^{I^D \times \{1\}} = \mathbb{R}^{I^D}$ is a tensor of order $\#S(D)$, and for any $\alpha \in T \setminus \{D\}$, the functions $\{f_{k_\alpha}^\alpha\}_{k_\alpha=1}^{r_\alpha}$ admit the following representation

$$f_{k_\alpha}^\alpha(x_\alpha) = \sum_{i_\alpha \in I^\alpha} C_{i_\alpha,k_\alpha}^\alpha \phi_{i_\alpha}^\alpha(x_\alpha), \quad 1 \le k_\alpha \le r_\alpha,$$

where $C^\alpha \in \mathbb{R}^{K^\alpha}$, $K^\alpha := I^\alpha \times \{1, \dots, r_\alpha\}$. The function f can finally be written $f = F(\boldsymbol{f})$, with $F : \mathbb{R}^I \to \mathcal{H}$ the linear isometry introduced in Section 1.2, and $\boldsymbol{f} \in \mathcal{T}_r^T(\mathbb{R}^I)$ the tensor whose components are given by

$$f_{i_1,\dots,i_d} = \sum_{\substack{1 \le k_\beta \le r_\beta \\ \beta \in T}} \prod_{\alpha \in T \setminus \mathcal{L}(T)} C_{(k_\beta)_{\beta \in S(\alpha)},k_\alpha}^\alpha \prod_{\alpha \in \mathcal{L}(T)} C_{i_\alpha,k_\alpha}^\alpha. \tag{1.2}$$

Remark 1.3.4. *Note that a function $f \in \mathcal{H}$ in a tensor format is associated with a tensor $\boldsymbol{f} = F^{-1}(f)$ in $\mathbb{R}^I$ in the same tensor format. In particular, f is in $\mathcal{T}_r^T(\mathcal{H})$ if and only if $\boldsymbol{f} = F^{-1}(f)$ is in $\mathcal{T}_r^T(\mathbb{R}^I)$.*

1.3.2 Interpretation as compositions of multilinear functions

For a node $\alpha \in T \setminus \mathcal{L}(T)$, a tensor $C^\alpha \in \mathbb{R}^{I_\alpha \times \{1,\dots,r_\alpha\}} = \mathbb{R}^{(\times_{\beta\in S(\alpha)}\{1,\dots,r_\beta\})\times\{1,\dots,r_\alpha\}}$ can be identified with a $\mathbb{R}^{r_\alpha}$-valued multilinear function $h^\alpha : \bigtimes_{\beta\in S(\alpha)} \mathbb{R}^{r_\beta} \to \mathbb{R}^{r_\alpha}$. Also, for a leaf node $\alpha \in \mathcal{L}(T)$, $C^\alpha \in \mathbb{R}^{I_\alpha\times\{1,\dots,r_\alpha\}}$ can be identified with a linear function $h^\alpha : \mathbb{R}^{\#I^\alpha} \to \mathbb{R}^{r_\alpha}$. Denoting by f^α the $\mathbb{R}^{r_\alpha}$-valued function defined for $x_\alpha \in \mathcal{X}_\alpha$ by $f^\alpha(x_\alpha) = (f_1^\alpha(x_\alpha), \dots, f_{r_\alpha}^\alpha(x_\alpha))$, we have

$$\begin{aligned} f^\alpha(x_\alpha) &= h^\alpha(\mathbf{\Phi}^\alpha(x_\alpha)) && \text{for } \alpha \in \mathcal{L}(T), \\ f^\alpha(x_\alpha) &= h^\alpha((f^\beta(x_\beta))_{\beta\in S(\alpha)}) && \text{for } \alpha \in T \setminus \mathcal{L}(T), \end{aligned}$$

where

$$\mathbf{\Phi}^\alpha(x_\alpha) = (\phi_{i_\alpha}^\alpha(x_\alpha))_{i_\alpha \in I^\alpha} \in \mathbb{R}^{I^\alpha}$$

with $\phi_{i_\alpha}^\alpha$ defined by (1.1) for $\alpha \in T \setminus \mathcal{L}(T)$, and finally

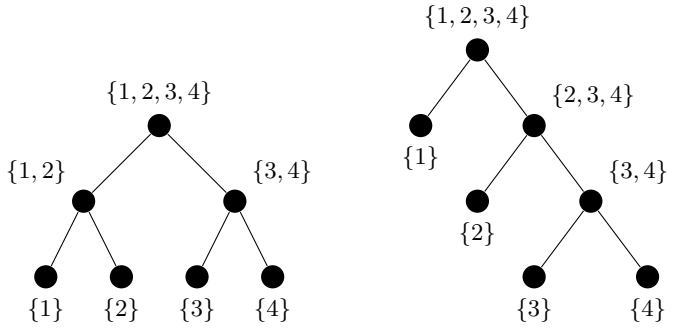
$$f(x) = f^D(x) = h^D((f^\alpha(x_\alpha))_{\alpha\in S(D)}).$$

For example, in the case of Figure 1.2a, the tensor f admits the representation

$$f(x) = h^D(h^{\{1,2\}}(h^{\{1\}}(\mathbf{\Phi}^1(x_1)), h^{\{2\}}(\mathbf{\Phi}^2(x_2))), h^{\{3,4\}}(h^{\{3\}}(\mathbf{\Phi}^3(x_3)), h^{\{4\}}(\mathbf{\Phi}^4(x_4)))),$$

and in the case of Figure 1.2b, it admits the representation

$$f(x) = h^D(h^{\{1\}}(\mathbf{\Phi}^1(x_1)), h^{\{2,3,4\}}(h^{\{2\}}(\mathbf{\Phi}^2(x_2)), h^{\{3,4\}}(h^{\{3\}}(\mathbf{\Phi}^3(x_3)), h^{\{4\}}(\mathbf{\Phi}^4(x_4))))).$$



(a) A balanced binary tree. (b) A linear binary tree.

Figure 1.2: Examples of dimension partition trees over $D = \{1, \dots, 4\}$.

1.4 Tree-based tensor formats as multilinear models

Let us define the multilinear map $G : \bigtimes_{\alpha\in T} \mathbb{R}^{K^\alpha} \to \mathbb{R}^I$ such that for a given set of tensors $(C^\alpha)_{\alpha\in T}$, the tensor $\boldsymbol{f} = G(C^\alpha)_{\alpha\in T}$ is given by (1.2). A function $f \in \mathcal{T}_r^T(\mathcal{H})$ therefore

admits the following parametrization

$$f = F \circ G((C^\alpha)_{\alpha \in T}),$$

where $F \circ G$ is a multilinear map. We denote by $\Psi(x) : \bigtimes_{\alpha \in T} \mathbb{R}^{K^\alpha} \to \mathbb{R}$ the multilinear map such that

$$f(x) = \Psi(x)((C^\alpha)_{\alpha \in T}) \tag{1.3}$$

provides the evaluation at x of $f = F \circ G((C^\alpha)_{\alpha \in T})$, and defined by

$$\Psi(x)((C^\alpha)_{\alpha \in T}) = \sum_{\substack{i_\nu \in I^\nu \\ \nu \in \mathcal{L}(T)}} \sum_{\substack{1 \leq k_\beta \leq r_\beta \\ \beta \in T}} \prod_{\alpha \in T \setminus \mathcal{L}(T)} C^\alpha_{(k_\beta)_{\beta \in S(\alpha)}, k_\alpha} \prod_{\alpha \in \mathcal{L}(T)} C^\alpha_{i_\alpha, k_\alpha} \phi^\alpha_{i_\alpha}(x_\alpha).$$

For $\alpha \in T$ and fixed tensors $(C^\beta)_{\beta \in T \setminus \{\alpha\}}$, the partial map $\Psi^\alpha(x) : C^\alpha \mapsto \Psi(x)((C^\beta)_{\beta \in T})$ is a linear map from $\mathbb{R}^{K^\alpha}$ to $\mathbb{R}$, which is such that

$$\begin{aligned} f(x) &= \Psi^\alpha(x)(C^\alpha) \\ &= \sum_{1 \leq k_\alpha \leq r_\alpha} f^\alpha_{k_\alpha}(x_\alpha) \tilde{f}^\alpha_{k_\alpha}(x_{\alpha^c}) \\ &= \sum_{1 \leq k_\alpha \leq r_\alpha} \sum_{i_\alpha \in I^\alpha} C^\alpha_{i_\alpha, k_\alpha} \phi^\alpha_{i_\alpha}(x_\alpha) \tilde{f}^\alpha_{k_\alpha}(x_{\alpha^c}), \end{aligned} \tag{1.4}$$

where $f^D_1 = f$ and $\tilde{f}^D_1 = 1$ when $\alpha = D$, and

$$\tilde{f}^\alpha_{k_\alpha}(x_{\alpha^c}) = \sum_{\substack{1 \leq k_\delta \leq r_\delta \\ \delta \in S(\gamma) \setminus \{\alpha\}}} \sum_{1 \leq k_\gamma \leq r_\gamma} C^\gamma_{(k_\beta)_{\beta \in S(\gamma)}, k_\gamma} \prod_{\beta \in S(\gamma) \setminus \{\alpha\}} f^\beta_{k_\beta}(x_\beta) \tilde{f}^\gamma_{k_\gamma}(x_{\gamma^c}),$$

with $\gamma = P(\alpha)$, when $\alpha \neq D$. The functions $\phi^\alpha_{i_\alpha}$ depend on the tensors $\{C_\beta : \beta \in D(\alpha)\}$, with $D(\alpha) = \{\beta \in T : \alpha \in A(\beta)\}$ the set of descendants of α, while the functions $\tilde{f}^\alpha_{k_\alpha}$ depend on the tensors $\{C_\beta : \beta \in T \setminus (D(\alpha) \cup \{\alpha\})\}$.

Denoting by $\langle \cdot, \cdot \rangle_\alpha$ the canonical inner product on $\mathbb{R}^{K^\alpha}$, $\Psi^\alpha(x)$ can be identified with a tensor $\mathbf{\Psi}^\alpha(x)$ in $\mathbb{R}^{K^\alpha}$ such that

$$\Psi^\alpha(x)(C^\alpha) = \langle \mathbf{\Psi}^\alpha(x), C^\alpha \rangle_\alpha. \tag{1.5}$$

We can write $\mathbf{\Psi}^\alpha(x) = \mathbf{\Phi}^\alpha(x_\alpha)$ for $\alpha = D$ and $\mathbf{\Psi}^\alpha(x) = \mathbf{\Phi}^\alpha(x_\alpha) \otimes \tilde{\boldsymbol{f}}^\alpha(x_{\alpha^c})$ otherwise, with $\tilde{\boldsymbol{f}}^\alpha(x_{\alpha^c}) = (\tilde{f}^\alpha_i(x_{\alpha^c}))_{i=1}^{r_\alpha} \in \mathbb{R}^{r_\alpha}$.

1.5 Representation in tree-based tensor format with orthogonality conditions

The parametrization of a function $f = F \circ G((C^\alpha)_{\alpha \in T})$ in tree-based tensor format $\mathcal{T}^T_r(\mathcal{H})$ is not unique. In other terms, the map G is not injective. For a given α, orthogonality

conditions can be imposed on the parameters C^β, for all $\beta \neq \alpha$. More precisely, one can impose orthogonality conditions on matricizations of the tensors C^β so that one obtains a representation (1.4) where the set of functions

$$\boldsymbol{\Psi}^\alpha(x) = \left(\phi^\alpha_{i_\alpha}(x_\alpha)\tilde{f}^\alpha_{k_\alpha}(x_{\alpha^c})\right)_{i_\alpha \in I^\alpha, 1 \leq k_\alpha \leq r_\alpha}$$

in (1.5) forms an orthonormal system in $\mathcal{H}$. Algorithms 1 and 2 present the procedure for such an orthogonalization of the representation. They use β-matricizations $\mathcal{M}_\beta(C^\alpha)$ of tensors C^α, defined in Section 1.3 (see also [32, Section 5.2]), and the notations

$$s_\alpha = \begin{cases} \#S(\alpha) & \text{for } \alpha \in T \setminus \mathcal{L}(T) \\ 1 & \text{for } \alpha \in \mathcal{L}(T) \end{cases}$$

and i^γ_α, for $\alpha \in T \setminus \{D\}$ and $\gamma = P(\alpha)$, such that α is the i^γ_α-th child of γ. Note that in Algorithm 2, the matrix $\boldsymbol{G}^\alpha$ in Step 11 is the gramian matrix of the set of functions $\{\tilde{f}^\alpha_{k_\alpha}\}_{k_\alpha=1}^{r_\alpha}$.

Algorithm 1 Orthogonalization of the representation of a function $f = F \circ G((C^\alpha)_{\alpha \in T})$ in tree-based tensor format.

Inputs: parametrization of $f = F \circ G((C^\alpha)_{\alpha \in T})$

Outputs: new parametrization of f with orthonormal sets of functions $\{f^\alpha_{k_\alpha}\}_{k_\alpha=1}^{r_\alpha}$ for all $\alpha \in T \setminus \{D\}$

1: **for** $\alpha \in T \setminus \{D\}$ by decreasing level **do**
2: set $\gamma = P(\alpha)$
3: $\mathcal{M}_{\{s_\alpha+1\}}(C^\alpha)^T = \boldsymbol{Q}\boldsymbol{R}$ (QR factorization)
4: $\mathcal{M}_{\{s_\alpha+1\}}(C^\alpha) \leftarrow \boldsymbol{Q}^T$
5: $\mathcal{M}_{\{i^\gamma_\alpha\}}(C^\gamma) \leftarrow \boldsymbol{R}\mathcal{M}_{\{i^\gamma_\alpha\}}(C^\gamma)$
6: **end for**

1.6 Higher order singular values of tensors and tensor truncation

As seen in Section 1.3, for a non-empty subset α of D, a tensor $f \in \mathcal{H}$ can be identified with an order-two tensor $\mathcal{M}_\alpha(f)$ in $\mathcal{H}_\alpha \otimes \mathcal{H}_{\alpha^c}$, with rank $r_\alpha = \operatorname{rank}(\mathcal{M}_\alpha(f)) = \operatorname{rank}_\alpha(f)$. Such an order-two tensor admits a singular value decomposition (SVD)

$$f(x) = \sum_{i=1}^{r_\alpha} \sigma^\alpha_i u^\alpha_i(x_\alpha) v^{\alpha^c}_i(x_{\alpha^c}),$$

where the σ^α_i are the singular values and u^α_i and $v^{\alpha^c}_i$ are the corresponding left and right singular functions respectively, and where the sets of functions $\{u^\alpha_i\}_{i=1}^{r_\alpha}$ and $\{v^{\alpha^c}_i\}_{i=1}^{r_\alpha}$ form orthonormal systems in $\mathcal{H}_\alpha$ and $\mathcal{H}_{\alpha^c}$ respectively. The set of singular values of $\mathcal{M}_\alpha(f)$ is

Algorithm 2 α-orthogonalization of a function $f = F \circ G((C^\alpha)_{\alpha\in T})$ in tree-based tensor format.

Inputs: parametrization of $f = F \circ G((C^\alpha)_{\alpha\in T})$

Outputs: new parametrization of f with orthonormal functions $\{f_{k_\alpha}^\alpha\}_{k_\alpha=1}^{r_\alpha}$ and $\{\tilde{f}_{k_\alpha}^\alpha\}_{k_\alpha=1}^{r_\alpha}$ for a certain $\alpha \in T \setminus \{D\}$

apply Algorithm 1 to obtain a parametrization of f with orthonormal sets of functions $\{f_{k_\alpha}^\alpha\}_{k_\alpha=1}^{r_\alpha}$ for all $\alpha \in T \setminus \{D\}$
for $\beta \in (A(\alpha) \setminus \{D\}) \cup \{\alpha\}$ by increasing level **do**
  set $\gamma = P(\beta)$
  $B \leftarrow C^\gamma$
  if $\gamma \neq D$ **then**
    $\mathcal{M}_{\{s_\gamma+1\}}(B) \leftarrow \boldsymbol{G}^\gamma \mathcal{M}_{\{s_\gamma+1\}}(B)$
  end if
  $\boldsymbol{G}^\beta \leftarrow \mathcal{M}_{\{i_\beta^\gamma\}}(C^\gamma)\mathcal{M}_{\{i_\beta^\gamma\}}(B)^T$
end for
set $\gamma = P(\alpha)$
$\boldsymbol{G}^\alpha = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T$ (spectral decomposition of the gramian matrix of the set $\{\tilde{f}_{k_\alpha}^\alpha\}_{k_\alpha=1}^{r_\alpha}$)
$\boldsymbol{L} \leftarrow \boldsymbol{U}\sqrt{\boldsymbol{D}}$
$\mathcal{M}_{\{i_\alpha^\gamma\}}(C^\gamma) \leftarrow \boldsymbol{L}^{-1}\mathcal{M}_{\{i_\alpha^\gamma\}}(C^\gamma)$
$\mathcal{M}_{\{s_\alpha+1\}}(C^\alpha) \leftarrow \boldsymbol{L}^T\mathcal{M}_{\{s_\alpha+1\}}(C^\alpha)$

denoted by $\Sigma^\alpha(f) = \{\sigma_i^\alpha\}_{i=1}^{r_\alpha}$, and they are called α-singular values of the tensor f. Singular values are assumed to be sorted in decreasing order, *i.e.* $\sigma_1^\alpha \geq \cdots \geq \sigma_{r_\alpha}^\alpha$.

An element f_{α,m_α} of best approximation of f in the set $\mathcal{T}_{m_\alpha}^{\{\alpha\}}(\mathcal{H})$ of tensors with α-rank bounded by m_α (with $m_\alpha \leq r_\alpha$), such that

$$\|f - f_{\alpha,m_\alpha}\| = \min_{g\in\mathcal{T}_{m_\alpha}^{\{\alpha\}}(\mathcal{H})} \|f - g\|,$$

is obtained by truncating the SVD of $\mathcal{M}_\alpha(f)$ at rank m_α,

$$f_{\alpha,m_\alpha}(x) = \sum_{i=1}^{m_\alpha} \sigma_i^\alpha u_i^\alpha(x_\alpha) v_i^{\alpha^c}(x_{\alpha^c}),$$

and satisfies

$$\|f - f_{\alpha,m_\alpha}\|^2 = \sum_{i=m_\alpha+1}^{r_\alpha} (\sigma_i^\alpha)^2.$$

Denoting by $U_{m_\alpha}^\alpha$ the subspace of $\mathcal{H}_\alpha$ spanned by the left singular functions $\{u_i^\alpha\}_{i=1}^{m_\alpha}$ of $\mathcal{M}_\alpha(f)$, and by $\mathcal{P}_{U_{m_\alpha}^\alpha}^{(\alpha)} = \mathcal{M}_\alpha^{-1} \circ \left(P_{U_{m_\alpha}^\alpha}^{(\alpha)} \otimes \mathrm{id}_{\alpha^c}\right) \circ \mathcal{M}_\alpha$ the orthogonal projection from $\mathcal{H}$ onto the subspace $\left\{g = \mathcal{M}_\alpha^{-1}(f) : f \in U_{m_\alpha}^\alpha \otimes \mathcal{H}_{\alpha^c}\right\}$, we have that $f_{\alpha,m_\alpha} = \mathcal{P}_{U_{m_\alpha}^\alpha}^{(\alpha)}(f)$.

When T is a dimension partition tree over D, an approximation f_m of a tensor f in the set of tensors $\mathcal{T}_m^T(\mathcal{H})$ with T-rank bounded by $m = (m_\alpha)_{\alpha \in T}$ can be defined by

$$f_m = \mathcal{P}^{(L)} \cdots \mathcal{P}^{(1)}(f), \quad \mathcal{P}^{(\ell)} = \prod_{\substack{\alpha \in T \\ \text{level}(\alpha)=\ell}} \mathcal{P}^{(\alpha)}_{U^\alpha_{m_\alpha}}, \tag{1.6}$$

with $1 \leq \ell \leq L = \text{depth}(T)$. The approximation f_m defined by (1.6) is one possible variant of a truncated higher-order singular value decomposition (HOSVD) (see [32, Section 11.4.2], or [37] for active learning algorithms based on higher-order SVD). For a tensor f in the same tree-based tensor format (*i.e.* $f \in \mathcal{T}_r^T(\mathcal{H})$ for some $r \geq m$), the approximation f_m can be computed efficiently using standard SVD algorithms. In the following, for a tensor f with T-rank r and for a certain $m \leq r$, we denote by $f_m = \text{Truncate}(f; T, m)$ the truncated HOSVD approximation with T-rank m.

The approximation f_m obtained by the truncated HOSVD is a quasi-best approximation of f in $\mathcal{T}_m^T(\mathcal{H})$ satisfying

$$\|f - f_m\|^2 \leq \sum_{\alpha \in T \setminus \{D\}} \|f - \mathcal{P}^{(\alpha)}_{U^\alpha_{m_\alpha}}(f)\|^2 \leq (\#T - 1) \min_{g \in \mathcal{T}_m^T(\mathcal{H})} \|f - g\|^2.$$

If for all α, the truncation rank m_α is chosen such that $\|f - \mathcal{P}^{(\alpha)}_{U^\alpha_{m_\alpha}}(f)\|^2 \leq \epsilon^2 (\#T - 1)^{-1} \|f\|^2$, which means

$$\sum_{i=m_\alpha+1}^{r_\alpha} (\sigma_i^\alpha)^2 \leq \frac{\epsilon^2}{\#T - 1} \sum_{i=1}^{r_\alpha} (\sigma_i^\alpha)^2, \tag{1.7}$$

then f_m provides an approximation of f with relative precision ϵ, *i.e.*

$$\|f - f_m\| \leq \epsilon \|f\|. \tag{1.8}$$

In the following, for a tensor f and a certain $\epsilon < 1$, we denote by $f_{m(\epsilon)} = \text{Truncate}(f; T, \epsilon)$ the truncated HOSVD approximation of f with T-rank $m(\epsilon)$ chosen as the highest tuple satisfying (1.7), which ensures (1.8). Algorithm 3 presents the procedure for applying $\text{Truncate}(f; T, \epsilon)$. It is similar to Algorithm 2, with additional truncation steps.

1.7 Sparse representations in tree-based tensor format

A function f has a sparse representation in a certain tree-based tensor format if it admits a parametrization (1.3) where parameters C^α contain zero entries. When the parameter C^α is an array containing the coefficients of functions on a given basis of functions $\mathbf{\Phi}^\alpha(x_\alpha) = (\phi_i^\alpha(x_\alpha))_{i \in I^\alpha}$ (see Section 1.4), sparsity in C^α means sparsity of the corresponding functions relatively to the basis $\mathbf{\Phi}^\alpha(x_\alpha)$. The choice of the bases is crucial for the existence of a sparse representation of a tensor (or of an accurate approximation with a sparse representation). For leaf nodes $\alpha \in \mathcal{L}(T)$, typical choices of bases $\mathbf{\Phi}^\alpha$ include polynomials, wavelets or other bases for multiresolution analysis, where the set of basis functions

Algorithm 3 Algorithm of the function Truncate($f; T, \epsilon$).

Inputs: parameters $(C^\alpha)_{\alpha \in T}$ of a function $f \in \mathcal{T}_r^T(\mathcal{H})$

Outputs: approximation $f_m \in \mathcal{T}_m^T(\mathcal{H})$ satisfying (1.8)

use Algorithm 1 to obtain orthonormal sets of functions $\{f_{k_\alpha}^\alpha\}_{k_\alpha=1}^{r_\alpha}$, $\alpha \in T \setminus \{D\}$
for $\alpha \in T \setminus \{D\}$ by increasing level **do**
  set $\gamma = P(\alpha)$
  $B \leftarrow C^\gamma$
  if $\gamma \neq D$ **then**
    $\mathcal{M}_{\{s_\gamma+1\}}(B) \leftarrow \boldsymbol{G}^\gamma \mathcal{M}_{\{s_\gamma+1\}}(B)$
  end if
  $\boldsymbol{G}^\alpha \leftarrow \mathcal{M}_{\{i_\alpha^\gamma\}}(C^\gamma)\mathcal{M}_{\{i_\alpha^\gamma\}}(B)^T$ with eigenvalues $((\sigma_i^\alpha)^2)_{i=1}^{r_\alpha}$
end for
for $\alpha \in T \setminus \{D\}$ by decreasing level **do**
  $\boldsymbol{G}^\alpha \approx \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T$ such that (1.7) is ensured (truncated spectral decomposition of the gramian matrix of the set of functions $\{\tilde{f}_{k_\alpha}^\alpha\}_{k_\alpha=1}^{r_\alpha}$ at rank m_α)
  $\mathcal{M}_{\{i_\alpha^\gamma\}}(C^\gamma) \leftarrow \boldsymbol{U}^T \mathcal{M}_{\{i_\alpha^\gamma\}}(C^\gamma)$
  $\mathcal{M}_{\{s_\alpha+1\}}(C^\alpha) \leftarrow \boldsymbol{U}^T \mathcal{M}_{\{s_\alpha+1\}}(C^\alpha)$
end for

can be partitioned into subsets of basis functions with different levels (or resolutions). For interior nodes $\alpha \in T \setminus \mathcal{L}(T)$, bases $\boldsymbol{\Phi}^\alpha$ introducing sparsity for the representation of a function f can be obtained by using for the bases $\{f_{k_\beta}^\beta\}$ of minimal subspaces $U_\beta^{\min}(f)$ the principal components of α-matricizations of f obtained with singular value decompositions (ordered by decreasing singular values). Algorithm 3 with $\epsilon = 0$ yields such a representation with a natural hierarchy in the bases $\{f_1^\beta, \ldots, f_{r_\beta}^\beta\}$, which induces a natural hierarchy in the bases $\boldsymbol{\Phi}^\alpha$ of interior nodes that are obtained through tensorization of bases $\{f_1^\beta, \ldots, f_{r_\beta}^\beta\}$, $\beta \in S(\alpha)$.

For such bases with a natural hierarchy of the basis functions, we introduce a nested sequence $I_0^\alpha \subset \cdots \subset I_p^\alpha$ of subsets in I^α, with $I_p^\alpha = I^\alpha$, such that $\{\phi_i^\alpha(x_\alpha)\}_{i \in I_\lambda^\alpha}$ is a basis of a subspace of functions $\mathcal{H}_\alpha^\lambda \subset \mathcal{H}_\alpha$, with level λ. This defines a sequence of nested spaces

$$\mathcal{H}_\alpha^0 \subset \cdots \subset \mathcal{H}_\alpha^p \subset \mathcal{H}_\alpha.$$

Example 1.7.1 (Hierarchical bases for the leaves $\alpha \in \mathcal{L}(T)$)**.**

- *For a univariate polynomial basis, where ϕ_i^α is a polynomial of degree $i-1$, we simply take $I_\lambda^\alpha = \{1, \ldots, \lambda+1\}$, so that $\mathcal{H}_\alpha^\lambda = \mathbb{P}_\lambda(\mathcal{X}_\alpha)$ is the space of polynomials with degree λ.*
- *For a multivariate polynomial basis on $\mathcal{X}_\alpha \subset \mathbb{R}^{d_\alpha}$, $\mathcal{H}_\alpha^\lambda$ can be chosen as the space of polynomials with partial (or total) degree λ.*

- *For a univariate wavelet (or multiresolution) basis, $\mathcal{H}_\alpha^\lambda$ can be taken as the space of functions with resolution λ.*

For a tensor f in tree-based tensor format and a node $\alpha \in T$, where $C^\alpha \in \mathbb{R}^{K^\alpha}$ collects the coefficients of the functions $\{f_k^\alpha(x_\alpha)\}_{k=1}^{r_\alpha}$ on a basis $\mathbf{\Phi}^\alpha(x_\alpha)$, we define

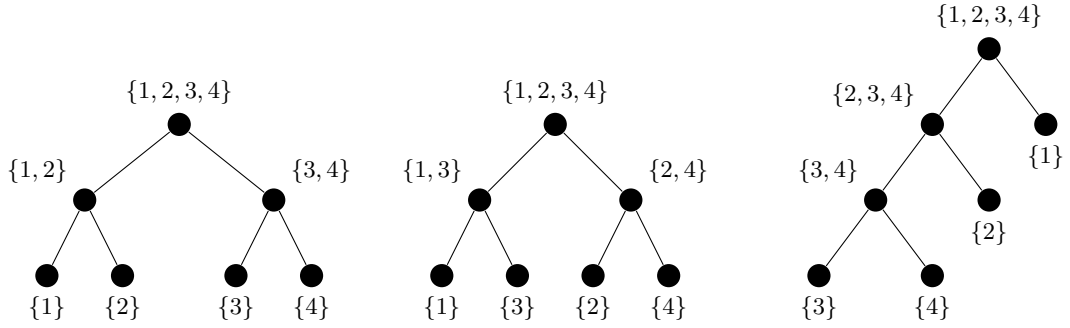
$$K_\lambda^\alpha = I_\lambda^\alpha \times \{1, \ldots, r_\alpha\},$$

so that $K_0^\alpha \subset \cdots \subset K_p^\alpha$ form a nested sequence of subsets in $K^\alpha = I^\alpha \times \{1, \ldots, r_\alpha\}$, with $K_p^\alpha = K^\alpha$. Therefore, if $C^\alpha = (C_k^\alpha)_{k \in K^\alpha}$ is such that $C_k^\alpha = 0$ for all $k \notin K_\lambda^\alpha$, then all functions $f_k^\alpha(x_\alpha)$ are in the space $\mathcal{H}_\alpha^\lambda$, and $f \in \mathcal{H}_\alpha^\lambda \otimes \mathcal{H}_{\alpha^c}$.

The proposed sequence of candidate patterns $K_0^\alpha \subset \cdots \subset K_p^\alpha$ for the parameter C^α can be used in learning algorithms using working set strategies for sparse approximation (see Section 2.5).

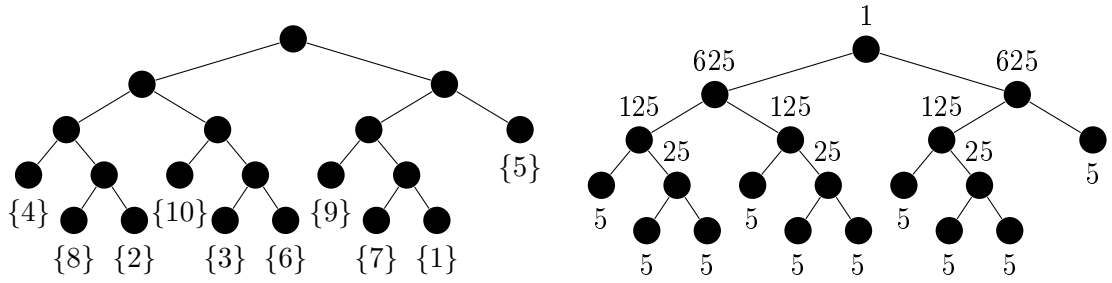
1.8 Dimension tree optimization

The complexity of the representation of a function in tree-based tensor format strongly depends on the selection of the dimension tree. The number of possible dimension trees being exponential in the dimension d, the selection of an optimal tree is intractable in high dimension. We here propose a heuristic stochastic algorithm for the optimization of the tree, within a class of trees having the same arity. The proposed approach consists in comparing trees obtained by successive permutations of nodes drawn randomly according to a suitable probability distribution. Starting from a given tree, the proposed strategy allows the exploration of a very large class of trees obtained by an arbitrary number of permutations of nodes. In particular, starting from a binary tree, the strategy is able to explore the whole set of binary trees over D (including all balanced and linear trees). For example, a single permutation of nodes allows to go from the tree T of Figure 1.3a to the tree T' of Figure 1.3b, with the same topology, but also to the tree T'' of Figure 1.3c with a different topology.



(a) Tree T. (b) Tree T'. (c) Tree T''.

Figure 1.3: Dimension tree T over $D = \{1, 2, 3, 4\}$, tree T' obtained by permuting the nodes $\{2\}$ and $\{3\}$ of T, tree T'' obtained by permuting the nodes $\{1\}$ and $\{3, 4\}$ of T.



(a) Dimension tree. (b) Representation ranks.

Figure 1.4: Representation in tree-based format with an initial random tree. The storage complexity is 10595875.

1.8.1 Motivating example

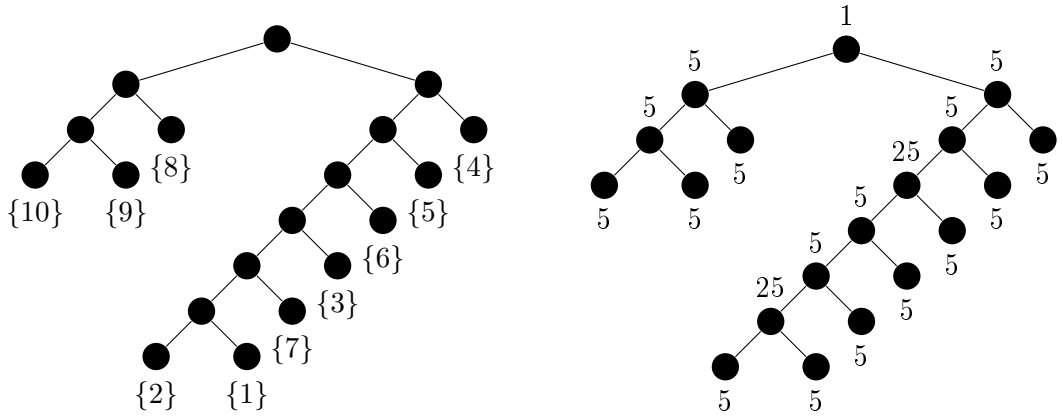
We consider, in dimension $d = 10$, discrete random variables X_ν taking $N = 5$ possible instances, so that $f(x)$ is identified with a tensor of size $N^d = 5^{10} = 9765625$ (see Remark 1.2.2). We assume $f(x)$ has the form

$$f(x) = g_{1,2,3,7}(x_1, x_2, x_3, x_7) g_{3,4,5,6}(x_3, x_4, x_5, x_6) g_{4,8}(x_4, x_8) g_{8,9,10}(x_8, x_9, x_{10}).$$

The values taken by the different functions g_α are drawn randomly in $[0, 1]$, and are normalized such that $\sum_{i \in I} f(x^i) = 1$.

We first consider the random binary tree of Figure 1.4 and compute a representation of f in the corresponding tree-based format, using a truncation algorithm at precision 10^{-13} (as described in Section 1.6). We observe a storage complexity of 10595875, higher than the storage complexity of the full tensor. After the tree optimization (with Algorithm 5, presented below), we obtain the tree in Figure 1.5 with a storage complexity of 3275. We see that this latter tree yields a representation with a storage complexity much smaller than with the former, and contains nodes that correspond to the groups of variables involved in the different functions g_α appearing in the definition of f.

This example shows the major influence of the choice of the tree on the storage complexity of the representation of a function in tree-based tensor format. Furthermore, the dimension tree yielding the smallest complexity can carry information about the represented function, for instance about the dependence structure of a probabilistic model, as we shall see in Chapter 2 when learning functions with tree-based tensor formats. These observations motivate the development of the proposed algorithm to perform tree optimization.



(a) Dimension tree. (b) Representation ranks.

Figure 1.5: Representation in tree-based format after tree optimization. The storage complexity is 3275.

1.8.2 Tree optimization for the representation of a given function

Let T be a given partition tree and consider a given function $f \in \mathcal{H}$. Letting $r = \text{rank}_T(f)$, we have that $f \in \mathcal{T}_r^T(\mathcal{H})$ admits a representation

$$f(x) = \sum_{\substack{1 \le i_\nu \le N_\nu \\ \nu \in \mathcal{L}(T)}} \sum_{\substack{1 \le k_\beta \le r_\beta \\ \beta \in T}} \prod_{\alpha \in T \setminus \mathcal{L}(T)} C^\alpha_{(k_\beta)_{\beta \in S(\alpha)}, k_\alpha} \prod_{\alpha \in \mathcal{L}(T)} C^\alpha_{i_\alpha, k_\alpha} \phi^\alpha_{i_\alpha}(x_\alpha), \tag{1.9}$$

with storage complexity[1]

$$C(T, r) = \sum_{\alpha \in T \setminus \mathcal{L}(T)} r_\alpha \prod_{\beta \in S(\alpha)} r_\beta + \sum_{\alpha \in \mathcal{L}(T)} N_\alpha r_\alpha.$$

If $r_\alpha = O(R)$ and $\dim(\mathcal{H}_\alpha) = O(N)$, then since $\#T = O(d)$, $C(T, r) = O(dNR + (\#T - d - 1)R^{s+1} + R^s)$, where $s = \max_{\alpha \in T \setminus \mathcal{L}(T)} \#S(\alpha)$ is the arity of the tree. For a binary tree, $s = 2$ and $\#T = 2d - 1$, so that $C(T, r) = O(dNR + (d-2)R^3 + R^2)$.

Then, we would like to find a tree solution of

$$\min_T C(T, \text{rank}_T(f)), \tag{1.10}$$

which minimizes over a set of dimension trees the storage complexity for the function f. In practice, when f is an approximation of a target function, we may be only interested in obtaining an approximation of f with a certain precision (*e.g.* related to an estimation of the generalization error of f) and with minimal storage complexity. Then problem (1.10) can be replaced by

$$\min_T C(T, \text{rank}^\epsilon_T(f)), \tag{1.11}$$

[1] Note that the possible sparsity of the representation in a given tensor format is not (but could be) taken into account for defining an effective storage complexity.

with $\text{rank}_T^\epsilon(f) = (\text{rank}_\alpha^\epsilon(f))_{\alpha \in T}$, where the ϵ-rank $\text{rank}_\alpha^\epsilon(f)$ is defined as the minimal integer r_α^ϵ such that there exists an approximation f^ϵ with $\text{rank}_\alpha(f^\epsilon) = r_\alpha^\epsilon$ and such that $\|f - f^\epsilon\| \leq \epsilon \|f\|$.

For solving (1.10), we propose a stochastic algorithm which successively compares the current tree T with a new tree $\widetilde{T}$ drawn from a suitable probability distribution over the set of trees, obtained by successive random permutations of nodes, and accepts the tree $\widetilde{T}$ if it yields a lower storage complexity for f (at relative precision ϵ). The probability distribution, defined below, gives a higher probability to trees $\widetilde{T}$ presenting the highest potential reduction of storage complexity (by preferably permuting nodes α whose parents have the highest ranks $r_{P(\alpha)}$) but the lowest computational complexity for changing the representation of f from T to $\widetilde{T}$.

Before presenting the stochastic algorithm and how to draw randomly a new tree $\widetilde{T}$, we first detail how to change the representation of a function by a permutation of two nodes in a current tree T. This will allow us to introduce a notion of computational complexity for the changes of representations.

1.8.3 Changing the representation by permutations of two nodes

For two nodes ν and μ in a tree T such that $\nu \cap \mu = \emptyset$ (*i.e.* one node is not the ascendant of the other), we denote by $\sigma_{\nu,\mu}$ the map such that $\sigma_{\nu,\mu}(T)$ is the tree obtained from T by a permutation of nodes ν and μ. Let $P(\alpha;T)$, $S(\alpha;T)$, $A(\alpha;T)$ and $\text{level}(\alpha;T)$ be the parent, children, ascendants and level of α in T, respectively. In the tree $\sigma_{\nu,\mu}(T)$, we have that $P(\nu;\sigma_{\nu,\mu}(T)) = (P(\mu;T) \setminus \mu) \cup \nu$ and $P(\mu;\sigma_{\nu,\mu}(T)) = (P(\nu;T) \setminus \nu) \cup \mu$. The map $\sigma_{\nu,\mu}$ only modifies the nodes of T in $(A(\mu;T) \cup A(\nu;T)) \setminus (A(\mu;T) \cap A(\nu;T))$, which is the set of all ascendants of ν or μ that are not common ascendants of these two nodes. In particular, we have that $\sigma_{\nu,\mu}(T) = T$ if $P(\mu;T) = P(\nu;T)$. Let

$$\gamma := \underset{\beta \in A(\nu;T) \cap A(\mu;T)}{\text{argmax}} \text{level}(\beta;T) \tag{1.12}$$

denote the highest-level common ascendant of ν and μ in T and let

$$T^{\nu,\mu} = (A(\nu;T) \cup A(\mu;T)) \setminus (\{\gamma\} \cup A(\gamma;T)) \tag{1.13}$$

be the subset of T containing all the ascendants of ν and μ up to γ, except γ. The tree $\widetilde{T} = \sigma_{\nu,\mu}(T)$ is such that $\widetilde{T} = (T \setminus T^{\nu,\mu}) \cup \widetilde{T}^{\nu,\mu}$.

The representation (1.4) of f can be written

$$f(x) = \sum_{\substack{1 \leq k_\delta \leq r_\delta \\ \delta \in S(T^{\nu,\mu})}} \sum_{1 \leq k_\gamma \leq r_\gamma} M^\gamma_{(k_\beta)_{\beta \in S(T^{\nu,\mu})}, k_\gamma} \prod_{\beta \in S(T^{\nu,\mu})} f^\beta_{k_\beta}(x_\beta) \tilde{f}^\gamma_{k_\gamma}(x_{\gamma^c})$$

with $S(T^{\nu,\mu}) = \{\alpha \in T \setminus T^{\nu,\mu} : \alpha \in S(\beta;T), \beta \in T^{\nu,\mu}\}$, and where

$$M^{\gamma}_{(k_\beta)_{\beta\in S(T^{\nu,\mu})},k_\gamma} = \sum_{\substack{1\le k_\delta\le r_\delta\\ \delta\in T^{\nu,\mu}}} \prod_{\alpha\in T^{\nu,\mu}\cup\{\gamma\}} C^{\alpha}_{(k_\eta)_{\eta\in S(\alpha;T)},k_\alpha} \tag{1.14}$$

are the components of a tensor M^γ of order $\#S(T^{\nu,\mu})+1$. Assuming that f has a γ-orthogonal representation (1.9), the functions

$$\left\{\prod_{\beta\in S(T^{\nu,\mu})} f^{\beta}_{k_\beta}(x_\beta)\right\}_{\substack{1\le k_\beta\le r_\beta\\ \beta\in S(T^{\nu,\mu})}} \quad \text{and} \quad \left\{\tilde{f}^{\gamma}_{k_\gamma}(x_{\gamma^c})\right\}_{1\le k_\gamma\le r_\gamma}$$

are orthonormal in $\mathcal{H}_\gamma$ and $\mathcal{H}_{\gamma^c}$ respectively, and we have $\|M^\gamma\| = \|f\|$. Up to a change in the ordering of the children of each node, the expression (1.14) can be rewritten, introducing the parents $\gamma^\nu = P(\nu;T)$ and $\gamma^\mu = P(\mu;T)$ of ν and μ in T, as

$$M^{\gamma}_{(k_\beta)_{\beta\in S(T^{\nu,\mu})},k_\gamma} = \sum_{\substack{1\le k_\delta\le r_\delta\\ \delta\in T^{\nu,\mu}}} C^{\gamma^\nu}_{(k_\eta)_{\eta\in S(\gamma^\nu;T)\setminus\{\nu\}},k_\nu,k_{\gamma^\nu}} C^{\gamma^\mu}_{(k_\eta)_{\eta\in S(\gamma^\mu;T)\setminus\{\mu\}},k_\mu,k_{\gamma^\mu}} \prod_{\alpha\in\{\gamma\}\cup(T^{\nu,\mu}\setminus\{\gamma^\nu,\gamma^\mu\})} C^{\alpha}_{(k_\eta)_{\eta\in S(\alpha;T)},k_\alpha}.$$

Also, the tensor M^γ can be identified with

$$M^{\gamma}_{(k_\beta)_{\beta\in S(T^{\nu,\mu}_{\tilde\sigma})},k_\gamma} = \sum_{\substack{1\le k_\delta\le \tilde r_\delta\\ \delta\in T^{\nu,\mu}}} \widetilde{C}^{\gamma^\nu}_{(k_\eta)_{\eta\in S(\gamma^\nu;T)\setminus\{\nu\}},k_\mu,k_{\gamma^\nu}} \widetilde{C}^{\gamma^\mu}_{(k_\eta)_{\eta\in S(\gamma^\mu;T)\setminus\{\mu\}},k_\nu,k_{\gamma^\mu}} \prod_{\alpha\in\{\gamma\}\cup(T^{\nu,\mu}\setminus\{\gamma^\nu,\gamma^\mu\})} \widetilde{C}^{\alpha}_{(k_\eta)_{\eta\in S(\alpha;T)},k_\alpha}, \tag{1.15}$$

where $\widetilde{C}^{\gamma^\nu}$ now has an index k_μ, and $\widetilde{C}^{\gamma^\mu}$ an index k_ν. The representation (1.15) of M^γ corresponds to a representation of f in $\mathcal{T}^{\widetilde{T}}_{\tilde r}(\mathcal{H})$ with $\widetilde{T} = \sigma_{\nu,\mu}(T)$ and $\tilde r_\beta = r_\beta$ for all $\beta \in T \setminus T^{\nu,\mu}$.

Algorithm 4 presents the permutation procedure, showing how to practically obtain an approximation in $\mathcal{T}^{\widetilde{T}}_{\tilde r}(\mathcal{H})$ of $f \in \mathcal{T}^T_r(\mathcal{H})$ with relative precision ϵ. At Step 10, the singular value decomposition of a matricization of $M^{\gamma}_{(k_\beta)_{\beta\in S_\gamma},k_\gamma}$ is computed:

$$M^{\gamma}_{(k_\beta)_{\beta\in S_\gamma},k_\gamma} = \sum_{k_\eta=1}^{\tilde r_\eta} \tilde\sigma_{k_\eta} \tilde a_{(k_\beta)_{\beta\in S_\gamma\setminus S(\eta;\widetilde{T})},k_\eta,k_\gamma} \tilde b_{(k_\beta)_{\beta\in S(\eta;\widetilde{T})},k_\eta}, \tag{1.16}$$

with singular values $\{\tilde\sigma_{k_\eta}\}_{k_\eta=1}^{\tilde r_\eta}$. We define $\tilde r^{\epsilon'}_\eta$ as the minimal integer ensuring

$$\sum_{k_\eta=\tilde r^{\epsilon'}_\eta+1}^{\tilde r_\eta} \tilde\sigma^2_{k_\eta} \le \epsilon'^2 \sum_{k_\eta=1}^{\tilde r_\eta} \tilde\sigma^2_{k_\eta} \tag{1.17}$$

with $\epsilon'^2 = \epsilon^2/\#T^{\nu,\mu}$, where $\#T^{\nu,\mu}$ is the number of singular value decompositions required for changing the representation of a function from tree T to tree $\sigma_{\nu,\mu}(T)$. The integers $(\tilde{r}_\alpha^{\epsilon'})_{\alpha\in\widetilde{T}}$ correspond to the ϵ'-ranks of f in the format associated with the tree $\sigma_{\nu,\mu}(T)$. By truncating (1.16) at rank $\tilde{r}_\eta^{\epsilon'}$, we obtain an approximation of f with relative precision ϵ', ultimately leading to an approximation of f with relative precision ϵ once the $\#T^{\nu,\mu}$ singular value decompositions are performed.

Algorithm 4 Change of representation of a tree-based tensor $f \in \mathcal{T}_r^T(\mathcal{H})$ by a permutation of two nodes ν and μ in T at precision ϵ.

Inputs: initial T, representation of the tensor $f \in \mathcal{T}_r^T(\mathcal{H})$ with T-rank r, nodes ν and μ to be permuted, such that $\nu \cap \mu = \emptyset$, relative precision ϵ.

Outputs: approximation of the tensor f in $\mathcal{T}_{\tilde{r}}^{\widetilde{T}}(\mathcal{H})$ with $\widetilde{T} = \sigma_{\nu,\mu}(T)$ and $\widetilde{T}$-rank $\tilde{r}$

compute γ defined by (1.12) and $T^{\nu,\mu}$ defined by (1.13)
perform a γ-orthogonalization of the representation of f using Algorithm 2
$M^\gamma \leftarrow C^\gamma$, $S_\gamma \leftarrow S(\gamma; T)$
for $\eta \in T^{\nu,\mu}$ by increasing level **do**
  $M^\gamma_{(k_\beta)_{\beta\in S_\gamma\setminus\{\eta\}},(k_\beta)_{\beta\in S(\eta;T)},k_\gamma} \leftarrow \sum_{k_\eta=1}^{r_\eta} M^\gamma_{(k_\beta)_{\beta\in S_\gamma},k_\gamma} C^\eta_{(k_\beta)_{\beta\in S(\eta;T)},k_\eta}$
  $S_\gamma \leftarrow (S_\gamma \setminus \{\eta\}) \cup S(\eta; T)$
end for
$\widetilde{T} \leftarrow \sigma_{\nu,\mu}(T)$
for $\eta \in \widetilde{T}^{\nu,\mu}$ by decreasing level **do**
  compute the SVD (1.16) of the matricization $\mathcal{M}_{\{i_\beta^\gamma\}_{\beta\in S(\eta;\widetilde{T})}}(M^\gamma)^T$ of M^γ
  truncate the SVD at rank $\tilde{r}_\eta^{\epsilon'}$, the minimal integer satisfying (1.17)
  $\tilde{M}^\gamma_{(k_\beta)_{\beta\in S_\gamma\setminus S(\eta;\widetilde{T})},k_\eta,k_\gamma} \leftarrow \tilde{\sigma}_{k_\eta}\tilde{a}_{(k_\beta)_{\beta\in S_\gamma\setminus S(\eta;\widetilde{T})},k_\eta,k_\gamma}, \quad k_\eta = 1,\dots,\tilde{r}_\eta^{\epsilon'}$
  $\widetilde{C}^\eta_{(k_\beta)_{\beta\in S(\eta;\widetilde{T})},k_\eta} \leftarrow \tilde{b}_{(k_\beta)_{\beta\in S(\eta;\widetilde{T})},k_\eta}, \quad k_\eta = 1,\dots,\tilde{r}_\eta^{\epsilon'}$
  $\tilde{r}_\eta \leftarrow \tilde{r}_\eta^{\epsilon'}$
  $M^\gamma \leftarrow \tilde{M}^\gamma$
  $S_\gamma \leftarrow (S_\gamma \cup \{\eta\}) \setminus S(\eta; \widetilde{T})$
end for
$\widetilde{C}^\gamma \leftarrow M^\gamma$, $\tilde{r}_\gamma \leftarrow r_\gamma$, $\widetilde{C}^\beta \leftarrow C^\beta, \tilde{r}_\beta \leftarrow r_\beta, \forall\beta \in \widetilde{T} \setminus \widetilde{T}^{\nu,\mu}$ with $\beta \neq \gamma$
return the approximation of the tensor f in $\mathcal{T}_{\tilde{r}}^{\widetilde{T}}(\mathcal{H})$ given by

$$\sum_{\substack{1\le k_\beta\le\tilde{r}_\beta\\ \beta\in\widetilde{T}}} \prod_{\alpha\in\widetilde{T}\setminus\mathcal{L}(\widetilde{T})} \widetilde{C}^\alpha_{(k_\beta)_{\beta\in S(\alpha;\widetilde{T})},k_\alpha} \prod_{\alpha\in\mathcal{L}(\widetilde{T})} f^\alpha_{k_\alpha}(x_\alpha).$$

1.8.4 Stochastic algorithm for the tree optimization

To describe the stochastic algorithm for tree optimization, it remains to detail how to draw a new tree $\widetilde{T}$ given a current tree T. The tree $\widetilde{T}$ is defined as $\widetilde{T} = \sigma_m \circ \cdots \circ \sigma_1(T)$, where

the map $\sigma_i = \sigma_{\nu_i,\mu_i}$ permutes two nodes of the tree $T_{i-1} = \sigma_{i-1} \circ \cdots \circ \sigma_1(T)$, with $T_0 := T$. The number of permutations m is first drawn according to the distribution

$$\mathbb{P}(m = k) \propto k^{-\gamma_1}, \quad k \in \mathbb{N}^*, \tag{1.18}$$

with $\gamma_1 > 0$, which gives a higher probability to low numbers of permutations. Then, given T_i, $\sigma_{i+1} = \sigma_{\nu_{i+1},\mu_{i+1}}$ is defined by first drawing randomly the node ν_{i+1} in $T_i \setminus \{D\}$ according to the distribution

$$\mathbb{P}(\nu_{i+1} = \alpha | T_i) \propto \mathrm{rank}_{P(\alpha;T_i)}(f)^{\gamma_2}, \quad \alpha \in T_i \setminus \{D\}, \tag{1.19}$$

with $P(\alpha; T_i)$ the parent of α in T_i and $\gamma_2 > 0$. This gives a higher probability to select a node ν_{i+1} with a high parent's rank (note that $P(\nu_{i+1}; T_i)$ will not be in the next tree T_{i+1}). Then, given the first node ν_{i+1}, we draw the second node μ_{i+1} in T_i according to the distribution

$$\mathbb{P}(\mu_{i+1} = \alpha | T_i, \nu_{i+1}) \propto \begin{cases} d_{T_i}(\nu_{i+1}, \alpha)^{-\gamma_3} & \text{if } \alpha \cap \nu_{i+1} = \emptyset \\ 0 & \text{otherwise} \end{cases}, \tag{1.20}$$

where $\gamma_3 > 0$ and

$$d_{T_i}(\nu, \mu) = r_\gamma \prod_{\beta \in S(T_i^{\nu,\mu})} r_\beta$$

is the storage complexity of the full tensor M^γ of order $\#S(T_i^{\nu,\mu}) + 1$ (defined in equation (1.14) with T replaced by T_i) which is computed when changing the representation of the function f from the tree T_i to the tree $\sigma_{\nu,\mu}(T_i)$. This choice of probability distribution therefore gives a higher probability to modifications of the tree with low computational complexity. The stochastic algorithm for solving (1.11) is finally given in Algorithm 5.

Remark 1.8.1. *The parameters $\gamma_1, \gamma_2, \gamma_3$ of the above probability distributions have an impact on the computational complexity and the ability to try large modifications of the current tree. In practice, we choose $\gamma_1 = \gamma_2 = \gamma_3 = 2$ in the numerical experiments but the choice of these parameters should deserve a deeper analysis.*

1.9 Conclusion

We presented in this chapter the model class of functions in tree-based tensor format, and related notions used in Chapter 2 to propose learning algorithms with this format.

The main contribution of this chapter is Algorithm 5 to perform dimension tree adaptation. This adaptive algorithm is able, by performing random changes in the tree, to reduce the storage complexity in tree-based tensor format of a given function.

We showed in a motivating example in Section 1.8.1 that this algorithm can have a significant impact on the complexity of the representation in tree-based tensor format of a function at

a given accuracy. We also noticed that the tree yielding the smallest storage complexity can give some information about the structure of the function to represent. As we shall see with the numerical experiments in Chapter 2, this means that the trees associated with the obtained approximations in tree-based tensor format can give insights into some structures of the functions to learn (*e.g.* block independence in the context of density estimation).

The probability to find an optimal tree could be further improved by proposing better probability distributions over the set of possible trees, possibly based on other definitions of the complexity. Also, a more general algorithm for tree adaptation should allow modifications of the arity of the tree. In particular, it could make possible the transition from a binary tree to a trivial tree, and by exploiting sparsity in tensor representations, it could create a bridge between sparse and low-rank tensor approximations.

Algorithm 5 Tree optimization algorithm for the representation of a given tensor f at precision ϵ.

Inputs: initial tree T, representation of the tensor $f \in \mathcal{T}_r^T(\mathcal{H})$ with T-rank r and storage complexity $C(T,r)$, precision ϵ, number of random trials N

Outputs: new tree $T^\star$ and approximation $f^\star$ of f in $\mathcal{T}_{r^\star}^{T^\star}(\mathcal{H})$ with new $T^\star$-rank $r^\star$ and storage complexity $C(T^\star, r^\star) \leq C(T,r)$

$C^\star \leftarrow C(T,r)$, $T^\star \leftarrow T$, $f^\star \leftarrow f$, $\widetilde{T} \leftarrow T$, $\tilde{r} \leftarrow r$, $m \leftarrow 0$
$\Sigma \leftarrow \emptyset$, $newtree = false$
for $k = 1, \ldots, N$ **do**
  $m_{\text{old}} \leftarrow m$
  draw randomly m drawn according to (1.18)
  if $m > m_{\text{old}}$ or $newtree$ **then**
    compute an approximation f_0 of f by applying successively all permutations in Σ using Algorithm 4 with precision $\epsilon/(m + \#\Sigma)$
    $T_0 \leftarrow T^\star$
  end if
  for $i = 0, \ldots, m$ **do**
    draw $\sigma_{i+1} = \sigma_{\nu_{i+1},\mu_{i+1}}$ according to (1.19) and (1.20) (with T replaced by T_i)
    $T_{i+1} \leftarrow \sigma_{i+1}(T_i)$
    compute an approximation f_{i+1} of f_i by applying the permutation σ_{i+1} using Algorithm 4 with precision $\epsilon/(m + \#\Sigma)$
  end for
  $\widetilde{T} \leftarrow T_m$, $\tilde{f} \leftarrow f_m$, $\tilde{r} \leftarrow \text{rank}_{\widetilde{T}}(\tilde{f})$
  if $C(\widetilde{T}, \tilde{r}) < C^\star$ **then**
    $newtree = true$
    $T^\star \leftarrow \widetilde{T}$, $C^\star \leftarrow C(\widetilde{T}, \tilde{r})$, $f^\star \leftarrow \tilde{f}$
    $\Sigma \leftarrow \Sigma \cup \{\sigma_1, \ldots, \sigma_m\}$ (ordered set)
  else
    $newtree = false$
  end if
end for

Chapter 2

Learning with tree-based tensor formats

Contents

2.1 Introduction

In this chapter, we propose adaptive algorithms for learning with functions in tree-based tensor format. We begin by introducing the notions of contrast and risk in Section 2.2, then we propose in Section 2.3 an algorithm to solve the empirical risk minimization problem with model classes of functions in tree-based tensor format, with given dimension tree and tree-based rank. Section 2.4 describes the used tools to evaluate the performances of the approximations and to perform model selection, that can be used when introducing sparsity in the coefficients of the representation, as presented in Section 2.5. We then present in Section 2.6 the main contributions of this chapter: adaptive learning algorithms with tree-based tensor formats. These algorithms are able to compute, given a training sample, an approximation in tree-based tensor format $\mathcal{T}_r^T(\mathcal{H})$ of a function, where both the dimension tree T and the tree-based rank r are adapted. The performances of the proposed algorithms are illustrated in Section 2.7 on numerical experiments in two classical contexts: least-squares regression (supervised learning) and least-squares density estimation (unsupervised learning).

This chapter is partly based on two articles [33, 34] for the learning with model classes of functions in tree-based tensor format.

2.2 Contrast function and risk

We introduce a contrast function $\gamma : L^0_\mu(\mathcal{X}) \times \mathcal{Z} \to \mathbb{R}$, which is so that $\gamma(g, z)$ measures the error due to the use of the approximation g for a sample z.

We denote by $S = \{z_i\}_{i=1}^n$ the training sample, consisting of realizations of a random variable Z with values in $\mathcal{Z}$, defined below in the cases of supervised learning and density estimation. An approximation can be obtained by minimizing the empirical risk $\mathcal{R} : L^0_\mu(\mathcal{X}) \to \mathbb{R}$ that writes

$$\mathcal{R}_n(g) = \frac{1}{n} \sum_{z=1}^{n} \gamma(g, z_i)$$

over a certain subset of functions g (hypothesis set), here the class of functions in tree-based tensor format. The empirical risk is a statistical estimate of the risk $\mathcal{R} : L^0_\mu(\mathcal{X}) \to \mathbb{R}$ defined by

$$\mathcal{R}(g) = \mathbb{E}(\gamma(g, Z)).$$

In the context of supervised learning, $Z = (X, Y)$, $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, and the aim is to construct an approximation of Y as a function of X (predictive model), from a set $S = \{z_i\}_{i=1}^n$ of n realizations of (X, Y). In least-squares regression, the contrast is taken as

$$\gamma(g, (x, y)) = (y - g(x))^2, \tag{2.1}$$

and

$$\mathcal{R}(g) = \mathbb{E}((Y - f(X))^2) + \mathbb{E}((f(X) - g(X))^2) = \mathcal{R}(f) + \|f - g\|^2,$$

where $\|\cdot\|$ is the L^2_μ-norm (with μ the probability law of X) and $f(X) = \mathbb{E}(Y|X)$ is the best approximation of Y by a measurable function of X, so that the minimization of the risk corresponds to the minimization of the approximation error $\|f - g\|^2$. Other contrast functions could be considered for other purposes in supervised learning (such as hinge loss or logistic loss for classification).

In the context of density estimation, $Z = X$ is a random variable that is assumed to have a density f with respect to some measure μ over $\mathcal{X}$ (possibly discrete) that we want to approximate, $\mathcal{Z} = \mathcal{X}$ and $S = \{z_i\}_{i=1}^n$ is a set of n realizations of X. Choosing the contrast function as

$$\gamma(g, x) = \|g\|^2 - 2g(x) \tag{2.2}$$

yields

$$\mathcal{R}(g) = \mathcal{R}(f) + \|f - g\|^2,$$

so that the minimization of $\mathcal{R}(g)$ is equivalent to the minimization of the distance (in L^2_μ norm) between g and the density f. Choosing $\gamma(g, z) = -\log(g(z))$ leads to

$$\mathcal{R}(g) = \mathcal{R}(f) + D_{\mathrm{KL}}(f\|g),$$

with $D_{\mathrm{KL}}(f\|g)$ the Kullback-Leibler divergence between f and g, and the empirical risk minimization corresponds to a maximum likelihood estimation.

2.3 Empirical risk minimization

An approximation of f in a tensor format can be obtained by minimizing the empirical risk $\mathcal{R}_n(g)$ over the set of functions

$$g(x) = \Psi(x)((C^\alpha)_{\alpha \in T}),$$

where the $(C^\alpha)_{\alpha \in T} \in \bigtimes_{\alpha \in T} \mathbb{R}^{K^\alpha}$ are the parameters of the representation of g and $\Psi(x) : \bigtimes_{\alpha \in T} \mathbb{R}^{K^\alpha} \to \mathbb{R}$ is a multilinear map depending on the chosen tensor format (see Section 1.4). This yields the optimization problem

$$\min_{(C^\alpha)_{\alpha \in T}} \frac{1}{n} \sum_{i=1}^n \gamma(\Psi(\cdot)((C^\alpha)_{\alpha \in T}), z_i) \tag{2.3}$$

over the set of parameters.

For the solution of the optimization problem (2.3), we use an alternating minimization algorithm which consists in successively solving

$$\min_{C^\alpha \in \mathbb{R}^{K^\alpha}} \frac{1}{n} \sum_{i=1}^{n} \gamma(\Psi(\cdot)((C^\beta)_{\beta \in T}), z_i)$$

for fixed parameters C^β, $\beta \neq \alpha$. Letting Ψ^α be the (linear) partial map defined by (1.4) yields the optimization problem

$$\min_{C^\alpha \in \mathbb{R}^{K^\alpha}} \frac{1}{n} \sum_{i=1}^{n} \gamma(\Psi^\alpha(\cdot)(C^\alpha), z_i). \tag{2.4}$$

The problem is then reduced to a succession of learning problems with linear models. For standard contrast functions, a closed form solution of (2.4) often exists: for instance, with the contrast function (2.2), the solution writes, using the orthonormality of the representation (see Section 1.5):

$$C^\alpha = \frac{1}{n} \sum_{i=1}^{n} \mathbf{\Psi}^\alpha(x_i),$$

with $\mathbf{\Psi}^\alpha$ defined by (1.5). The learning algorithm is described in Algorithm 6, where Step 3 consists in solving (2.4).

Algorithm 6 Learning algorithm for an approximation in a given subset of low-rank tensors.

Inputs: sample $S = \{z_i\}_{i=1}^n$, contrast function γ, tensor format with a multilinear parametrization $\Psi(\cdot)((C^\alpha)_{\alpha \in T})$ and initial values for the parameters $\{C^\alpha\}_{\alpha \in T}$

Outputs: approximation $g(\cdot) = \Psi(\cdot)((C^\alpha)_{\alpha \in T})$

1: **while** not converged **do**
2: **for** $\alpha \in T$ **do**
3: estimate C^α for fixed parameters C^β, $\beta \neq \alpha$ (learning problem with a linear model)
4: **end for**
5: **end while**

Regularization (or penalization). The optimization problem (2.3) may be replaced by

$$\min_{(C^\alpha)_{\alpha \in T}} \frac{1}{n} \sum_{i=1}^{n} \gamma(\Psi(\cdot)((C^\alpha)_{\alpha \in T}), z_i) + \sum_{\alpha \in T} \lambda_\alpha \Omega_\alpha(C^\alpha), \tag{2.5}$$

where $\lambda_\alpha \Omega_\alpha(C^\alpha)$ is a regularization (or penalization) term promoting some properties for the parameter C^α (*e.g.* sparsity or smoothness of functions associated with the parameter C^α). Regularization may be required for stability when only a few training samples are available or for exploiting a prior information on the parameters (with a bayesian point of view).

Example 2.3.1. *Usual regularization functions $\Omega_\alpha(C^\alpha)$ for promoting sparsity include the ℓ^0-norm $\|C^\alpha\|_0$, which is the number of non-zero coefficients in C^α, or its convex regularization provided by the ℓ^1-norm $\|C^\alpha\|_1$ (see [10]).*

When using an alternating minimization algorithm to solve (2.5) (Algorithm 6), the problem is reduced to the solution of successive learning problems with linear models (Step 3 of the algorithm)

$$\min_{C^\alpha \in \mathbb{R}^{K^\alpha}} \frac{1}{n} \sum_{i=1}^{n} \gamma(\Psi^\alpha(\cdot)(C^\alpha), z_i) + \lambda_\alpha \Omega_\alpha(C^\alpha), \tag{2.6}$$

for which efficient algorithms are usually available (for standard contrast functions and regularization functionals). Also, standard statistical methods such as cross-validation (see Section 2.4) can be used for the selection of the regularization parameter λ_α and of the regularization functional Ω_α (possibly depending on other parameters).

2.4 Validation and model selection

We first recall the principle of validation methods for the estimation of the risk $\mathcal{R}(g_S)$ for a function g_S estimated from the sample $S = \{z_i\}_{i=1}^n$ (see *e.g.* [38]). If V is a sample independent of S, then

$$\mathcal{R}_V(g_S) = \frac{1}{\#V} \sum_{z \in V} \gamma(g_S, z)$$

provides an unbiased estimator of the risk $\mathcal{R}(g_S)$. If S is the only available information, the risk can be estimated by the hold-out estimator $\mathcal{R}_V(g_{S \setminus V})$, where $V \subset S$ is a validation sample contained in the sample S, and $g_{S \setminus V}$ is the model estimated from the sample $S \setminus V$. Also, by introducing a partition of S into L validation samples $V_1, \ldots, V_L$, we define the cross-validation estimator of the risk $\mathcal{R}(g_S)$ by

$$\mathcal{R}^{CV}(g_S) = \frac{1}{L} \sum_{i=1}^{L} \mathcal{R}_{V_i}(g_{S \setminus V_i}).$$

The case where $L = n$, with $V_i = \{z_i\}$, corresponds to the leave-one-out estimator

$$\mathcal{R}^{\text{loo}}(g_S) = \frac{1}{n} \sum_{i=1}^{n} \gamma(g_{-i}, z_i),$$

with $g_{-i} = g_{S \setminus \{z_i\}}$. In some cases, cross-validation estimators can be computed efficiently, without computing the models $g_{S \setminus V_i}$ for all i.

Example 2.4.1 (Leave-one-out estimator for ordinary least-squares regression)**.** *Consider the case of least-squares regression, with the contrast function (2.1), and consider a linear model $g_S(x) = \boldsymbol{\Psi}(x)^T \boldsymbol{a}_S$, with a given $\boldsymbol{\Psi}(x) \in \mathbb{R}^m$, and coefficients $\boldsymbol{a}_S \in \mathbb{R}^m$ obtained by ordinary least-squares regression, i.e. by minimizing the empirical risk $\mathcal{R}_S(\boldsymbol{\Psi}(\cdot)^T \boldsymbol{a}) = \frac{1}{n}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{a}\|_2^2$ over $\boldsymbol{a} \in \mathbb{R}^m$, where $\boldsymbol{y} = (y_i)_{i=1}^n \in \mathbb{R}^n$ and $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ is the matrix whose i-th*

row is $\boldsymbol{\Psi}(x_i)^T$. *In this case, the leave-one-out estimator of the risk is*

$$\mathcal{R}^{loo}(g_S) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - g_S(x_i)}{1-h_i}\right)^2,$$

where h_i *is the diagonal term* $\boldsymbol{H}_{ii}$ *of the matrix* $\boldsymbol{H} = \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$. *This estimator only depends on* g_S *and not on the functions* g_{-i}, $1 \leq i \leq n$.

Remark 2.4.2 (Corrected estimators for ordinary least-squares regression)**.** *When the sample size* n *is small compared to the number of parameters* m, *several corrected estimators have been proposed. For the case of ordinary least-squares regression with linear models described in Example 2.4.1, a corrected estimator has been proposed by [39] in the form*

$$\widetilde{\mathcal{R}}^{loo}(g_S) = \mathcal{R}^{loo}(g_S)\left(1-\frac{m}{n}\right)^{-1}\left(1+\frac{1}{n}trace(\boldsymbol{G}^{-1}\bar{\boldsymbol{G}})\right),$$

where $\bar{\boldsymbol{G}}$ *is the Gram matrix* $\mathbb{E}(\boldsymbol{\Psi}(X)\boldsymbol{\Psi}(X)^T)$ *of* $\boldsymbol{\Psi}(X)$ *and* $\boldsymbol{G} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\Psi}(x_i)\boldsymbol{\Psi}(x_i)^T = \frac{1}{n}\boldsymbol{A}^T\boldsymbol{A}$ *is the empirical Gram matrix. When* $n \to \infty$, $\boldsymbol{G}$ *converges (almost surely) to* $\bar{\boldsymbol{G}}$, *so that* $trace(\boldsymbol{G}^{-1}\bar{\boldsymbol{G}})$ *converges to* m *and the correction factor converges to* 1.

Example 2.4.3 (Leave-one-out estimator for L^2 density estimation)**.** *Consider the contrast function (2.2) and a linear model* $g_S(x) = \boldsymbol{\Psi}(x)^T\boldsymbol{a}_S$, *with a given orthonormal basis* $\boldsymbol{\Psi}(x) \in \mathbb{R}^m$, *and coefficients* $\boldsymbol{a}_S \in \mathbb{R}^m$ *obtained by risk minimization. The leave-one-out estimator of the risk can be expressed as*

$$\begin{aligned}\mathcal{R}^{loo}(g_S) &= \frac{-n^2}{(n-1)^2}\|\boldsymbol{a}_S\|^2 + \frac{2n-1}{n(n-1)^2}\sum_{i=1}^{n}\sum_{k\in K_\lambda}\boldsymbol{\Psi}_k(x_i)^2 \\ &= \frac{1}{n(n-1)}\sum_{k\in K_\lambda}^{m}\left[\sum_{i=1}^{n}\boldsymbol{\Psi}_i(x_i)^2 - \frac{n}{n-1}\sum_{i\neq k}\boldsymbol{\Psi}_i(x_i)\boldsymbol{\Psi}_i(x_k)\right],\end{aligned}$$

with K_λ *the sparsity pattern such that* $(\boldsymbol{a}_S)_k = 0$ *for all* $k \notin K_\lambda$ *(see Sections 1.7 and 2.5). This is a special case for* $p = 1$ *of the result of [40, Prop. 2.1] for the leave-p-out estimator.*

Cross-validation estimators can be used for model selection. Suppose that different functions g_S^λ, $\lambda \in \Lambda$, have been estimated from the sample S (*e.g.* low-rank models with different bases, different ranks or different trees, see Section 2.6). Denoting by $\mathcal{R}^{CV}(g_S^\lambda)$ the estimator of the risk for the model g_S^λ, the optimal model with respect to this estimation of the risk is $g_S^{\lambda_S}$ with λ_S such that

$$\mathcal{R}^{CV}(g_S^{\lambda_S}) = \min_{\lambda\in\Lambda}\mathcal{R}^{CV}(g_S^\lambda).$$

2.5 Sparsity exploitation

We here present possible strategies for exploiting sparsity in the parameters C^α of a function in tree-based format, where $C^\alpha = (C^\alpha_k)_{k\in K^\alpha} \in \mathbb{R}^{K^\alpha}$, with K^α a finite set of indices. We define the support, or pattern, of C^α as $\text{support}(C^\alpha) = \{k \in K^\alpha : C^\alpha_k \neq 0\}$. We will first

describe how to select a pattern among a set of candidate patterns for C^α. Then, we will describe how to determine a set of candidate patterns.

2.5.1 Selection of a pattern from a set of candidate patterns

Let us suppose that we have a collection of candidate patterns K^α_λ, $\lambda \in \Lambda$, for the parameter C^α. At step α of the alternating minimization procedure for the empirical risk minimization (Step 3 of Algorithm 6), instead of solving (2.4), we compute for all $\lambda \in \Lambda$ the solution $C^{\alpha,\lambda}$ of

$$\min_{C^\alpha \in \mathbb{R}^{K^\alpha}} \frac{1}{n} \sum_{i=1}^{n} \gamma(\Psi^\alpha(\cdot)(C^\alpha), z_i) \quad \text{subject to } \operatorname{support}(C^\alpha) \subset K^\alpha_\lambda, \tag{2.7}$$

which provides a collection of approximations $g^\lambda(\cdot) = \Psi^\alpha(\cdot)(C^{\alpha,\lambda})$, $\lambda \in \Lambda$. Then, cross-validation methods can be used in order to select a particular solution $C^\alpha = C^{\alpha,\hat{\lambda}}$, with $\hat{\lambda}$ minimizing over $\lambda \in \Lambda$ a certain cross-validation estimator of the risk $\mathcal{R}(g^\lambda)$ (see Section 2.4). Usually, the pattern of $C^{\alpha,\lambda}$ will coincide with K^α_λ but it may be strictly contained in K^α_λ.

Remark 2.5.1. *An equivalent form of the problem* (2.7) *is given by*

$$\min_{C^\alpha} \frac{1}{n} \sum_{i=1}^{n} \gamma(\Psi^\alpha(\cdot)(C^\alpha), z_i) + \Omega^\lambda_\alpha(C^\alpha),$$

where Ω^λ_α *is the characteristic function of the subset of elements* C^α *whose support is contained in* K^α_λ*, i.e.* $\Omega^\lambda_\alpha(C^\alpha) = 0$ *if* $\operatorname{support}(C^\alpha) \subset K^\alpha_\lambda$ *and* $\Omega^\lambda_\alpha(C^\alpha) = +\infty$ *if* $\operatorname{support}(C^\alpha) \not\subset K^\alpha_\lambda$*. This formulation can be seen as a regularized version of the empirical risk minimization problem, where* λ *plays the role of the regularization parameter which can be estimated using cross-validation methods.*

2.5.2 Determination of the set of candidate patterns

Let us now discuss how to propose the set of candidate patterns K^α_λ, $\lambda \in \Lambda$. In the case where C^α collects the coefficients of functions on a hierarchical (or multilevel) basis $\mathbf{\Phi}^\alpha(x_\alpha)$, the patterns can be determined by hand as a nested sequence of patterns $K^\alpha_0 \subset \cdots \subset K^\alpha_p = K^\alpha$, where K^α_λ is associated with a basis of level λ (*e.g.* for a polynomial approximation, λ may be the degree of the polynomial space); see Example 1.7.1. The set of candidate patterns can also be determined automatically by using a greedy algorithm, such as a matching pursuit algorithm [41], which provides a sequence of nested patterns $K^\alpha_0, \ldots, K^\alpha_{\#K^\alpha - 1}$. Another approach consists in solving (2.6) with a sparsity-inducing penalization for several values of $\lambda = \lambda_\alpha$, therefore leading to a collection of solutions g^λ. The solutions g^λ may be directly considered as the set of candidate approximations. However, in practice, we extract the patterns K^α_λ of the different approximations and re-estimate the coefficients by solving the

problem (2.7) without regularization. This usually provides estimates with better statistical properties and allows the use of fast procedures for the estimation of cross-validation estimators [42].

Example 2.5.2. *In supervised learning, when using a square loss and a ℓ^1-norm regularization, and assuming (up to a vectorization) that $C^\alpha \in \mathbb{R}^{m_\alpha}$, with $m_\alpha = \#K^\alpha$, (2.6) is a LASSO problem for which efficient algorithms are available, such as the LARS algorithm [43], which directly provides a set of solutions associated with different patterns (the so-called regularization path).*

2.6 Adaptive approximation in tree-based tensor format

In this section, we present adaptive learning algorithms for the approximation of functions in tree-based tensor format. First, we present an algorithm with tree-based rank adaptation for the approximation in tree-based tensor format with a given dimension tree. Then, we present a strategy combining the tree-based rank adaptation with dimension tree optimization (presented in Section 1.8).

2.6.1 Tree-based rank adaptation

Here, we propose a learning algorithm for the approximation in tree-based tensor format with rank adaptation. The tree T is supposed to be given. This algorithm provides a sequence of approximations $g^m \in \mathcal{T}^T_{r^m}(\mathcal{H})$ with increasing T-rank $r^m = (r^m_\alpha)_{\alpha \in T}$ (relatively to the partial ordering on $\mathbb{N}^d$). The sequence is defined as follows. We start by computing an approximation g^1 with T-rank $r^1 = (1, \ldots, 1)$ using Algorithm 6. At iteration m, given an approximation g^m with T-rank r^m, we first define the T-rank r^{m+1} of the next iterate by

$$r^{m+1}_\alpha = \begin{cases} r^m_\alpha + 1 & \text{if } \alpha \in T^\theta_m, \\ r^m_\alpha & \text{if } \alpha \notin T^\theta_m, \end{cases} \tag{2.8}$$

where T^θ_m is a suitably chosen subset of nodes (see below), and we use Algorithm 6 to obtain an approximation g^{m+1} in the set $\mathcal{T}^T_{r^{m+1}}(\mathcal{H})$ of tensors with fixed tree-based rank r^{m+1}. Algorithm 7 presents this learning algorithm in a given tree-based tensor format with rank adaptation, which returns an element of the generated sequence of approximations selected using an estimation of the generalization error.

The crucial ingredient of Algorithm 7 is the selection of the subset of nodes T^θ_m at Step 3. A natural idea is to select the nodes $\alpha \in T$ associated with the highest truncation errors

$$\min_{\mathrm{rank}_\alpha(g) \leq r^m_\alpha} \mathcal{R}(g) - \mathcal{R}(f) \coloneqq \eta_\alpha(f, r^m_\alpha)^2,$$

where f is the oracle function. When $\mathcal{R}(g) - \mathcal{R}(f)$ is the square of the distance in L^2_μ-norm between g and f (for least-squares regression or density estimation in L^2_μ), then the

Algorithm 7 Learning algorithm in a given tree-based tensor format with rank adaptation.

Inputs: sample $S = \{z_i\}_{i=1}^n$, contrast function, dimension tree T and maximal number of iterations M

Outputs: approximation g in $\mathcal{T}_r^T(\mathcal{H})$ with T-rank r

1: compute an approximation g^1 with T-rank $r^1 = (1, \ldots, 1)$ using Algorithm 6
2: **for** $m = 1, \ldots, M-1$ **do**
3: compute T_m^θ with Algorithm 8 and r^{m+1} defined by (2.8)
4: compute an approximation g^{m+1} in $\mathcal{T}_{r^{m+1}}^T(\mathcal{H})$ using Algorithm 6
5: **end for**
6: select $m^\star = \arg\min_{1\leq m\leq M} \mathcal{R}_V(g^m)$, where V is a validation set independent of S, and return $g = g^{m^\star}$, with $r = r^{m^\star}$

truncation errors become

$$\eta_\alpha(f, r_\alpha^m)^2 = \min_{\mathrm{rank}_\alpha(g)\leq r_\alpha^m} \|g - f\|^2 = \sum_{k>r_\alpha^m} (\sigma_k^\alpha(f))^2,$$

where $\sigma_k^\alpha(f)$ are the α-singular values of f. In practice, these truncation errors are estimated by $\eta_\alpha^2(\tilde{g}; r_\alpha^m)$ where $\tilde{g}$ is an approximation obtained by a correction of g^m. Algorithm 8 presents the procedure of selection of T_m^θ, and proceeds as follows. We first compute an approximation $\tilde{g}$ with a rank r such that $r_\alpha^m \leq r_\alpha \leq r_\alpha^m + 1$ for all $\alpha \in T$[1]. In practice, it is done by first computing a rank-one correction c of g^m, which yields an approximation $g^m + c$ with T-rank $r \geq r^m$, and then by using Algorithm 6, with $g^m + c$ as an initialization, to compute an approximation $\tilde{g}$ with rank r. Then, we compute the α-singular values $\Sigma^\alpha(\tilde{g}) = \{\sigma_i^\alpha\}_{i=1}^{r_\alpha}$ of $\tilde{g}$ for all $\alpha \in T \setminus \{D\}$ (see Section 1.6), and define a subset $\hat{T}$ of candidate nodes for increasing the ranks as follows:

$$\hat{T} = T \setminus (\{D\} \cup \{\alpha \in \mathcal{L}(T) : r_\alpha = \#I^\alpha\} \cup \{\alpha \in T \setminus \{D\} : \eta_\alpha(\tilde{g}; r_\alpha^m) \leq \varepsilon \|\tilde{g}\|\}), \tag{2.9}$$

which contains all the nodes of T except the root, the leaf nodes α for which $r_\alpha = \#I^\alpha = N_\alpha$ (necessary condition for the tree-based rank to be admissible, see [26, Section 2.3]) and the nodes for which $\eta_\alpha(\tilde{g}; r_\alpha^m)$ is smaller than a constant ε (typically machine precision) multiplied by $\|\tilde{g}\|$. We then define

$$T_m^\theta = \{\alpha \in \hat{T} : \eta_\alpha(\tilde{g}; r_\alpha^m) \geq \theta \max_{\beta\in\hat{T}} \eta_\beta(\tilde{g}; r_\beta^m)\}, \tag{2.10}$$

where $\theta \in [0, 1]$ is a parameter controlling the number of nodes to be selected.

Note that since $r_\alpha = r_\alpha^m + 1$ for all $\alpha \in \hat{T}$, $\eta_\alpha(\tilde{g}; r_\alpha^m) = \sigma_{r_\alpha}^\alpha$, the minimal α-singular value of $\tilde{g}$. When $\theta = 0$, $T_m^\theta = \hat{T}$, which means that the α-rank is increased for every $\alpha \in \hat{T}$, so possibly for every $\alpha \in T \setminus \{D\}$, which is not a desired adaptive strategy. When $\theta = 1$,

[1] An approximation $\tilde{g}$ with higher rank could be computed in order to improve the estimation of the truncation errors.

we increase only the ranks associated with nodes α such that $\sigma_{r_\alpha}^\alpha = \max_{\beta \in \hat{T}} \sigma_{r_\beta}^\beta$ (usually a single node). This choice results in a slow increase of the ranks and possibly to non admissible ranks. To ensure that r^{m+1} satisfies the necessary conditions of admissibility, we select θ automatically as the highest value in $[0, \theta^\star]$ such that r^{m+1} is admissible, where $\theta^\star \in [0,1]$ is a user-defined parameter selected in order to increase sufficiently many ranks at each iteration. In the following numerical experiments, $\theta^\star$ is chosen equal to 0.8.

Algorithm 8 Computation of the subset of nodes T_m^θ whose rank is increased.

Inputs: sample $S = \{z_i\}_{i=1}^n$, approximation $g^m \in \mathcal{T}_{r^m}^T(\mathcal{H})$, parameter $\theta^\star$

Outputs: subset T_m^θ

1: use Algorithm 6 to compute an approximation $\tilde{g}$ with ranks $r_\alpha \in \{r_\alpha^m, r_\alpha^m + 1\}$, $\alpha \in T$
2: compute the set of α-singular values $\Sigma^\alpha(\tilde{g}) = \{\sigma_i^\alpha\}_{i=1}^{r_\alpha}$ of $\tilde{g}$ for all $\alpha \in T$
3: compute the subset of candidate nodes $\hat{T}$ defined by (2.9)
4: compute the subset T_m^θ defined by (2.10) and the corresponding rank r^{m+1} defined by (2.8), with θ the highest value in $[0, \theta^\star]$ such that r^{m+1} satisfies the necessary conditions of admissibility [26, Section 2.3]

Remark 2.6.1. *A correction c in tree-based tensor format of $g^m \in \mathcal{T}_{r^m}^T(\mathcal{H})$ is obtained by solving the problem*

$$\min_{c \in \mathcal{T}_{\tilde{r}}^T(\mathcal{H})} \mathcal{R}_n(g^m + c).$$

In supervised learning, Algorithm 6 can be directly used to compute c, replacing y_k by $y_k - g^m(x_k)$, $k = 1, \ldots, n$, in the training sample S.

However, this not the case when performing density estimation. The empirical risk minimization problem writes

$$\min_{c \in \mathcal{T}_{\tilde{r}}^T(\mathcal{H})} \|c\|^2 - \frac{2}{n} \sum_{i=1}^n c(x_i) + 2 \int_{\mathcal{X}} c(x) g^m(x) \mathrm{d}\mu(x).$$

It is solved using an alternating minimization algorithm which consists in minimizing alternatively on each parameter $\widetilde{C}^\alpha$, $\alpha \in T$. For a given α, using the representation $c(x) = \langle \mathbf{\Psi}^\alpha(x), \widetilde{C}^\alpha \rangle_\alpha$ with orthonormal functions $\{\mathbf{\Psi}^\alpha(x)\}$ depending on the fixed parameters $\widetilde{C}^\beta$, $\beta \in T \setminus \{\alpha\}$, the minimization problem writes

$$\min_{\widetilde{C}^\alpha \in \mathbb{R}^{\tilde{K}^\alpha}} \|\widetilde{C}^\alpha\|_\alpha^2 - \frac{2}{n} \sum_{i=1}^n \langle \mathbf{\Psi}^\alpha(x_i), \widetilde{C}^\alpha \rangle_\alpha + 2 \int_{\mathcal{X}} \langle \mathbf{\Psi}^\alpha(x), \widetilde{C}^\alpha \rangle_\alpha g^m(x) \mathrm{d}\mu(x). \tag{2.11}$$

The solution of (2.11) is

$$\widetilde{C}^\alpha = \frac{1}{n} \sum_{i=1}^n \mathbf{\Psi}^\alpha(x_i) - S^\alpha,$$

with S^α such that

$$\langle S^\alpha, \widetilde{C}^\alpha \rangle_\alpha = \int_{\mathcal{X}} \langle \mathbf{\Psi}^\alpha(x), \widetilde{C}^\alpha \rangle_\alpha g^m(x) \mathrm{d}\mu(x),$$

that is

$$S^\alpha = \sum_{\substack{i_\nu \in I^\nu \\ \nu \in \mathcal{L}(T)}} \sum_{\substack{1 \leq k_\delta \leq \tilde{r}_\delta^m \\ \delta \notin S(\alpha) \cup \{\alpha\}}} \sum_{\substack{1 \leq l_\gamma \leq r_\gamma^m \\ \gamma \in T}} \prod_{\beta \notin \mathcal{L}(T) \cup \{\alpha\}} \widetilde{C}^\beta_{(k_\nu)_{\nu \in S(\beta)}, k_\beta} \prod_{\beta \notin \mathcal{L}(T)} C^\beta_{(l_\nu)_{\nu \in S(\beta)}, l_\beta} \prod_{\beta \in \mathcal{L}(T)} \widetilde{C}^\beta_{i_\beta, k_\beta} C^\beta_{i_\beta, l_\beta}$$

if $\alpha \in T \setminus \mathcal{L}(T)$, *and*

$$S^\alpha = \sum_{\substack{i_\nu \in I^\nu \\ \nu \in \mathcal{L}(T) \setminus \{\alpha\}}} \sum_{\substack{1 \leq k_\delta \leq \tilde{r}_\delta^m \\ \delta \in T \setminus \{\alpha\}}} \sum_{\substack{1 \leq l_\gamma \leq r_\gamma^m \\ \gamma \in T}} \prod_{\beta \notin \mathcal{L}(T)} \widetilde{C}^\beta_{(k_\nu)_{\nu \in S(\beta)}, k_\beta} C^\beta_{(l_\nu)_{\nu \in S(\beta)}, l_\beta} \prod_{\beta \in \mathcal{L}(T) \setminus \{\alpha\}} \widetilde{C}^\beta_{i_\beta, k_\beta} C^\beta_{i_\beta, l_\beta} C^\alpha_{i_\alpha, l_\alpha}$$

if $\alpha \in \mathcal{L}(T)$. *In the above expressions, a summation over* $\beta \notin J$ *means over* $\beta \in T \setminus J$. *In the numerical experiments, we will only consider a rank-one correction c, which means* $\tilde{r}_\alpha = 1$ *for all* α *and* $\widetilde{C}^\beta = 1$ *for all* $\beta \notin \mathcal{L}(T)$.

Remark 2.6.2. *In practice, we observe that it is important to provide a good initialization for the learning algorithm 6 used at Step 4 of Algorithm 7 for the computation of* g^{m+1}. *We take as an initialization the truncation* $\mathrm{Truncate}(\tilde{g}; T, r^{m+1})$ *at rank* r^{m+1} *of the approximation* $\tilde{g}$ *computed at Step 1 of Algorithm 8.*

2.6.2 Learning scheme with tree-based rank and dimension tree adaptation

Algorithm 9 describes a global algorithm for the approximation in tree-based tensor format. It is similar to Algorithm 7, with an additional step of tree adaptation using Algorithm 5 of Chapter 1. The algorithm stops when reaching a maximum number of iterations M, or when at least one of the following stopping criteria is met:

$$\begin{aligned} &\mathcal{R}_V(g^m) \leq \varepsilon_{\text{goal}} \mathcal{R}_V(0), \\ &\mathcal{R}_V(g^m) \geq \tau_{\text{overfit}} \min_{1 \leq i \leq m-1} \mathcal{R}_V(g^i) \end{aligned}$$

where V is a validation set independent of S. For the recovery of functions in the noiseless case, $\varepsilon_{\text{goal}}$ is taken of the order of the machine precision. The second criterion is met once overfitting occurs.

Remark 2.6.3 (Choice of τ_{overfit})**.** *The coefficient* τ_{overfit} *should be taken greater or equal than 1 if the empirical risk* $\min_{1 \leq i \leq m-1} \mathcal{R}_V(g^i)$ *is positive, and less or equal than 1 if it is negative. Indeed, we want to ensure that the stopping criterion is such that*

$$\mathcal{R}_V(g^m) \geq \tau_{\text{overfit}} \min_{1 \leq i \leq m-1} \mathcal{R}_V(g^i) \geq \min_{1 \leq i \leq m-1} \mathcal{R}_V(g^i),$$

with the parameter τ_{overfit} *specifying how much the risk* $\mathcal{R}_V(g^m)$ *has to be greater than* $\min_{1 \leq i \leq m-1} \mathcal{R}_V(g^i)$.

In the following experiments, we choose $\tau_{\text{overfit}} = 10^2$ for supervised learning (where the empirical risk is positive), and $\tau_{\text{overfit}} = 10^{-2}$ for density estimation (where the empirical risk can take negative values).

Algorithm 9 Learning scheme with rank and tree adaptation.

Inputs: sample $\{z_i\}_{i=1}^n$, contrast function, initial tree T

Outputs: new tree T and approximation g in $\mathcal{T}_r^T(\mathcal{H})$ with T-rank r

compute an approximation g^1 with T-rank $r^1 = (1, \ldots, 1)$ using Algorithm 6
$m \leftarrow 1$
while the stopping criteria are not met **do**
  compute T_m^θ with Algorithm 8 and corresponding rank r^{m+1} defined by (2.8)
  compute an approximation g^{m+1} in $\mathcal{T}_{r^{m+1}}^T(\mathcal{H})$ using Algorithm 6
  using Algorithm 5, search for a new tree $\widetilde{T}$ to obtain an approximation $\tilde{g}$ in $\mathcal{T}_{\tilde{r}^{m+1}}^{\widetilde{T}}(\mathcal{H})$ of g^{m+1} with reduced storage complexity
  if $C(\widetilde{T}, \tilde{r}^{m+1}) < C(T, r^{m+1})$ **then**
    $g^{m+1} \leftarrow \tilde{g}$
    compute an approximation $\hat{g}$ in $\mathcal{T}_{\tilde{r}^{m+1}}^{\widetilde{T}}(\mathcal{H})$ of g using Algorithm 6 with initialization g^{m+1}
    $T \leftarrow \widetilde{T}$, $g^{m+2} \leftarrow \hat{g}$, $r^{m+2} \leftarrow \tilde{r}^{m+1}$
    $m \leftarrow m + 1$
  end if
  $m \leftarrow m + 1$
end while
select $m_{opt} = \arg\min_{1 \leq j \leq m} \mathcal{R}_V(g^j)$, where V is a validation set independent of S, and return $g = g^{m_{opt}}$

2.7 Numerical experiments

This section presents numerical experiments in a statistical learning setting where, for given samples $S = \{(z_i)\}_{i=1}^n$ of Z, we try to find an approximation g of a function f using tree-based tensor formats. Two frameworks are considered: least-squares regression, where $Z = (X, Y)$, and L^2 density estimation, where $Z = X$.

We use Algorithm 9 for the approximation in tree-based tensor format $\mathcal{T}_r^T(\mathcal{H})$ with adaptation of both the T-rank r and the dimension tree T over $D = \{1, \ldots, d\}$. We exploit sparsity for the representation of one-dimensional functions in the bases $\{\phi_{i_\nu}^\nu\}_{i_\nu \in I^\nu}$, $\nu \in D$, or equivalently, in the tensors $C^\alpha \in \mathbb{R}^{I_\alpha \times \{1,\ldots,r_\alpha\}}$, $\alpha \in \mathcal{L}(T)$. For the estimation of sparse approximations, we use the strategy described in Section 2.5, which consists in estimating a succession of approximations with nested sparsity patterns, and then in selecting the optimal approximation based on an estimation of the risk using (possibly corrected) leave-one out estimators (see Example 2.4.1, Remarks 2.4.2 and 2.4.3).

2.7.1 Supervised learning

In this section, we illustrate the performance of the proposed algorithms in a supervised learning framework.

Assumption on Z and contrast function. We assume that $Y = f(X) + \varepsilon$, where $X = (X_1, \ldots, X_d)$ has a known probability law μ over $\mathcal{X} = \mathbb{R}^d$, where $f \in L^2_\mu(\mathcal{X})$, and where the noise ε is a random variable, independent of X, with finite variance. In all examples, we consider a least-squares regression setting by choosing a contrast function $\gamma(g, (x, y)) = (g(x) - y)^2$.

Approximation spaces. For the spaces $\mathcal{H}_\nu$ ($\nu \in D$), we choose polynomial spaces $\mathbb{P}_p(\mathcal{X}^\nu)$ of degree p and we use orthonormal polynomial bases $\{\phi^\nu_{i_\nu}\}_{i_\nu \in I^\nu}$ in $L^2_{\mu_\nu}(\mathcal{X}_\mu)$, with $I^\nu = \{0, \ldots, p\}$. We exploit sparsity in the leaf tensors $(C^\alpha)_{\alpha \in \mathcal{L}(T)}$ by using a working set strategy. We use the natural sequence of candidate patterns associated with spaces of polynomials with increasing degree.

Tensor formats. When using Algorithm 9 with tree-based rank and dimension tree adaptation, the starting dimension tree is drawn randomly in the set of trees with a given arity, chosen equal to 2 (binary trees). We set $N = 100$ in Algorithm 5, and the leave-one out estimator of the risk, divided by the empirical second moment of Y (to define a relative error in the noiseless case), is used as the tolerance ϵ. We also let the maximum number of iterations in Algorithm 9 be sufficiently high.

Error measures. We estimate the generalization (L^2) error using a test sample S_{test} of size 10000, independent of S. The relative test error $\varepsilon(S_{\text{test}}, g)$ associated with the function g and test sample S_{test} is defined as

$$\varepsilon(S_{\text{test}}, g)^2 = \frac{\sum_{(x,y) \in S_{\text{test}}} (y - g(x))^2}{\sum_{(x,y) \in S_{\text{test}}} y^2}.$$

Robustness study. For studying the robustness of the proposed algorithms, we run it 10 times for each example, each run using a different training sample S and a different test sample S_{test}, and starting from a tree drawn randomly in the set of trees with a given arity. This allows us to provide ranges for the obtained quantities of interest (errors and complexities). We also present an estimation $\hat{\mathbb{P}}(T \text{ is optimal})$ of the probability of obtaining an approximation in $\mathcal{T}^T_r(\mathcal{H})$ with a dimension tree T optimal in a sense specified in each

example, obtained by counting, out of the 10 runs, how many times the algorithm returned an optimal tree.[2]

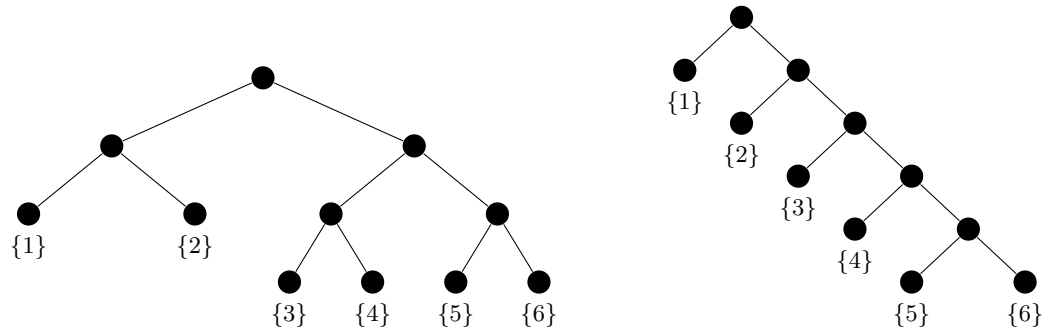
2.7.1.1 Anisotropic multivariate function

We consider the following function in dimension $d = 6$:

$$f(X) = \frac{1}{(10 + 2X_1 + X_3 + 2X_4 - X_5)^2} \tag{I.i}$$

where the random variables $X = (X_1, \ldots, X_6)$ are uniform on $[-1, 1]$, and we consider a noise $\varepsilon = 0$. We choose polynomial spaces $\mathcal{H}_\nu = \mathbb{P}_{10}(\mathcal{X}_\nu)$. We use Algorithm 9 to obtain an approximation of Function $(I.i)$ in $\mathcal{T}_r^T(\mathcal{H})$, with adapted tree T and tree-based rank r.

The tree T is selected in the family of trees of arity 2, starting from two different families of trees $T_\sigma^1 = \{\sigma(\alpha) : \alpha \in T^1\}$ and $T_\sigma^2 = \{\sigma(\alpha) : \alpha \in T^2\}$, with σ a permutation of D and where the trees T^1 and T^2 are represented in Figures 2.1a and 2.1b respectively.



(a) Tree T^1. (b) Tree T^2.

Figure 2.1: Two particular dimension trees over $D = \{1, \ldots, 6\}$ yielding two different families of trees obtained by permutations.

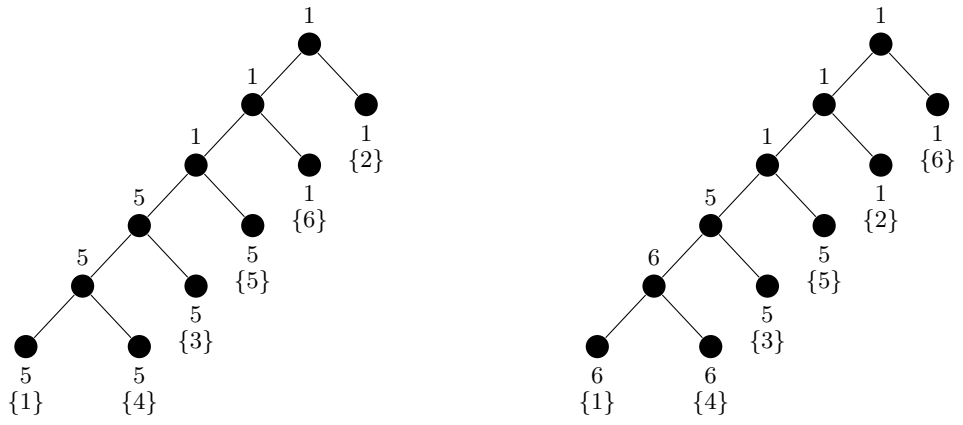
Table 2.1 summarizes the results. We observe that we obtain with high probability a very accurate approximation with only a small training sample, and a fast decrease of the error with the training sample size n. We also notice that with very high probability, the algorithm finds a tree T containing the node $\{1, 3, 4, 5\}$ (associated with the only variables on which Y depends) and almost only increases the ranks associated with the nodes involving the dimensions 1, 3, 4 and 5. We also notice that, when the training sample size is large enough, the cross-validation error is a good estimator of the generalization error, enabling model selection in Step 6 of Algorithm 7 and Step 15 of Algorithm 9, without the need of an independent test sample.

[2] Knowing that the number of times the algorithm returns an optimal tree out of 10 trials follows a binomial distribution of parameters a probability p and the number of trials 10, we can propose a lower bound of p with confidence level $1 - \alpha$ by using the Clopper-Pearson confidence interval. For example, with a level $1 - \alpha = 0.95$, $\hat{\mathbb{P}}(T \text{ is optimal}) = 0.9$ leads to $\mathbb{P}(p \geq 0.55) = 0.95$, and $\hat{\mathbb{P}}(T \text{ is optimal}) = 1$ leads to $\mathbb{P}(p \geq 0.69) = 0.95$.

T_σ	n	$\hat{\mathbb{P}}(\{1,3,4,5\} \in T)$	$\varepsilon(S_{\text{test}}, g)$	CV error	$C(T, r)$
T_σ^1	10^2	90%	$[2.32 \cdot 10^{-3}, 8.36 \cdot 10^{-3}]$	$[1.08 \cdot 10^{-4}, 1.34 \cdot 10^{-3}]$	$[132, 255]$
	10^3	100%	$[6.93 \cdot 10^{-6}, 4.21 \cdot 10^{-5}]$	$[3.11 \cdot 10^{-8}, 8.49 \cdot 10^{-7}]$	$[407, 1338]$
	10^4	100%	$[3.13 \cdot 10^{-8}, 4.47 \cdot 10^{-6}]$	$[3.64 \cdot 10^{-9}, 1.64 \cdot 10^{-6}]$	$[376, 881]$
T_σ^2	10^2	90%	$[1.18 \cdot 10^{-3}, 8.23 \cdot 10^{-3}]$	$[3.67 \cdot 10^{-5}, 5.54 \cdot 10^{-4}]$	$[132, 266]$
	10^3	90%	$[2.23 \cdot 10^{-6}, 3.73 \cdot 10^{-5}]$	$[2.24 \cdot 10^{-8}, 8.74 \cdot 10^{-7}]$	$[374, 1182]$
	10^4	100%	$[2.06 \cdot 10^{-8}, 2.06 \cdot 10^{-6}]$	$[1.76 \cdot 10^{-9}, 1.82 \cdot 10^{-6}]$	$[344, 1403]$

Table 2.1: Results for Function $(I.i)$ starting from two families of trees T_σ^1 and T_σ^2: training sample size n, estimation of the probability of having $\{1,3,4,5\} \in T$, and ranges (over the 10 trials) for the test error, the cross-validation (CV) error estimator and the storage complexity.

Figure 2.2 presents examples of trees and associated tree-based ranks obtained by the algorithm, starting from a tree of one or the other family of trees. We observe that the algorithm modified the structure of the starting tree in order to reduce the storage complexity of the representation by isolating the dimensions 2 and 6 from the other dimensions. Furthermore, by studying Function $(I.i)$, we notice that the variables X_1 and X_4 have the same influence on the output of the function, which explains why they appear grouped in the shown trees. A similar conclusion can be drawn for the variables X_3 and X_5.



(a) Starting from T_σ^1.

(b) Starting from T_σ^2.

Figure 2.2: Examples of trees T obtained using Algorithm 9 on the same training sample of size 10^4, starting from a tree of the first family (a) or the second family (b). The obtained α-ranks are indicated at each node and the dimensions associated with the leaf nodes are displayed in brackets.

Illustration of the behavior of Algorithm 9. Table 2.2 illustrates the behavior of a single run of Algorithm 9 for the construction of an approximation of Function $(I.i)$ using a training sample of size $n = 10000$ and starting from the tree 2.3a. The trees returned by the algorithm at each iteration are displayed in Figure 2.3. We observe that the algorithm presents a fast convergence and yields a very accurate approximation after 21 iterations (and 9 adaptations of the tree).

m	Tree T	Tree-based rank r^m	$\varepsilon(S_{\text{test}}, g)$	$C(T, r^m)$
1	Fig. 2.3a	$(1,1,1,1,1,1,1,1,1,1,1)$	$4.88 \cdot 10^{-2}$	71
2	Fig. 2.3b	$(1,1,1,1,1,1,1,1,1,1,1)$	$4.88 \cdot 10^{-2}$	71
3		$(1,1,1,1,1,1,1,1,1,1,1)$	$4.88 \cdot 10^{-2}$	71
4	Fig. 2.3c	$(1,1,1,1,1,1,2,1,2,1,1)$	$3.81 \cdot 10^{-2}$	96
5		$(1,1,1,1,1,1,2,1,2,1,1)$	$3.81 \cdot 10^{-2}$	96
6	Fig. 2.3d	$(1,2,1,2,1,2,2,1,2,2,1)$	$1.95 \cdot 10^{-3}$	132
7		$(1,2,1,2,1,2,2,1,2,2,1)$	$1.95 \cdot 10^{-3}$	132
8	Fig. 2.3e	$(1,2,1,3,1,2,3,1,3,2,1)$	$8.50 \cdot 10^{-4}$	174
9		$(1,2,1,3,1,2,3,1,3,2,1)$	$8.49 \cdot 10^{-4}$	174
10		$(1,3,1,3,1,3,3,1,3,3,1)$	$6.53 \cdot 10^{-5}$	219
11	Fig. 2.3f	$(1,3,3,3,1,3,3,1,3,3,1)$	$5.88 \cdot 10^{-5}$	227
12		$(1,3,3,3,1,3,3,1,3,3,1)$	$6.08 \cdot 10^{-5}$	227
13	Fig. 2.3g	$(1,4,1,1,4,3,4,1,4,3,1)$	$1.86 \cdot 10^{-5}$	290
14		$(1,4,1,1,4,3,4,1,4,3,1)$	$1.86 \cdot 10^{-5}$	290
15		$(1,4,1,1,4,4,4,1,4,4,1)$	$2.05 \cdot 10^{-6}$	344
16	Fig. 2.3h	$(1,5,1,1,4,4,4,1,4,4,1)$	$1.56 \cdot 10^{-6}$	376
17		$(1,5,1,1,4,4,4,1,4,4,1)$	$1.54 \cdot 10^{-6}$	376
18	Fig. 2.3i	$(1,5,1,1,4,4,5,1,5,4,1)$	$5.03 \cdot 10^{-7}$	438
19		$(1,5,1,1,4,4,5,1,5,4,1)$	$4.59 \cdot 10^{-7}$	438
20		$(1,5,1,1,5,5,5,1,5,5,1)$	$6.71 \cdot 10^{-8}$	519
21	Fig. 2.3j	$(1,5,1,1,5,5,5,1,5,5,1)$	$4.88 \cdot 10^{-8}$	519

Table 2.2: Behavior of Algorithm 9 for the approximation of Function $(I.i)$, with $n = 10000$ and starting from the dimension tree T^1_σ of Figure 2.3a. The node numbers can be seen for each tree in Figure 2.3.

2.7.1.2 Sum of bivariate functions

We here consider a sum of bivariate functions

$$f(X) = h(X_1, X_2) + h(X_3, X_4) + \cdots + h(X_{d-1}, X_d), \tag{I.ii}$$

with $h(X_{\nu-1}, X_\nu) = \sum_{i=0}^{m} X^i_{\nu-1} X^i_\nu$ and where the random variables X_ν are independent and uniform on $[-1, 1]$. The problem addressed here is the recovery of the function using few samples and using Algorithm 9 with rank and tree adaptation. We choose $d = 10$ and $m = 3$ and we consider $\mathcal{H} = \bigotimes_{\nu=1}^{d} \mathbb{P}_5(\mathcal{X}^\nu)$, so that $f \in \mathcal{H}$ (no discretization errors). Figures 2.4a and 2.4b present the tree-based ranks and storage complexity for the exact representation of the function using two different dimension trees T^1 (balanced tree) and T^2 (linear tree). We observe that the function can be represented in both formats with a moderate storage complexity. When running the algorithm with tree adaptation, the tree T is selected in the family of trees of arity 2, starting from two different families of trees $T^1_\sigma = \{\sigma(\alpha) : \alpha \in T^1\}$ and $T^2_\sigma = \{\sigma(\alpha) : \alpha \in T^2\}$, with σ a permutation of D.

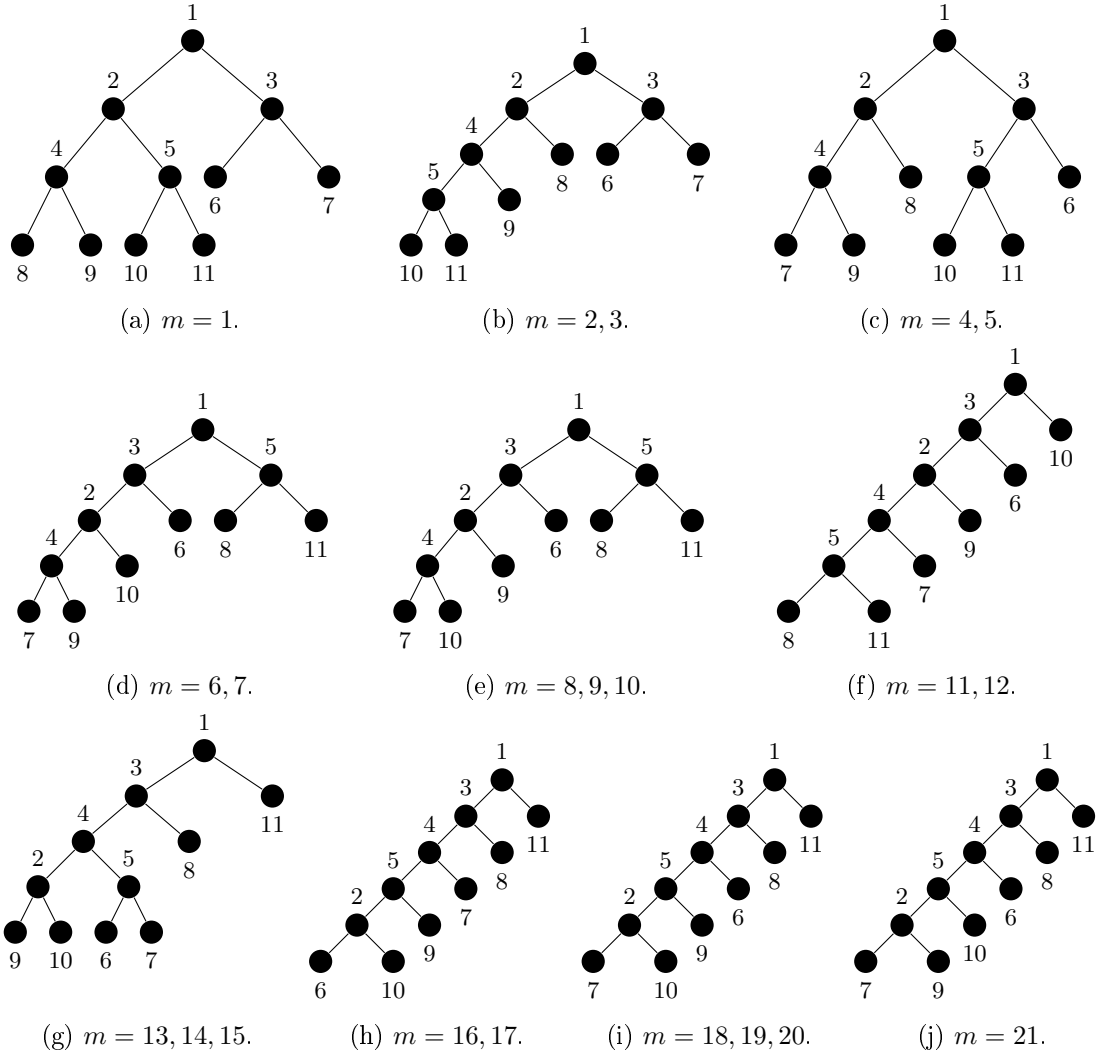
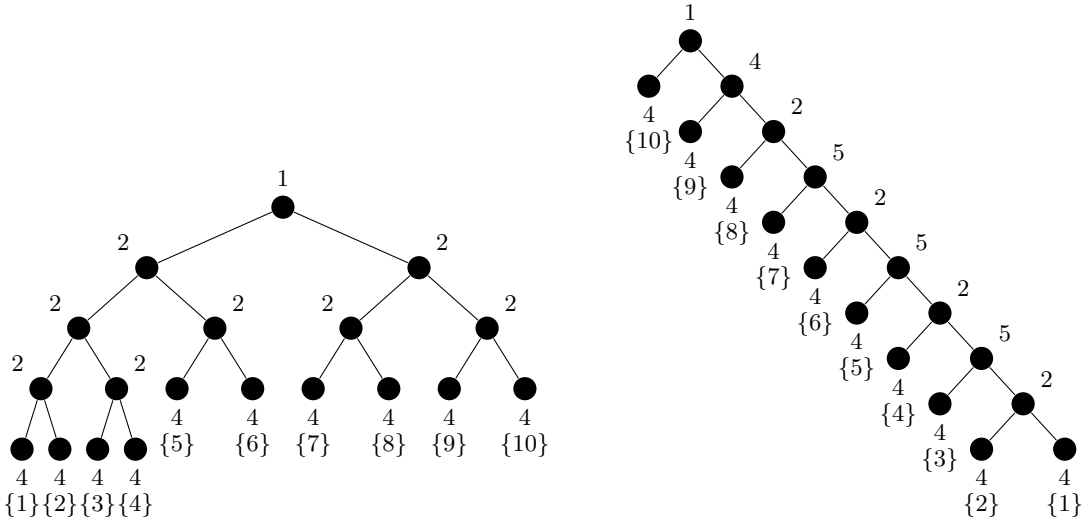


Figure 2.3: Dimension trees associated with the iteration number m in Table 2.2, with each node numbered. The singletons $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}$ correspond to the leaf nodes numbered $7, 11, 10, 9, 6, 8$ respectively.

Given the structure of the function $(I.ii)$, a tree T will be said to be optimal if $\{k, k+1\} \in T$ for all odd $k \in \{1, \ldots, d-1\}$. As we will see below, this optimal tree is recovered with high probability by the proposed algorithm when large enough training sets are used.

Illustration in the noiseless case. Table 2.3 summarizes the obtained results in the noiseless case $\varepsilon = 0$. We first observe that with a training sample large enough, with high probability, the algorithm is able to recover the function f at machine precision with an optimal tree, this probability being higher when we use the family of balanced trees rather than the family of linear trees. For $n = 10^4$, all the obtained approximations use an optimal tree, which can be shown to lead to the smallest storage complexity for an exact representation of the function f.

We also notice the importance of the tree adaptation: the highest obtained errors correspond to non-optimal trees, whereas the machine precision errors are obtained with optimal trees.



(a) Tree T^1, storage complexity of 428.

(b) Tree T^2, storage complexity of 560.

Figure 2.4: Exact representations of Function $(I.ii)$ using two different dimension trees. The dimensions associated with the leaf nodes are displayed in brackets and the α-ranks are indicated at each node.

T_σ	n	$\hat{\mathbb{P}}(T \text{ is optimal})$	$\varepsilon(S_{\text{test}}, g)$	CV error	$C(T, r)$
T^1_σ	$5 \cdot 10^2$	50%	$[4.23 \cdot 10^{-15}, 1.80 \cdot 10^{-1}]$	$[7.75 \cdot 10^{-16}, 1.64 \cdot 10^{-1}]$	$[\ 84, 921]$
	10^3	100%	$[6.64 \cdot 10^{-16}, 9.60 \cdot 10^{-15}]$	$[5.91 \cdot 10^{-16}, 1.84 \cdot 10^{-15}]$	$[428, 673]$
	10^4	100%	$[5.34 \cdot 10^{-16}, 1.18 \cdot 10^{-15}]$	$[5.24 \cdot 10^{-16}, 1.18 \cdot 10^{-15}]$	$[428, 428]$
T^2_σ	$5 \cdot 10^2$	70%	$[5.83 \cdot 10^{-15}, 1.94 \cdot 10^{-1}]$	$[8.87 \cdot 10^{-16}, 1.88 \cdot 10^{-1}]$	$[69, 1114]$
	10^3	90%	$[7.72 \cdot 10^{-16}, 2.43 \cdot 10^{-2}]$	$[6.61 \cdot 10^{-16}, 1.87 \cdot 10^{-2}]$	$[357, 515]$
	10^4	100%	$[5.59 \cdot 10^{-16}, 1.74 \cdot 10^{-15}]$	$[5.55 \cdot 10^{-16}, 1.75 \cdot 10^{-15}]$	$[428, 428]$

Table 2.3: Results for Function $(I.ii)$: training sample size n, starting from two families of trees T^1_σ and T^2_σ, estimation of the probability of obtaining an optimal tree and ranges (over the 10 trials) for the test error, the cross-validation (CV) error estimator and the storage complexity.

Table 2.4 shows the influence of the tree adaptation for a training sample size of $n = 1000$. We notice that the tree adaptation enables the recovery of Function $(I.ii)$, whereas without tree adaptation, the obtained error is high, not going below 10^{-3}. Even if there exists an exact representation of the function f whatever the chosen tree, this example shows that the tree adaptation is essential when only a few samples are available.

Illustration in the noisy case. We now consider a noisy case $Y = f(X) + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \zeta^2)$ is independent of X, and a training sample $S = \{(x_i, y_i)\}_{i=1}^n$ with $y_i = f(x_i) + \varepsilon_i$, where the ε_i are i.i.d. realizations of ε.

Table 2.5 shows the obtained results for different training sample sizes and standard deviations of the noise. We observe that, except with the smallest sample size and largest noise standard deviation, the algorithm is able to recover with high probability an optimal tree

T_σ	Tree adaptation	$\varepsilon(S_{\text{test}}, g)$	CV error	$C(T, r)$
T_σ^1	Yes	$[6.64{\cdot}10^{-16}, 9.60{\cdot}10^{-15}]$	$[5.91{\cdot}10^{-16}, 1.84{\cdot}10^{-15}]$	$[428, 673]$
	No	$[9.31 \cdot 10^{-3}, 1.25 \cdot 10^{-1}]$	$[4.68 \cdot 10^{-3}, 1.13 \cdot 10^{-1}]$	$[184, 786]$
T_σ^2	Yes	$[7.72 \cdot 10^{-16}, 2.43 \cdot 10^{-2}]$	$[6.61 \cdot 10^{-16}, 1.87 \cdot 10^{-2}]$	$[357, 515]$
	No	$[9.89 \cdot 10^{-3}, 9.69 \cdot 10^{-2}]$	$[4.77 \cdot 10^{-3}, 8.55 \cdot 10^{-2}]$	$[221, 728]$

Table 2.4: Results for Function $(I.ii)$ with $n = 1000$, with and without tree adaptation and starting from the tree T_σ^1 or T_σ^2 with a random initial permutation σ: ranges (over the 10 trials) for the test error, the cross-validation (CV) error and the storage complexity.

grouping consecutive input variables. We also notice that the cross-validation error is not always a good estimator of the generalization error for noisy observations. This can be an issue for model selection: at the last step of Algorithm 9, the selected model is the one leading to the smallest risk, estimated using a validation set V independent of the training set S. In many practical cases, one does not have access to a validation set, and must then rely on other estimators of the generalization error, such as a cross-validation estimator. Hence, the model selected using a cross-validation estimator might not be the one minimizing the generalization error.

n	ζ	$\hat{\mathbb{P}}(T \text{ is optimal})$	$\varepsilon(S_{\text{test}}, g)$	CV error	$C(T, r)$
$5 \cdot 10^2$	10^{-1}	50%	$[3.26{\cdot}10^{-2}, 9.78{\cdot}10^{-2}]$	$[2.30{\cdot}10^{-2}, 9.06{\cdot}10^{-2}]$	$[142, 271]$
	10^{-2}	80%	$[3.64{\cdot}10^{-3}, 1.35{\cdot}10^{-2}]$	$[1.35{\cdot}10^{-3}, 6.25{\cdot}10^{-3}]$	$[298, 518]$
	10^{-3}	80%	$[1.27{\cdot}10^{-4}, 1.52{\cdot}10^{-1}]$	$[1.34{\cdot}10^{-4}, 1.16{\cdot}10^{-1}]$	$[114, 515]$
10^3	10^{-1}	30%	$[1.00{\cdot}10^{-2}, 1.23{\cdot}10^{-1}]$	$[1.65{\cdot}10^{-2}, 9.43{\cdot}10^{-2}]$	$[117, 399]$
	10^{-2}	80%	$[7.52{\cdot}10^{-4}, 6.19{\cdot}10^{-3}]$	$[1.55{\cdot}10^{-3}, 3.75{\cdot}10^{-3}]$	$[428, 546]$
	10^{-3}	100%	$[7.63{\cdot}10^{-5}, 1.04{\cdot}10^{-4}]$	$[1.62{\cdot}10^{-4}, 1.73{\cdot}10^{-4}]$	$[428, 468]$
10^4	10^{-1}	90%	$[2.17{\cdot}10^{-3}, 1.65{\cdot}10^{-2}]$	$[1.70{\cdot}10^{-2}, 1.90{\cdot}10^{-2}]$	$[282, 631]$
	10^{-2}	90%	$[2.08{\cdot}10^{-4}, 2.31{\cdot}10^{-4}]$	$[1.71{\cdot}10^{-3}, 1.79{\cdot}10^{-3}]$	$[428, 527]$
	10^{-3}	100%	$[2.09{\cdot}10^{-5}, 2.40{\cdot}10^{-5}]$	$[1.71{\cdot}10^{-4}, 1.79{\cdot}10^{-4}]$	$[428, 528]$

Table 2.5: Results for Function $(I.ii)$ in the noisy case: training sample size n, standard deviation of the noise, estimation of the probability of obtaining an optimal tree and ranges (over the 10 trials) for the test error, the cross-validation (CV) error estimator and the storage complexity.

We recall that the risk writes $\mathcal{R}(g) = \mathcal{R}(f) + \|f - g\|^2$, with $\mathcal{R}(f) = \mathbb{E}((Y - f(X))^2) = \mathbb{E}(\epsilon^2) = \zeta^2$. Then $\mathcal{R}(g)$ is the sum of the squared approximation error $\|f - g\|^2$ and of the variance of the noise. Table 2.6 presents, for different values of n and ζ, the ranges (over the 10 trials) for the estimated squared approximation error defined by $\|f - g\|^2_{S_{\text{test}}} = \frac{1}{\#S_{\text{test}}} \sum_{(x,y) \in S_{\text{test}}} (f(x) - g(x))^2$, with a sample S_{test} independent of S. We note that the algorithm is robust with respect to noise and yields (with high probability) an approximation error which is below the noise level, and we clearly observe (as expected) that the approximation error decreases with n, whatever the noise level.

n	ζ^2	$\|f-g\|^2_{S_{\text{test}}}$
$5\cdot 10^2$	10^{-2}	$[3.40\cdot 10^{-2}, 3.09\cdot 10^{-1}]$
	10^{-4}	$[4.29\cdot 10^{-4}, 5.93\cdot 10^{-3}]$
	10^{-6}	$[5.23\cdot 10^{-7}, 7.44\cdot 10^{-1}]$
10^3	10^{-2}	$[3.23\cdot 10^{-3}, 4.89\cdot 10^{-1}]$
	10^{-4}	$[1.81\cdot 10^{-5}, 1.23\cdot 10^{-3}]$
	10^{-6}	$[1.87\cdot 10^{-7}, 3.50\cdot 10^{-7}]$
10^4	10^{-2}	$[1.52\cdot 10^{-4}, 8.77\cdot 10^{-3}]$
	10^{-4}	$[1.38\cdot 10^{-6}, 1.72\cdot 10^{-6}]$
	10^{-6}	$[1.42\cdot 10^{-8}, 1.85\cdot 10^{-8}]$

Table 2.6: Results for Function $(I.ii)$ in the noisy case: training sample size n, variance of the noise and ranges (over the 10 trials) for the estimated squared approximation error.

2.7.1.3 Function of a sum of bivariate functions

We here consider the approximation of the function

$$f(X) = \log\left(1 + (h(X_1, X_2) + \cdots + h(X_{d-1}, X_d))^2\right) \qquad (I.iii)$$

where $d = 10$, the X_i are independent and uniform on $[-1, 1]$, we consider the noiseless case $\varepsilon = 0$ and the function h is defined in Section 2.7.1.2. We use approximation spaces $\mathcal{H}_\nu = \mathbb{P}_{10}(\mathcal{X}^\nu)$ and Algorithm 9, with an adaptive selection of the tree T in the family of trees of arity 2, each run starting from two different families of trees $T^1_\sigma = \{\sigma(\alpha) : \alpha \in T^1\}$ and $T^2_\sigma = \{\sigma(\alpha) : \alpha \in T^2\}$, with σ a permutation of D and where T^1 and T^2 are visible in Figure 2.4.

Table 2.7 shows that with high probability, the algorithm yields a very accurate approximation with a small sample size and that the accuracy increases with the sample size. Also, we observe that the algorithm yields the expected optimal tree (as defined in Section 2.7.1.2) with high probability. Decreasing further the error would require an increase of the sample size and of the degree of the polynomial spaces.

T_σ	n	$\hat{\mathbb{P}}(T \text{ is optimal})$	$\varepsilon(S_{\text{test}}, v)$	CV error	$C(T, r)$
T^1_σ	$5\cdot 10^2$	80%	$[4.19\cdot 10^{-3}, 5.57\cdot 10^{-2}]$	$[2.15\cdot 10^{-3}, 5.00\cdot 10^{-2}]$	$[219, 573]$
	10^3	100%	$[8.77\cdot 10^{-5}, 2.04\cdot 10^{-2}]$	$[8.79\cdot 10^{-6}, 1.53\cdot 10^{-2}]$	$[277, 1417]$
	10^4	90%	$[1.11\cdot 10^{-5}, 1.99\cdot 10^{-2}]$	$[6.50\cdot 10^{-6}, 1.68\cdot 10^{-2}]$	$[277, 1834]$
T^2_σ	$5\cdot 10^2$	70%	$[1.01\cdot 10^{-3}, 7.09\cdot 10^{-2}]$	$[1.00\cdot 10^{-4}, 5.02\cdot 10^{-2}]$	$[211, 1289]$
	10^3	90%	$[8.34\cdot 10^{-5}, 5.29\cdot 10^{-2}]$	$[1.23\cdot 10^{-5}, 4.87\cdot 10^{-2}]$	$[277, 1566]$
	10^4	100%	$[1.15\cdot 10^{-5}, 1.98\cdot 10^{-2}]$	$[8.65\cdot 10^{-6}, 1.64\cdot 10^{-2}]$	$[277, 1290]$

Table 2.7: Results for Function $(I.iii)$: training sample size n, estimation of the probability of obtaining an optimal tree and ranges (over the 10 trials) for the test error, the cross-validation (CV) error estimator and the storage complexity.

2.7.1.4 Another sum of bivariate functions

We consider another sum of bivariate functions

$$f(X) = h(X_1, X_2) + h(X_2, X_3) + h(X_3, X_4) + \cdots + h(X_{d-1}, X_d), \qquad (I.iv)$$

where $d = 16$, $\varepsilon = 0$, the X_i are independent and uniform on $[-1, 1]$ and the function h is defined in Section 2.7.1.2. We consider $\mathcal{H} = \bigotimes_{\nu=1}^{d} \mathbb{P}_5(\mathcal{X}^\nu)$ so that $f \in \mathcal{H}$ (no discretization errors). For such a function, the linear tree $T^1 = \{\{1\}, \ldots, \{d\}, \{1, 2\}, \ldots, \{1, \ldots, d\}\}$ seems to be a natural choice, although it is not obvious that it is the optimal one. The algorithm 9 is run several times starting from T^1 or random permutations T^1_σ of T^1. Table 2.8 shows the obtained results with a training sample size $n = 10^4$, with or without tree adaptation. It first illustrates that, without tree adaptation, choosing a linear tree T^1 leads to a recovery of the function whereas choosing a tree T^1_σ with σ randomly drawn leads to a poor approximation of the function. However, with tree adaptation, the algorithm recovers the function at machine precision, whatever the starting permutation σ (over the 10 trials).

Figure 2.5 shows examples of final trees obtained when running Algorithm 9, starting from two different random permutations of T^1. We notice that the algorithm returns non obvious trees, selected in the family of trees of arity 2, whose nodes contain consecutive variables.

T_σ	Tree adaptation	$\varepsilon(S_{\text{test}}, v)$	CV error	$C(T, r)$
T^1_{id}	false	$[2.28 \cdot 10^{-15}, 1.98 \cdot 10^{-14}]$	$[1.67 \cdot 10^{-15}, 9.87 \cdot 10^{-15}]$	$[1760, 3131]$
	true	$[3.79 \cdot 10^{-15}, 2.09 \cdot 10^{-14}]$	$[2.30 \cdot 10^{-15}, 1.26 \cdot 10^{-14}]$	$[1800, 2974]$
T^1_σ	false	$[4.48 \cdot 10^{-03}, 5.06 \cdot 10^{-03}]$	$[2.49 \cdot 10^{-03}, 3.14 \cdot 10^{-03}]$	$[4779, 5490]$
	true	$[4.81 \cdot 10^{-15}, 3.35 \cdot 10^{-14}]$	$[2.88 \cdot 10^{-15}, 1.59 \cdot 10^{-14}]$	$[1791, 2428]$

Table 2.8: Results for Function $(I.iv)$ for a training sample size $n = 10^4$, with and without tree adaptation, and starting from $T^1 = T^1_{id}$ or random permutations T^1_σ of T^1: ranges (over the 10 trials) for the test error, the cross-validation (CV) error estimator and the storage complexity.

2.7.1.5 Compositions of functions

We consider the approximation of

$$f(X) = h(h(h(X_1, X_2), h(X_3, X_4)), h(h(X_5, X_6), h(X_7, X_8))), \qquad (I.v)$$

where h is a bivariate function and where the $d = 8$ random variables $X_1, \ldots, X_8$ are independent and uniform on $[-1, 1]$. The noise level is set to $\varepsilon = 0$. The function f is obtained by tree-structured compositions of the function h, illustrated in Figure 2.6, where each interior node of the tree corresponds to the application of h to the outputs of its children nodes.

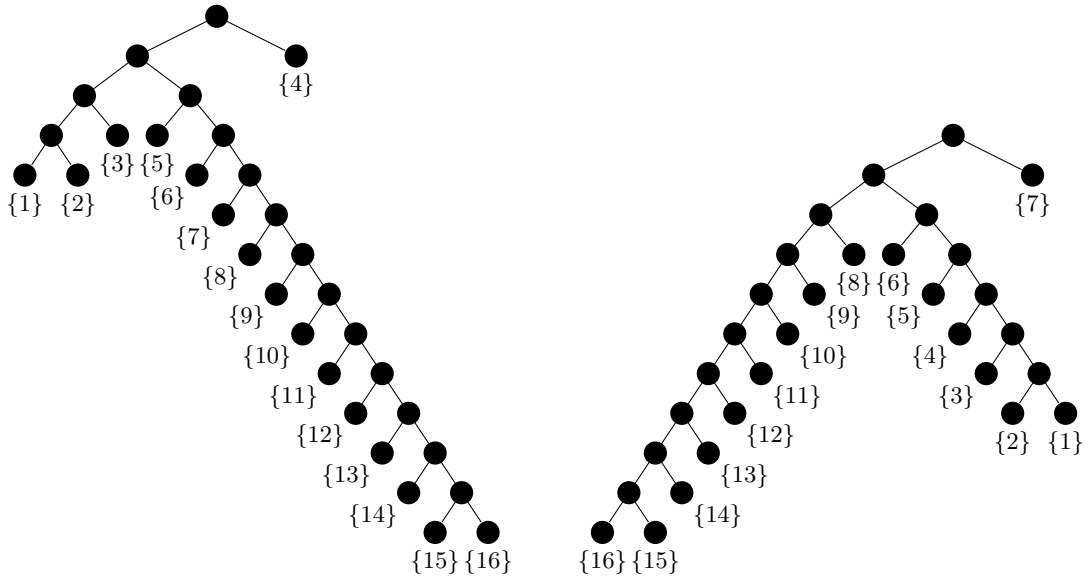


Figure 2.5: Examples of final trees obtained when running Algorithm 9 with a training sample size $n = 10^4$, starting from two different random permutations of T^1.

Here we choose $h(t, s) = 9^{-1}(2 + ts)^2$. Therefore, f is a polynomial function of degree 8. Then, we use approximation spaces $\mathcal{H}_\nu = \mathbb{P}_8(\mathcal{X}_\nu)$, so that f belongs to $\mathcal{H}$ and could (in principle) be recovered exactly for any choice of tree with a sufficiently high rank.

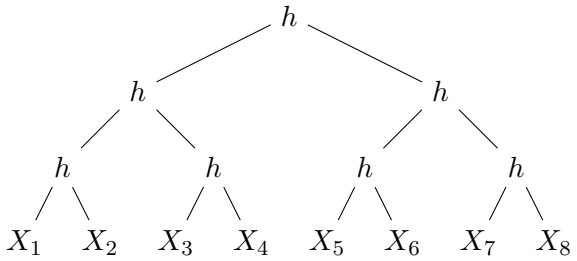
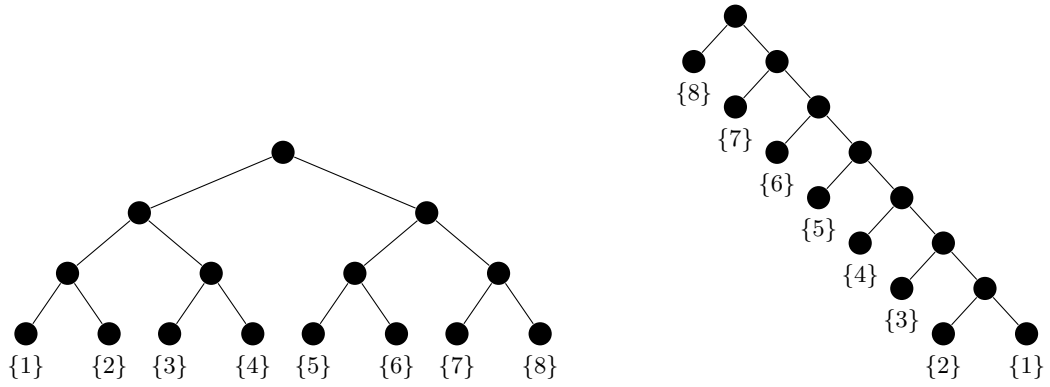


Figure 2.6: Schematic representation of Function $(I.v)$.

There is a natural dimension tree T^1 associated with this function, illustrated in Figure 2.7a. Using this tree T^1, the function can be exactly represented in the format $\mathcal{T}_r^{T^1}(\mathcal{H})$ with a tree-based rank $r = (r_\alpha)_{\alpha \in T^1}$ such that $r_\alpha = 1 + 2^{\text{level}(\alpha)}$ for $\alpha \neq D$, and a storage complexity of 2427. Although the function can be exactly represented with any choice of tree, the ranks could be dramatically high for bad choices of tree. For example, when considering the tree T^1_σ obtained by applying the permutation $\sigma = (8, 1, 6, 4, 7, 2, 3, 5)$ to T^1, we obtain a representation with ranks more than 1000 (at level 1) and a storage complexity greater than $9 \cdot 10^6$ for a representation with relative precision 10^{-14}.

For the application of Algorithm 9 with tree adaptation, we start from trees belonging to two families T^1_σ and T^2_σ, respectively obtained by permutations of the trees T^1 and T^2 shown in Figures 2.7a and 2.7b. Each run of the algorithm starts with a random permutation σ. Recall that different trees T_σ may yield the same tree-based format, if they coincide as elements of 2^{2^D}. For example, for $T = T^1_\sigma$ with $\sigma = (7, 8, 6, 5, 4, 3, 1, 2)$, $T = T^1$. Then, we

will say the algorithm finds an optimal tree if it yields a permutation such that $T = T^1$ as elements of 2^{2^D}.



(a) Tree T^1. (b) Tree T^2.

Figure 2.7: Two different dimension trees T^1 and T^2.

Table 2.9 summarizes the obtained results. We observe that the algorithm is able to recover with high probability an optimal tree, even when starting from the tree T^2_σ that does not coincide with the natural tree structure of the function f.

With high probability, the algorithm yields an approximation with very low error, even with a small sample size n. With a training sample large enough, the algorithm is able to recover the function f at machine precision. Figure 2.8 shows the α-ranks of an approximation computed by the algorithm with $n = 10^5$ with a generalization error at machine precision.

T_σ	n	$\hat{\mathbb{P}}(T = T^1)$	$\varepsilon(S_{\text{test}}, v)$	CV error	$C(T, r)$
T^1_σ	10^3	90%	$[1.55 \cdot 10^{-05}, 1.32 \cdot 10^{-04}]$	$[8.46 \cdot 10^{-07}, 3.38 \cdot 10^{-05}]$	$[529, 1121]$
	10^4	100%	$[1.04 \cdot 10^{-08}, 6.80 \cdot 10^{-06}]$	$[4.34 \cdot 10^{-11}, 4.91 \cdot 10^{-06}]$	$[593, 2688]$
	10^5	100%	$[3.29 \cdot 10^{-15}, 1.80 \cdot 10^{-04}]$	$[1.74 \cdot 10^{-15}, 1.96 \cdot 10^{-04}]$	$[342, 2800]$
T^2_σ	10^3	90%	$[1.75 \cdot 10^{-05}, 1.75 \cdot 10^{-04}]$	$[1.01 \cdot 10^{-06}, 8.71 \cdot 10^{-05}]$	$[360, 1062]$
	10^4	90%	$[2.15 \cdot 10^{-08}, 4.10 \cdot 10^{-03}]$	$[1.21 \cdot 10^{-09}, 4.26 \cdot 10^{-03}]$	$[185, 2741]$
	10^5	100%	$[4.67 \cdot 10^{-15}, 8.92 \cdot 10^{-03}]$	$[2.29 \cdot 10^{-15}, 6.83 \cdot 10^{-03}]$	$[163, 2594]$

Table 2.9: Results for Function $(I.v)$: training sample size n, estimation of the probability of obtaining T^1 and ranges (over the 10 trials) for the test error, the cross-validation (CV) error estimator and the storage complexity.

Illustration of the behavior of algorithm 9. Table 2.10 illustrates the behavior of Algorithm 9 when using a training sample of size $n = 10^5$ and starting from a tree T^2_σ shown in Figure 2.9a. The adapted trees at each iteration m are displayed in Figure 2.9. We observe that the algorithm recovers the function with a high accuracy after 25 iterations and 7 adaptations of the tree.

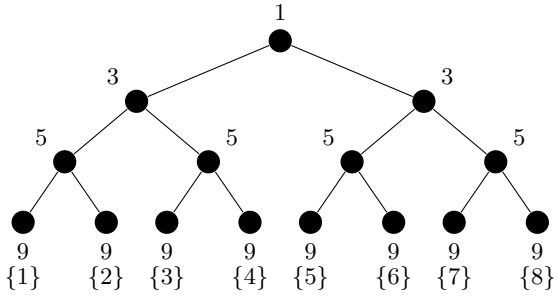


Figure 2.8: α-ranks and dimensions associated with leaf nodes of an approximation of Function $(I.v)$ obtained using Algorithm 9 with $n = 10^5$, starting from a tree T^2_σ. The obtained tree is T^1.

2.7.2 Density estimation

In this section, we illustrate the performance of the proposed algorithm for learning probability distributions using tree-based formats. Note that we do not normalize the obtained approximations nor impose their nonnegativity, so that they *a priori* do not define probability density functions.

Contrast function and reference measure. In all examples, we consider the L^2_μ contrast function $\gamma(g, x) = \|g\|^2 - 2g(x)$. The reference measure μ is always a product measure and $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$. For examples involving discrete random variables, we consider $\mu = \sum_{x\in\mathcal{X}} \delta_x$, so that $L^2_\mu(\mathcal{X})$ is identified with $\ell^2(\mathcal{X})$. For continuous random variables, μ is taken as the Lebesgue measure or a uniform probability measure on $\mathcal{X}$ when $\mathcal{X}$ is a compact set.

Approximation spaces. In the case of discrete random variables with a finite set $\mathcal{X}$, we let $\mathcal{H} = L^2_\mu(\mathcal{X})$ so that there is no discretization error, and we use a canonical basis (see Remark 1.2.2). In the case of continuous random variables, for each dimension $\nu = 1, \ldots, d$, we introduce a finite dimensional space $\mathcal{H}_\nu$ in $L^2_{\mu_\nu}(\mathcal{X}_\nu)$ and use orthonormal bases of $\mathcal{H}_\nu$ (*e.g.* polynomials, wavelets). We exploit sparsity in the leaf tensors $(C^\alpha)_{\alpha\in\mathcal{L}(T)}$ by using a working set strategy. For polynomial bases, we use the natural sequence of candidate patterns associated with spaces of polynomials with increasing degree.

Tensor formats. We only consider tensor formats associated with binary dimension partition trees. We use Algorithm 9 with rank and tree adaptation, always starting with a linear dimension tree (such as in Figure 3c) where the dimensions $\nu = 1, \ldots, d$ are randomly assigned to the leaf nodes. We set $\varepsilon = 10^{-6}$ and $N = 1000$ in Algorithm 5, and let the maximum number of iterations in Algorithm 9 be sufficiently high.

m	Tree T	Tree-based rank r^m	$\varepsilon(S_{\text{test}}, v)$	$C(T, r^m)$
1	Fig. 2.9a	$(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)$	$3.38 \cdot 10^{-2}$	79
2	Fig. 2.9b	$(1,1,2,1,1,1,1,1,1,1,1,1,2,1,1)$	$2.95 \cdot 10^{-2}$	100
3		$(1,1,2,1,1,1,1,1,1,1,1,1,2,1,1)$	$2.95 \cdot 10^{-2}$	100
4	Fig. 2.9c	$(1,1,2,1,2,1,1,1,2,1,1,1,2,1,1)$	$2.45 \cdot 10^{-2}$	121
5		$(1,1,2,1,2,1,1,1,2,1,1,1,2,1,1)$	$2.45 \cdot 10^{-2}$	121
6	Fig. 2.9d	$(1,1,2,1,2,1,1,1,2,1,2,1,2,2,1)$	$1.85 \cdot 10^{-2}$	142
7		$(1,1,2,1,2,1,1,1,2,1,2,1,2,2,1)$	$1.85 \cdot 10^{-2}$	142
8		$(1,1,2,1,2,1,2,1,2,1,2,1,2,2,2)$	$8.97 \cdot 10^{-3}$	163
9	Fig. 2.9e	$(1,2,2,2,2,2,2,2,2,1,2,2,2,2,2)$	$9.54 \cdot 10^{-3}$	188
10		$(1,2,2,2,2,2,2,2,2,1,2,2,2,2,2)$	$8.89 \cdot 10^{-3}$	188
11	Fig. 2.9f	$(1,2,2,2,2,2,2,2,2,1,2,2,2,2,2)$	$9.47 \cdot 10^{-3}$	188
12		$(1,2,2,2,2,2,2,2,2,1,2,2,2,2,2)$	$8.87 \cdot 10^{-3}$	188
13	Fig. 2.9g	$(1,2,2,2,2,2,2,2,2,1,2,2,2,2,2)$	$5.22 \cdot 10^{-3}$	188
14		$(1,2,2,2,2,2,2,2,2,1,2,2,2,2,2)$	$3.97 \cdot 10^{-3}$	188
15		$(1,2,3,2,3,2,3,2,3,2,3,2,3,3,3)$	$1.55 \cdot 10^{-4}$	308
16	Fig. 2.9h	$(1,3,3,3,3,2,3,3,3,2,3,3,3,3,3)$	$1.18 \cdot 10^{-4}$	364
17		$(1,3,3,3,3,2,3,3,3,2,3,3,3,3,3)$	$1.18 \cdot 10^{-4}$	364
18		$(1,3,4,3,4,2,4,3,4,2,4,3,4,4,4)$	$6.65 \cdot 10^{-6}$	520
19		$(1,3,5,3,5,3,5,3,5,3,5,3,5,5,5)$	$1.19 \cdot 10^{-6}$	723
20		$(1,4,5,4,5,3,5,4,5,3,5,4,5,5,5)$	$1.72 \cdot 10^{-7}$	865
21		$(1,4,6,4,6,3,6,4,6,3,6,4,6,6,6)$	$1.47 \cdot 10^{-8}$	1113
22		$(1,5,6,5,6,3,6,5,6,3,6,5,6,6,6)$	$7.02 \cdot 10^{-9}$	1311
23		$(1,5,7,5,7,3,7,5,7,3,7,5,7,7,7)$	$1.27 \cdot 10^{-10}$	1643
24		$(1,5,8,5,8,3,8,5,8,3,8,5,8,8,8)$	$3.87 \cdot 10^{-12}$	2015
25		$(1,5,9,5,9,3,9,5,9,3,9,5,9,9,9)$	$2.95 \cdot 10^{-14}$	2427

Table 2.10: Behavior of Algorithm 9 for the approximation of Function $(I.v)$, with $n = 10^5$ and an initial dimension tree shown in Figure 2.9a.

Error measures. The quality of the obtained approximation g is assessed by estimating the risk by

$$\mathcal{R}_{S_{\text{test}}}(g) = \|g\|^2 - \frac{2}{\#S_{\text{test}}} \sum_{x \in S_{\text{test}}} g(x),$$

with S_{test} a sample of X, independent of S, as well as, when f can be evaluated, by computing the relative error

$$\varepsilon(S_\varepsilon, g) = \left(\frac{\sum_{x \in S_\varepsilon} (f(x) - g(x))^2}{\sum_{x \in S_\varepsilon} (f(x))^2} \right)^{1/2},$$

with S_ε a sample from μ if μ is a probability measure, or from $\frac{1}{\mu(\mathcal{X})}\mu$ when μ is a finite measure (*e.g.* when μ is the Lebesgue measure over a compact set $\mathcal{X}$). In the case of discrete random variables, a function f in $\mathbb{R}^{\mathcal{X}}$ is identified with a multi-dimensional array, and S_ϵ corresponds to a sample of the entries of the array.

2.7.2.1 Truncated multivariate normal distribution

We first consider the estimation of the density of a random vector $X = (X_1, \dots, X_6)$ following a truncated normal distribution with zero mean and covariance matrix Σ. Its support is $\mathcal{X} = \times_{\nu=1}^{6}[-5\sigma_\nu, 5\sigma_\nu]$, with $\sigma_\nu^2 = \Sigma_{\nu\nu}$. The reference measure μ is the Lebesgue measure on $\mathcal{X}$ and the density to approximate is such that

$$f(x) \propto \exp\left(-\frac{1}{2}x^T\Sigma^{-1}x\right)\mathbf{1}_{x\in\mathcal{X}}. \tag{II.i}$$

We use in each dimension ν polynomials of maximal degree 50, orthonormal in $L^2(\mathcal{X}_\nu)$.

Groups of independent random variables. We consider the following covariance matrix

$$\Sigma = \begin{pmatrix} 2 & 0 & 0.5 & 1 & 0 & 0.5 \\ 0 & 1 & 0 & 0 & 0.5 & 0 \\ 0.5 & 0 & 2 & 0 & 0 & 1 \\ 1 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 1 & 0 \\ 0.5 & 0 & 1 & 0 & 0 & 2 \end{pmatrix}. \tag{Σ_1}$$

Up to a permutation $(3, 6, 1, 4, 2, 5)$ of its rows and columns, it can be written

$$\begin{pmatrix} 2 & 1 & 0.5 & 0 & 0 & 0 \\ 1 & 2 & 0.5 & 0 & 0 & 0 \\ 0.5 & 0.5 & 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 0.5 & 1 \end{pmatrix}$$

so that one can see that the random variables (X_1, X_3, X_4, X_6) and (X_2, X_5) are independent, as well as X_4 and (X_3, X_6). Therefore, the density has the form

$$f(x) = f_{1,3,4,6}(x_1, x_3, x_4, x_6)f_{2,5}(x_2, x_5) = f_{4|1}(x_4|x_1)f_{1,3,6}(x_1, x_3, x_6)f_{2,5}(x_2, x_5).$$

Then one can then expect that, when approximating the density of X in tree-based format, a suitable dimension tree T would contain the nodes $\{2, 5\}$ and $\{1, 3, 4, 6\}$, since $\text{rank}_{\{2,5\}}(f) = \text{rank}_{\{1,3,4,6\}}(f) = 1$. If we further assume that $f_{1,3,6}$ has low ranks, it would contain the nodes $\{3, 6\}$ and $\{1, 4\}$, since $\text{rank}_{\{3,6\}}(f) = \text{rank}_{\{3,6\}}(f_{1,3,6})$ and $\text{rank}_{\{1,4\}}(f) = \text{rank}_{\{1,4\}}(f_{1,3,6})$ (see a possible tree in Figure 2.10).

Table 2.11 shows the results obtained with the learning algorithm with different sizes of training set. We notice that, as expected, $\mathcal{R}_{S_{\text{test}}}(g)$ and $\varepsilon(S_\varepsilon, g)$ decrease with n. In Figure 2.11, we observe the obtained tree (associated with the smallest error over 10 trials) for

different training sample sizes n. For $n \geq 10^4$, the algorithm yields a tree with the expected nodes.

n	$\mathcal{R}_{S_{\text{test}}}(g) \cdot 10^{-2}$	$\varepsilon(S_\varepsilon, g)$	T	$C(T, r)$
10^2	$[-5.50, 119]$	$[0.53, 4.06]$	Fig. 2.11a	$[311, 311]$
10^3	$[-7.29, -5.93]$	$[0.22, 0.47]$	Fig. 2.11b	$[311, 637]$
10^4	$[-7.60, -6.85]$	$[0.11, 0.33]$	Fig. 2.11c	$[521, 911]$
10^5	$[-7.68, -7.66]$	$[0.04, 0.07]$	Fig. 2.11c	$[911, 1213]$
10^6	$[-7.70, -7.69]$	$[0.01, 0.01]$	Fig. 2.11c	$[1283, 1546]$

Table 2.11: Ranges over 10 trials of the obtained results for the learning of $(II.i)$ with covariance matrix (Σ_1), with different training sample sizes n.

Band-diagonal covariance matrix. We now consider the following covariance matrix

$$\Sigma = \begin{pmatrix} 2 & 1/5 & 0 & 0 & 1/4 & 0 \\ 1/5 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1/3 & 1/2 \\ 0 & 0 & 0 & 2 & 0 & 1 \\ 1/4 & 0 & 1/3 & 0 & 2 & 0 \\ 0 & 0 & 1/2 & 1 & 0 & 2 \end{pmatrix} \tag{Σ_2}$$

which is, after applying the permutation $\sigma = (4, 6, 3, 5, 1, 2)$, a band diagonal matrix

$$\begin{pmatrix} 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 2 & 1/3 & 0 & 0 \\ 0 & 0 & 1/3 & 2 & 1/4 & 0 \\ 0 & 0 & 0 & 1/4 & 2 & 1/5 \\ 0 & 0 & 0 & 0 & 1/5 & 2 \end{pmatrix}.$$

The vector $(X_{\sigma(1)}, \ldots, X_{\sigma(6)})$ therefore represents a Markov process and the density f has the following form

$$f(x) = f_{2|1}(x_2|x_1) f_{1|5}(x_1|x_5) f_{5|3}(x_5|x_3) f_{3|6}(x_3|x_6) f_{6|4}(x_6|x_4).$$

Given this structure, one might expect the density of X to be efficiently and accurately represented in tree-based tensor format with one of the linear trees of Figure 2.12 or any tree containing the same internal nodes.

Table 2.12 shows the results obtained when using the learning algorithm to approximate the density of X. Figure 2.13 shows the obtained trees (associated with the smallest error over 10 trials) for different sizes of training sets. We observe that except for $n = 10^2$, the algorithm yields trees that contain most of the expected internal nodes.

n	$\mathcal{R}_{S_{\text{test}}}(g) \cdot 10^{-2}$	$\varepsilon(S_\varepsilon, g)$	T	$C(T, r)$
10^2	$[-4.21, 1.91]$	$[0.58, 1.11]$	Fig. 2.13a	$[311, 311]$
10^3	$[-5.83, -5.13]$	$[0.28, 0.45]$	Fig. 2.13b	$[311, 579]$
10^4	$[-6.26, -6.01]$	$[0.09, 0.22]$	Fig. 2.13c	$[416, 837]$
10^5	$[-6.30, -6.29]$	$[0.03, 0.05]$	Fig. 2.13d	$[835, 1083]$
10^6	$[-6.31, -6.31]$	$[0.01, 0.02]$	Fig. 2.13d	$[1008, 1291]$

Table 2.12: Obtained results for the learning of $(II.i)$ with covariance matrix (Σ_2), with different training sample sizes n.

2.7.2.2 Markov chain

In this section, we study a discrete time discrete state space Markov process. We have that $X = (X_1, \ldots, X_8)$, where each random variable X_ν takes values in $\mathcal{X}_\nu = \{1, \ldots, 5\}$. The distribution of X writes

$$f(i_1, \ldots, i_8) = \mathbb{P}(X_1 = i_1, \ldots, X_8 = i_8) = f_{d|d-1}(i_8|i_7) \cdots f_{2|1}(i_2|i_1) f_1(i_1) \qquad (II.ii)$$

with $f_1(i_1) = 1/5$ for all $i_1 \in \mathcal{X}_1$, and for $\nu = 1, \ldots, d-1$, $f_{\nu+1|\nu}(i_{\nu+1}|i_\nu) = P_{i_\nu, i_{\nu+1}}$ the $(i_\nu, i_{\nu+1})$ component of a randomly chosen rank-2 transition matrix P, independent of the dimension ν.

As shown in Example A.3.1 of Appendix A, that studies the same function, the choice of the tree has a great impact on the storage complexity of the representation of the Markov process. We then expect the adaptive learning algorithm to compute an approximation of f with a tree containing the same internal nodes as in Figure A.1a. Table 2.13 shows the ranges over 10 trials of the obtained results. One can notice that, even though the algorithm did not recover an optimal tree for the representation of the Markov chain, it is able, with a sample size high enough, to represent it with dimension trees including most of the internal nodes yielding the smallest α-ranks, limiting the complexity of the representation.

n	$\mathcal{R}_{S_{\text{test}}}(g)$	$\varepsilon(S_\varepsilon, g)$	T	$C(T, r)$
10^3	$[-2.22, -1.25]$	$[0.49, 0.75]$	Fig. 2.14a	$[47, 109]$
10^4	$[-2.85, -2.04]$	$[0.16, 0.55]$	Fig. 2.14b	$[72, 298]$
10^5	$[-2.93, -2.91]$	$[0.04, 0.08]$	Fig. 2.14c	$[294, 519]$
10^6	$[-2.93, -2.93]$	$[0.01, 0.02]$	Fig. 2.14d	$[384, 1010]$

Table 2.13: Ranges over 10 trials of the obtained results for the learning of $(II.ii)$, with different training sample sizes n.

2.7.2.3 Graphical model with discrete random variables

We consider the graphical model represented in Figure 2.15, in dimension $d = 10$. The random variable X_ν takes values in $\mathcal{X}_\nu = \{1, 2, 3, 4, 5\}$, $1 \leq \nu \leq d$, so that $f(i) = \mathbb{P}(X = i)$

is defined by

$$f(i_1, \ldots, i_{10}) = f_{1,2,3,7}(i_1, i_2, i_3, i_7) f_{3,4,5,6}(i_3, i_4, i_5, i_6) f_{4,8}(i_4, i_8) f_{8,9,10}(i_8, i_9, i_{10}). \quad (II.iii)$$

The tensors f_α are randomly selected under the constraint that any of their matricization has a rank equal to 3.

As shown in the motivating example of Section 1.8.1 of Chapter 1 for the tree adaptation, representing this function in tree-based tensor format with the binary tree in Figure 1.4 yields a storage complexity of 117027, whereas using the tree in Figure 1.5, which exhibits the dependence structure of the graphical model, leads to a representation with a storage complexity of 675. We then expect our algorithms to be able to learn f with a tree representing its dependence structure.

Table 2.14 shows the ranges over 10 trials of the obtained results. We observe that, even though the obtained errors are high, the algorithm is able to provide approximations of f with a tree that exhibits the dependence structure of the graphical model (for instance by containing the nodes $\{1, 2, 3, 7\}$, $\{4, 8\}$ or $\{4, 8, 9, 10\}$, which are cliques of the graph of f).

n	$\mathcal{R}_{S_{\text{test}}}(g)$	$\varepsilon(S_\varepsilon, g)$	T	$C(T, r)$
10^3	$[-1.08, -1.04]$	$[0.53, 0.55]$	Fig. 2.16a	$[59, 99]$
10^4	$[-1.36, -1.13]$	$[0.30, 0.49]$	Fig. 2.16b	$[72, 421]$
10^5	$[-1.48, -1.43]$	$[0.10, 0.20]$	Fig. 2.16c	$[496, 885]$
10^6	$[-1.49, -1.42]$	$[0.07, 0.22]$	Fig. 2.16d	$[373, 813]$

Table 2.14: Ranges over 10 trials of the obtained results for the learning of $(II.iii)$, with different training sample sizes n.

2.8 Conclusion

This chapter focused on the learning of functions with model classes of functions in tree-based tensor format.

After a presentation of the problem to solve, we described the main contributions of this chapter: adaptive learning algorithms in tree-based tensor format. We proposed in Chapter 1 a stochastic algorithm to perform adaptation of the dimension tree, and in Section 2.6.1, we proposed a heuristic algorithm to perform rank adaptation, by increasing a subset of α-ranks associated with the notion of highest truncation errors. These two adaptive algorithms were combined in Section 2.6.2 to propose a learning algorithm with model classes of functions in tree-based tensor format, with adaptation of the dimension tree and the tree-based ranks.

The performances of the algorithms were illustrated in Section 2.7 on numerical experiments in least-squares regression and least-squares density estimation. We showed that the obtained approximations in tree-based tensor format can, if enough training data is available, give information about the structure of the function to approximate.

A larger probability of recovering an optimal tree could be obtained by running Algorithm 9 M times, each starting from a different tree (of same arity), and by retaining the approximation giving the best result: if the probability of obtaining an optimal tree out of one trial is p, the probability of obtaining the optimal tree out of M trials by selecting the best approximation among them is $1 - (1 - p)^M$.

As expected, the quality of the approximation improves with the training sample size n. However, computational costs increase with n. In the case of large data sets, variants of the proposed algorithm using subsamples of the training sample could be proposed, in the spirit of stochastic gradient methods.

Finally, the proposed adaptive algorithms provide numerous approximations, each associated with different trees and different tree-based ranks. This calls for a robust approach for model selection or aggregation that does not rely on statistical estimations of the generalization error, either using an independent test sample (not used to train the model) or a cross-validation estimator (which may be a bad estimator if n is small and/or if the observations are noisy).

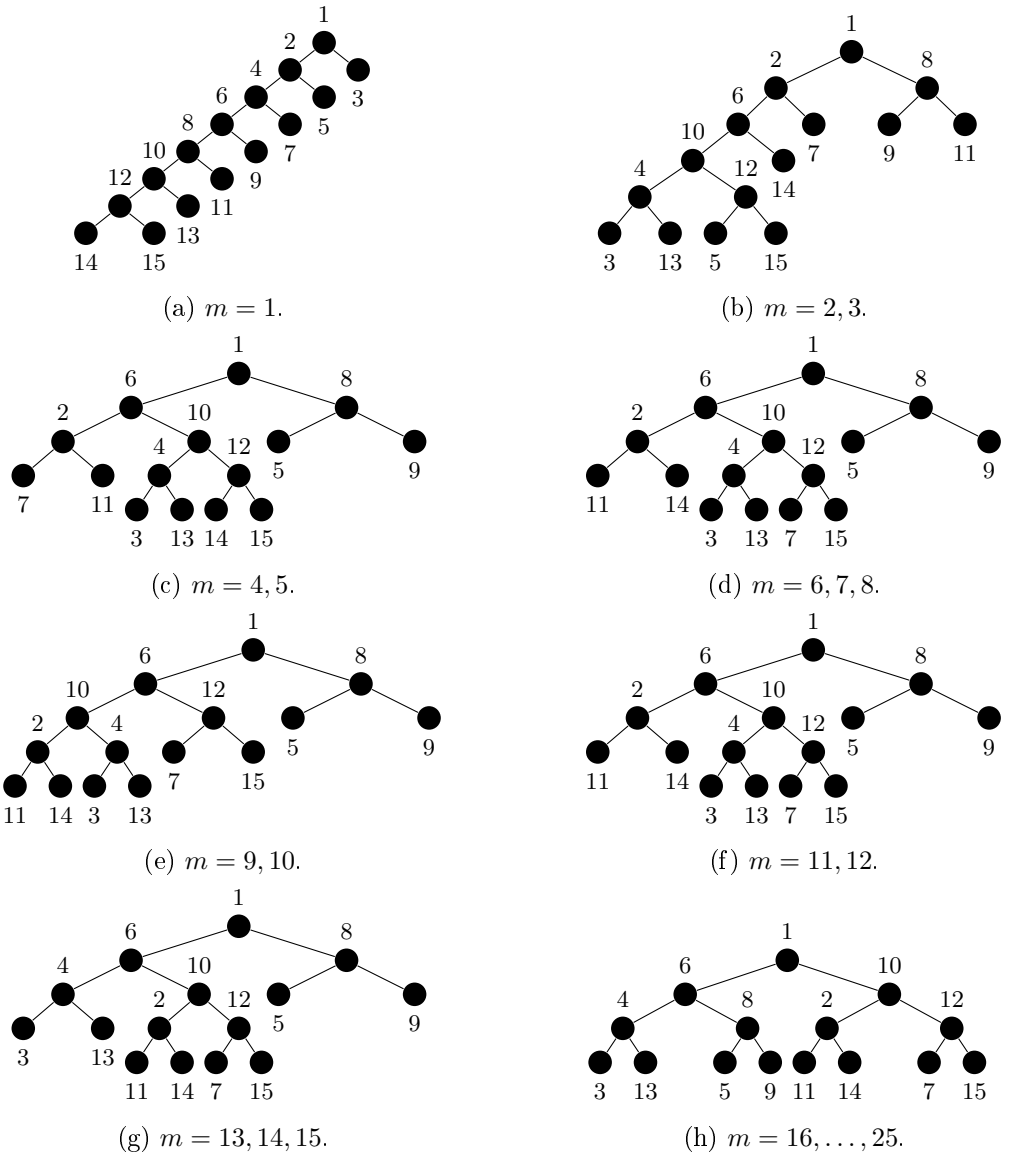


Figure 2.9: Dimension trees associated with the iteration number m in Table 2.10, with each node numbered. The singletons $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}$ correspond to the leaf nodes numbered $9, 5, 3, 13, 11, 14, 7, 15$ respectively.

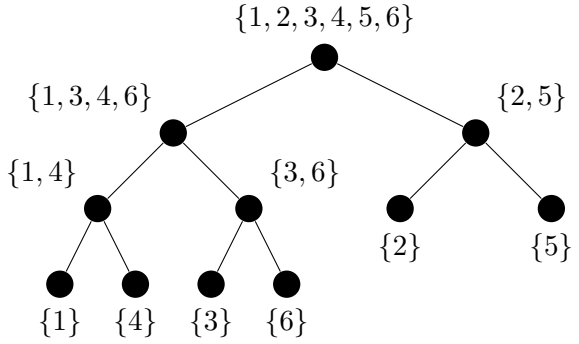
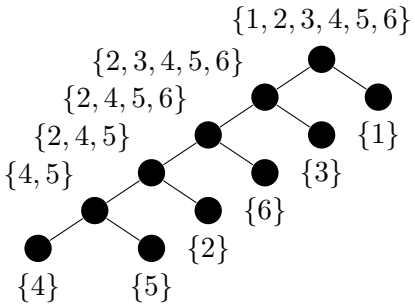
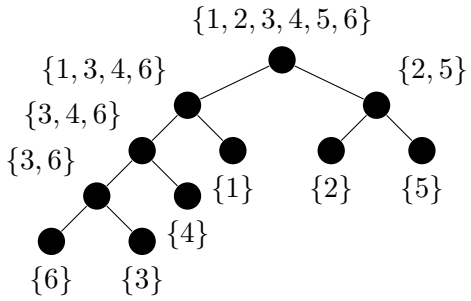


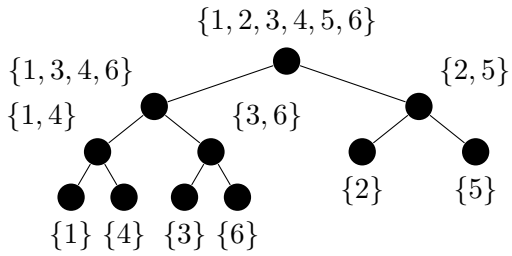
Figure 2.10: Example of expected tree T for the approximation of $(II.i)$ with covariance matrix (Σ_1) in tree-based tensor format with tree adaptation.



(a) $n = 10^2$.



(b) $n = 10^3$.



(c) $n = 10^4, 10^5, 10^6$.

Figure 2.11: Dimension trees T obtained after computing an approximation in tree-based tensor format of $(II.i)$ with covariance matrix (Σ_1), using different training sample sizes n. The displayed trees are associated with the smallest error over 10 trials.

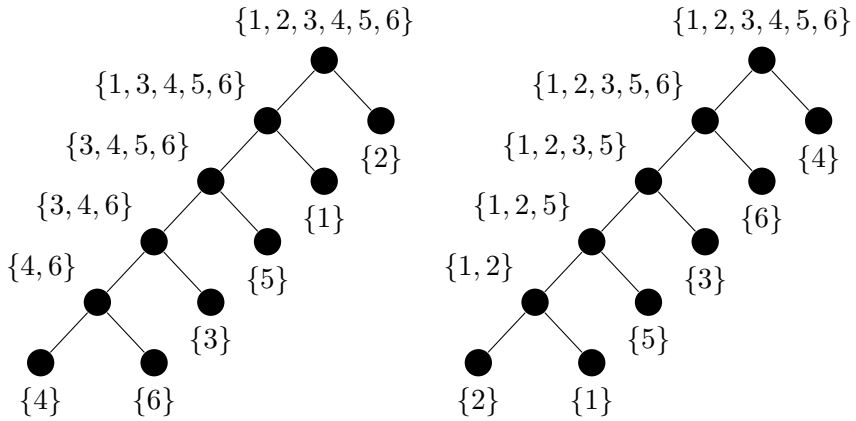
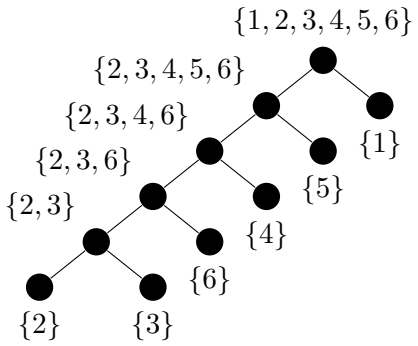
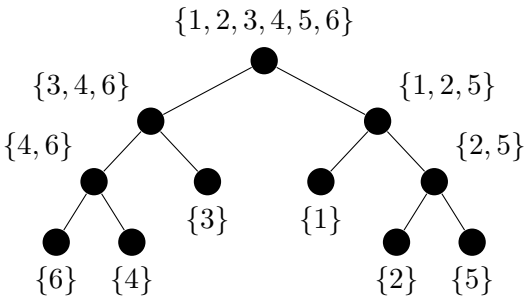


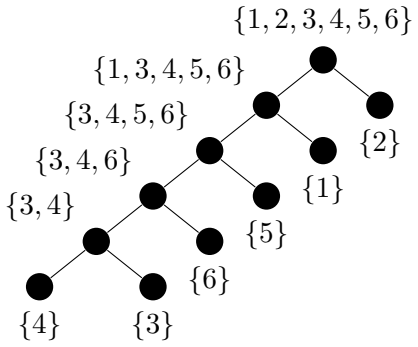
Figure 2.12: Example of expected trees T for the approximation of $(II.i)$ with covariance matrix (Σ_2) in tree-based tensor format.



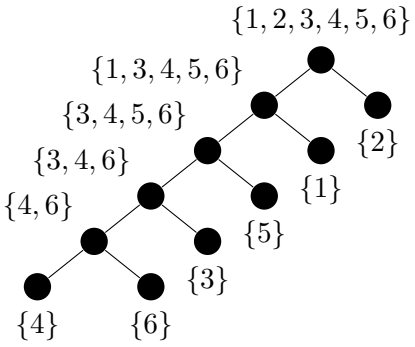
(a) $n = 10^2$.



(b) $n = 10^3$.

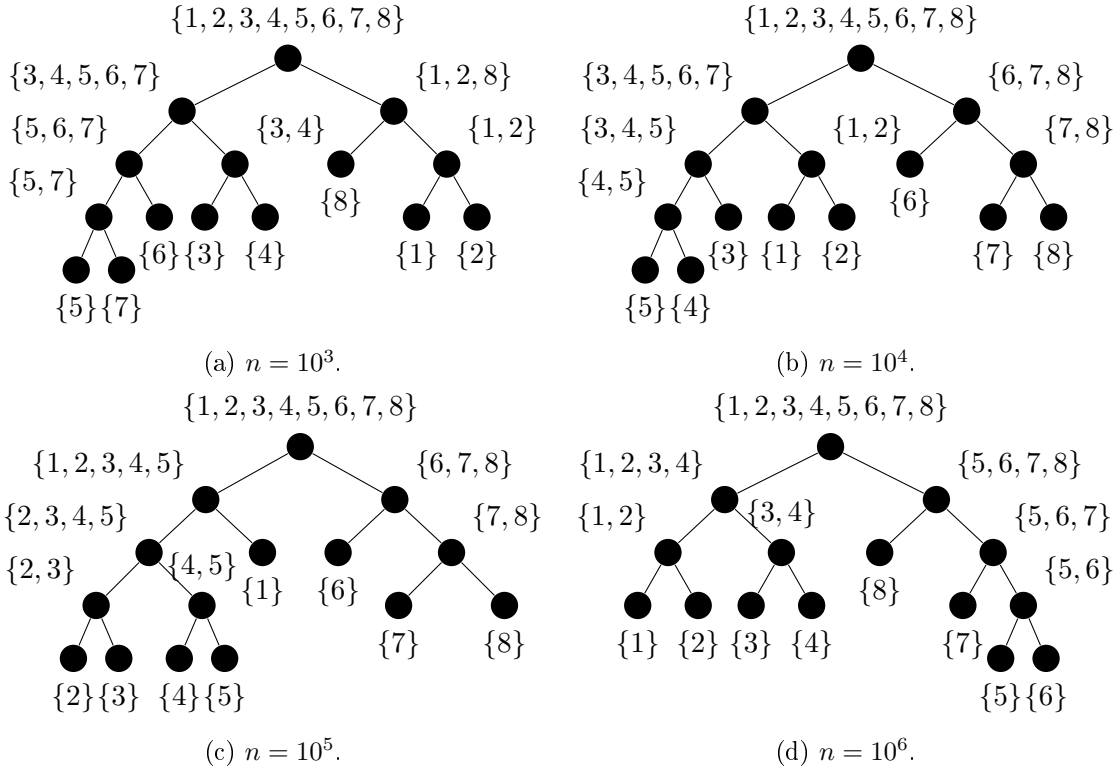


(c) $n = 10^4$.



(d) $n = 10^5, 10^6$.

Figure 2.13: Dimension trees T obtained after computing an approximation in tree-based tensor format of $(II.i)$ with covariance matrix (Σ_2) using different training sample sizes n. The displayed trees are associated with the smallest error over 10 trials.



(a) $n = 10^3$.

(b) $n = 10^4$.

(c) $n = 10^5$.

(d) $n = 10^6$.

Figure 2.14: Dimension trees T obtained after computing an approximation in tree-based tensor format of $(II.ii)$, using different training sample sizes n. The displayed trees are associated with the smallest error over 10 trials.

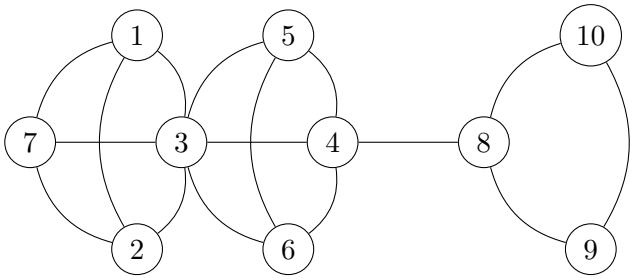
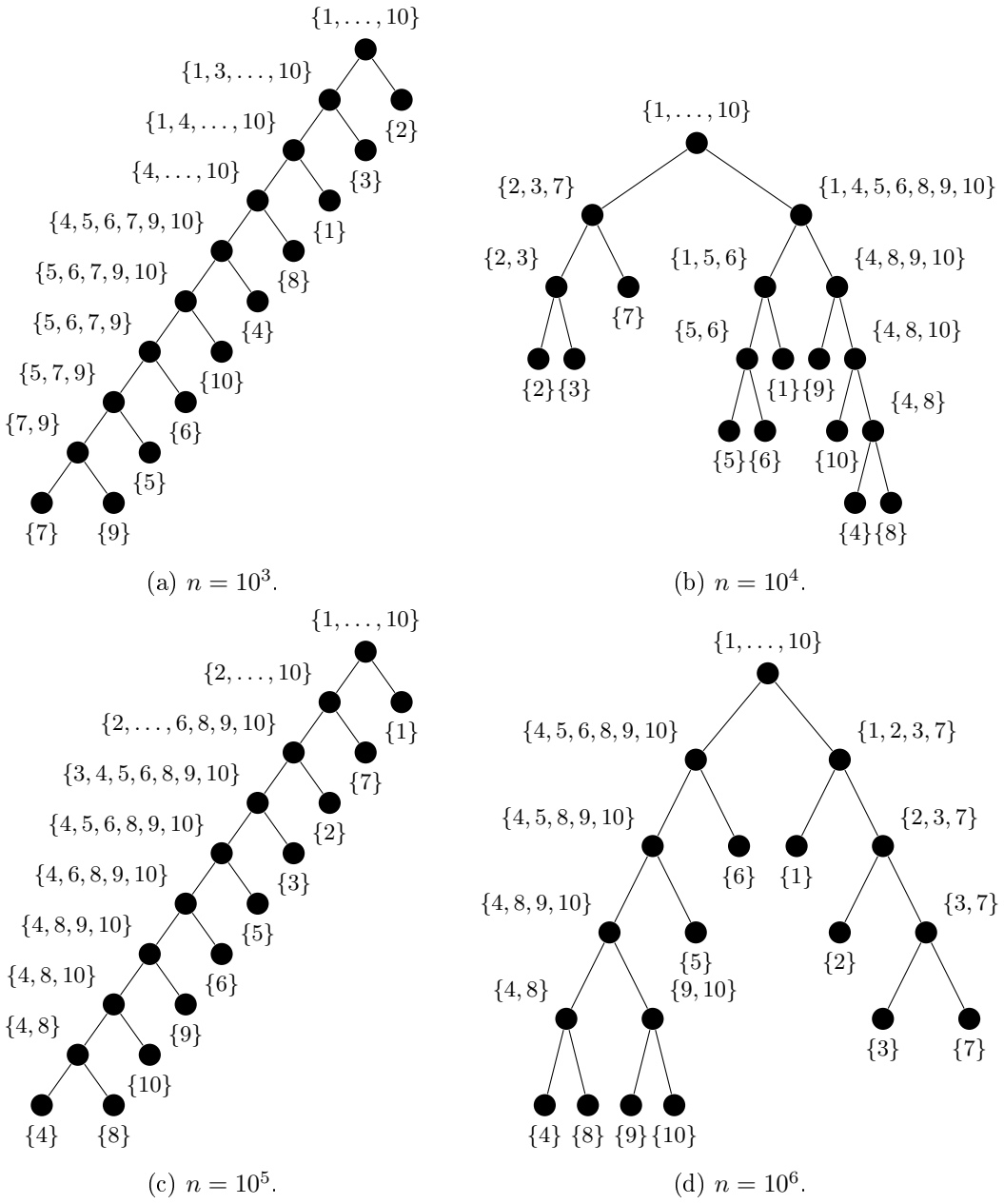


Figure 2.15: Example of graphical model.



(a) $n = 10^3$. (b) $n = 10^4$.

(c) $n = 10^5$. (d) $n = 10^6$.

Figure 2.16: Dimension trees T obtained after computing an approximation in tree-based tensor format of $(II.iii)$, using different training sample sizes n. The displayed trees are associated with the smallest error over 10 trials.

Chapter 3

Learning with tree-based tensor formats combined with changes of variables

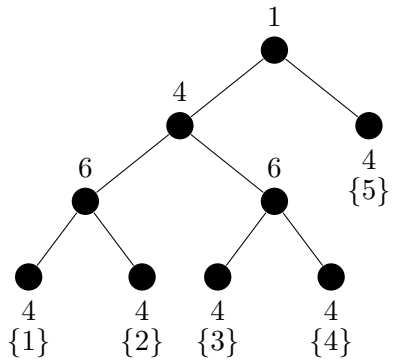
Contents

3.1 Introduction

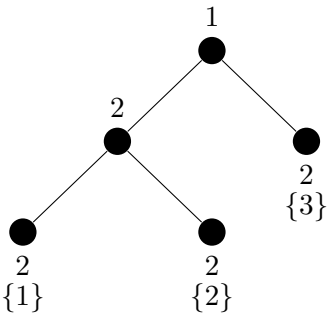
Some functions might only exhibit a low-rank structure after a suitable change of variables. A simple example is as follows: consider the function f, in dimension $d = 5$, that writes

$$f(x) = h_1(x)h_2(x) + h_2(x)^2 h_3(x), \tag{3.1}$$

with $h_i(x) = w_i^\intercal x$ and $w_i \in \mathbb{R}^d$ with components randomly drawn in $[-1, 1]$. It admits a representation in $\mathcal{T}_r^T(\mathcal{H})$ with the tree and ranks represented in Figure 3.1a and a storage complexity equal to 432. Introducing the changes of variables $z_i = h_i(x)$, $i = 1, 2, 3$, the function $f(x) = v(z) = z_1 z_2 + z_2^2 z_3$ admits a representation in tree-based tensor format with the tree and ranks represented in Figure 3.1b. Its complexity (taking into account the coefficients of h_1, h_2, h_3) is equal to 45, almost 10 times smaller than without the change of variables. We say that we reduced the complexity and the effective dimension of f, going from $d = 5$ to the so-called effective dimension $m = 3$.



(a) Without change of variables.



(b) With the change of variables $z_i = h_i(x)$, $i = 1, 2, 3$.

Figure 3.1: Dimension trees used for the representation of Function (3.1) with and without change of variables. The obtained α-ranks are indicated at each node and the dimensions associated with the leaf nodes are displayed in brackets.

In this chapter, we seek to construct approximations of the form $g = v \circ h$, with h a mapping from $\mathbb{R}^d$ to $\mathbb{R}^m$ (that operates the change of variables) chosen in a given function space, and v a m-dimensional function belonging to the model class of functions in tree-based tensor format.

In the example above, one could introduce the changes of variables $z_1' = h_1(x)h_2(x)$ and $z_2' = h_2(x)^2 h_3(x)$, yielding another representation of f with effective dimension 2, with a different complexity. One could also consider the change of variables $z_1'' = h_1(x)h_2(x) + h_2(x)^2 h_3(x)$, with $v = \mathrm{id}$, an effective dimension equal to 1 and again another complexity. This simple illustration shows that a key goal is to find an optimal effective dimension m and approximations v and h (in suitable model classes and with a given accuracy) in the sense of the storage complexity. In a learning framework, with a given training sample of finite size, this translates into finding the value of m and approximations v and h (in some model classes) yielding a good trade-off between estimation and approximation errors.

In this chapter, we propose algorithms that compute, in a statistical learning framework, a sequence of approximations $g^m = v^m \circ h^m$ with increasing effective dimension $m = 1, \ldots, m_{\max}$ with, for each m, adaptation of both the dimension tree and tree-based rank using the learning algorithms proposed in Chapter 2. Then, the approximation $g^{m_{\text{opt}}}$, $1 \leq m_{\text{opt}} \leq m_{\max}$, yielding the smallest risk (estimated on a sample independent from the training sample) is retained.

The outline of this chapter is as follows: Section 3.2 presents the considered model class of functions in tree-based tensor format combined with changes of variables, Section 3.3 introduces the proposed algorithms for learning with such model classes and finally, Section 3.4 demonstrates their performances on several numerical experiments in least-squares regression.

3.2 Tree-based tensor formats combined with changes of variables

We consider $X = (X_1, \ldots, X_d)$, a set of independent random variables, with X_ν with values in $\mathcal{X}_\nu$ and with probability law μ_ν, $\nu = 1, \ldots, d$. The random variable X is with values in $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$ and with probability law $\mu = \mu_1 \otimes \cdots \otimes \mu_d$.

We consider a mapping $h \in H^m = \mathbb{R}^m \otimes H$ from $\mathbb{R}^d$ to $\mathbb{R}^m$, with H a finite-dimensional space of functions defined on $\mathcal{X}$. It is such that $h(x) = (h_1(x), \ldots, h_m(x))$ with, for $\nu \in M = \{1, \ldots, m\}$, $h_\nu \in H$.

We denote by U the random variable obtained after applying the mapping h to X:

$$U = (U_1, \ldots, U_m) = h(X),$$

with values in $\mathcal{U} = \mathcal{U}_1 \times \cdots \times \mathcal{U}_m$ and with probability law denoted by $h_\# \mu$, the push-forward measure of μ.

We then have $U_\nu = h_\nu(X)$ with probability law $h_{\nu\#}\mu$, $\nu \in M$. We introduce the product measure $h_\otimes \mu = h_{1\#}\mu \otimes \cdots \otimes h_{m\#}\mu$ which is, in general, such that $h_\# \mu \neq h_\otimes \mu$ (the push-forward measure of μ is in general not a product measure).

Remark 3.2.1 (Estimation of the push-forward measure $h_{\nu\#}\mu$, $\nu \in M$)**.** *In practice, for $\nu \in M$, the density of the push-forward measure $h_{\nu\#}\mu$ can be estimated from a sample $\{h_\nu(x_k)\}_{k=1}^n$, using a Gaussian kernel density estimator with selection of the bandwidth according to Scott's rule [44] (for more information, see Appendix B).*

For $\nu \in M$, let V_ν be a finite dimensional subspace of $L^2_{h_{\nu\#}\mu}(\mathcal{U}_\nu)$, equipped with the norm $\|v^\nu\|^2_{L^2_{h_{\nu\#}\mu}(\mathcal{U}_\nu)} = \mathbb{E}(v^\nu(U_\nu)^2)$. Then, $V^m = V_1 \otimes \cdots \otimes V_m$ is a subspace of $L^2_{h_\otimes \mu}(\mathcal{U})$, equipped with the norm $\|v\|^2_{L^2_{h_\otimes \mu}(\mathcal{U})} = \mathbb{E}(v(U)^2)$. Let $\{\phi^\nu_i : i \in I^\nu\}$ be an orthonormal basis of V_ν, and $N_\nu = \dim(V_\nu) = \#I^\nu$. For a multi-index $i = (i_1, \ldots, i_d) \in I = I^1 \times \cdots \times I^d$, we let

$\phi_i = \phi^1_{i_1} \otimes \cdots \otimes \phi^d_{i_d}$. The set of functions $\{\phi_i : i \in I\}$ constitutes an orthonormal basis of $V^m \subset L^2_{h\otimes\mu}(\mathcal{U})$.

Given a dimension tree $T \subset 2^M$ and a tree-based rank $r = (r_\alpha)_{\alpha\in T}$, we denote by

$$\mathcal{G}(H, V^m, T, r) = \left\{v \circ h \colon h \in H^m, v \in \mathcal{T}^T_r(V^m)\right\}$$

the set of functions in tree-based tensor format combined with changes of variables.

Proposition 3.2.2. *If* $\mathrm{d}h_{\#}\mu(u) = p(u)\mathrm{d}h_{1\#}\mu(u_1)\cdots\mathrm{d}h_{m\#}\mu(u_m)$, *with* $\sup_{u\in\mathcal{U}} p(u) < \infty$, *then* $\mathcal{G}(H, V^m, T, r) \subset L^2_\mu(\mathcal{X})$.

Proof. For $g = v \circ h \in \mathcal{G}(H, V^m, T, r)$, we have

$$\begin{aligned}
\|v \circ h\|^2_{L^2_\mu(\mathcal{X})} &= \int_{\mathcal{X}} v(h(x))^2 d\mu(x) = \int_{\mathcal{U}} v(u)^2 dh_{\#}\mu(u) \\
&= \int_{\mathcal{U}} v(u)^2 p(u) dh_{1\#}\mu(u_1)\cdots\mathrm{d}h_{m\#}\mu(u_m) \\
&\leq \sup_{u\in\mathcal{U}} p(u) \int_{\mathcal{U}} v(u)^2 \mathrm{d}h_{1\#}\mu(u_1)\cdots\mathrm{d}h_{m\#}\mu(u_m) \\
&= \sup_{u\in\mathcal{U}} p(u) \|v\|^2_{L^2_{h\otimes\mu}(\mathcal{U})},
\end{aligned}$$

hence, if $\sup_{u\in\mathcal{U}} p(u) < \infty$, $v \in L^2_{h\otimes\mu}(\mathcal{U})$ implies $v \circ h \in L^2_\mu(\mathcal{X})$. By definition, $v \in \mathcal{T}^T_r(V^m) \subset L^2_{h\otimes\mu}(\mathcal{U})$, hence $\mathcal{G}(H, V^m, T, r) \subset L^2_\mu(\mathcal{X})$. □

Also, if $\mathcal{X}$ is bounded and if h and v are continuous, or if $h \in L^\infty_\mu(\mathcal{X})$ and v is continuous, we have $v \circ h \in L^\infty_\mu(\mathcal{X}) \subset L^2_\mu(\mathcal{X})$.

In the sequel, we assume that $\mathcal{G}(H, V^m, T, r)$ is a subset of $L^2_\mu(\mathcal{X})$.

3.2.1 Representation of functions in $\mathcal{G}(H, V^m, T, r)$

Let $\{\varphi_j\}_{j=1}^q$ be a basis of H, orthonormal with respect to the measure μ of X. For $\nu \in M$, the change of variables $h_\nu(x)$ writes

$$h_\nu(x) = \sum_{j=1}^q W^\nu_j \varphi_j(x),$$

with parameters $W^\nu = (W^\nu_j)_{j=1}^q$.

A function $g \in \mathcal{G}(H, V^m, T, r)$ then admits the representation

$$g(x) = \sum_{\substack{i_\nu \in I^\nu \\ \nu\in\mathcal{L}(T)}} \sum_{\substack{1\leq k_\beta \leq r_\beta \\ \beta\in T}} \prod_{\alpha\in T\setminus\mathcal{L}(T)} C^\alpha_{(k_\beta)_{\beta\in S(\alpha)},k_\alpha} \prod_{\alpha\in\mathcal{L}(T)} C^\alpha_{i_\alpha,k_\alpha} \phi^\alpha_{i_\alpha}\left(\sum_{j=1}^q W^\alpha_j \varphi_j(x)\right),$$

with parameters $(C^\alpha)_{\alpha\in T}$ and $(W^\alpha)_{\alpha\in\mathcal{L}(T)}$. We recall that $\mathcal{L}(T)$ denotes the leaves of T.

For $\alpha \in T$, we redefine the linear partial map (1.4) for functions $g \in \mathcal{G}(H, V^m, T, r)$:

$$\begin{aligned} g(x) &= \Psi^\alpha(h(x))(C^\alpha) \\ &= \sum_{1 \le k_\alpha \le r_\alpha} \sum_{i_\alpha \in I^\alpha} C^\alpha_{i_\alpha, k_\alpha} \phi^\alpha_{i_\alpha}(h_\alpha(x)) \tilde{f}^\alpha_{k_\alpha}(h_{\alpha^c}(x)), \end{aligned}$$

with the functions $\phi^\alpha_{i_\alpha}$ and $\tilde{f}^\alpha_{k_\alpha}$ introduced in Section 1.4, and with $h_\alpha(x) = (h_\nu(x))_{\nu \in \alpha}$.

Denoting by W the parameters $(W^\alpha)_{\alpha \in \mathcal{L}(T)}$, we also introduce the nonlinear partial map $\Psi^h(x)$, defined by

$$\begin{aligned} g(x) &= \Psi^h(x)(W) \\ &= \sum_{\substack{i_\nu \in I^\nu \\ \nu \in \mathcal{L}(T)}} v_{i_1, \dots, i_m} \prod_{\alpha \in \mathcal{L}(T)} \phi^\alpha_{i_\alpha} \left(\sum_{j=1}^{q} W^\alpha_j \varphi_j(x) \right), \end{aligned}$$

with $v_{i_1,\dots,i_m}$ the components of a tensor $\boldsymbol{v} \in \mathcal{T}_r^T(\mathbb{R}^I)$ that write

$$v_{i_1,\dots,i_m} = \sum_{\substack{1 \le k_\beta \le r_\beta \\ \beta \in T}} \prod_{\alpha \in T \setminus \mathcal{L}(T)} C^\alpha_{(k_\beta)_{\beta \in S(\alpha)}, k_\alpha} \prod_{\alpha \in \mathcal{L}(T)} C^\alpha_{i_\alpha, k_\alpha}.$$

3.2.2 About the representation with orthogonality conditions

If the push-forward measure $h_\# \mu$ is not a product measure, the functions $\{\phi_i\}_{i \in I}$ are not orthonormal with respect to $h_\# \mu$. Hence, for a function $g = v \circ h \in \mathcal{G}(H, V^m, T, r)$, $\|v\|_{L^2_{h_\# \mu}(\mathcal{U})} \neq \|\boldsymbol{v}\|$. Instead, the equality $\|v\|_{L^2_{h_\otimes \mu}(\mathcal{U})} = \|\boldsymbol{v}\|$ holds, involving the product measure $h_\otimes \mu$.

Proposition 3.2.3. *If* $\mathrm{d}h_\# \mu(u) = p(u) \mathrm{d}h_\otimes \mu(u)$ *with* $0 < c \le p(u) \le C < \infty$, *then the norms* $\|\cdot\|_{L^2_{h_\# \mu}(\mathcal{U})}$ *and* $\|\cdot\|_{L^2_{h_\otimes \mu}(\mathcal{U})}$ *are equivalent.*

Proof. First, $p(u) \le C < \infty$ yields, for $g = v \circ h \in \mathcal{G}(H, V^m, T, r)$,

$$\begin{aligned} \|v\|^2_{L^2_{h_\# \mu}(\mathcal{U})} &= \int_{\mathcal{U}} v(u)^2 dh_\# \mu(u) = \int_{\mathcal{U}} v(u)^2 p(u) dh_{1\#} \mu(u_1) \cdots \mathrm{d}h_{m\#} \mu(u_m) \\ &\le C \int_{\mathcal{U}} v(u)^2 \mathrm{d}h_{1\#} \mu(u_1) \cdots \mathrm{d}h_{m\#} \mu(u_m) = C \|v\|^2_{L^2_{h_\otimes \mu}(\mathcal{U})}. \end{aligned}$$

Similarly, $0 < c \le p(u)$ yields

$$\begin{aligned} \|v\|^2_{L^2_{h_\# \mu}(\mathcal{U})} &= \int_{\mathcal{U}} v(u)^2 \mathrm{d}h_\# \mu(u) = \int_{\mathcal{U}} v(u)^2 p(u) \mathrm{d}h_{1\#} \mu(u_1) \cdots \mathrm{d}h_{m\#} \mu(u_m) \\ &\ge c \int_{\mathcal{U}} v(u)^2 \mathrm{d}h_{1\#} \mu(u_1) \cdots \mathrm{d}h_{m\#} \mu(u_m) = c \|v\|^2_{L^2_{h_\otimes \mu}(\mathcal{U})}. \end{aligned}$$

Hence $c\|v\|^2_{L^2_{h_\otimes \mu}(\mathcal{U})} \le \|v\|^2_{L^2_{h_\# \mu}(\mathcal{U})} \le C\|v\|^2_{L^2_{h_\otimes \mu}(\mathcal{U})}$, which ends the proof. □

Proposition 3.2.4. *If* $\mathrm{d}h_{\#}\mu(u) = p(u)\mathrm{d}h_{\otimes}\mu(u)$ *with* $p(u) \leq C < \infty$*, and if the functions* $\{\phi_i\}_{i\in I}$ *are linearly independent on* $\mathrm{supp}(h_{\#}\mu)$ *(the support of* $h_{\#}\mu$*) and are orthonormal with respect to* $h_{\otimes}\mu$*, then the norms* $\|\cdot\|_{L^2_{h_{\#}\mu}(\mathcal{U})}$ *and* $\|\cdot\|_{L^2_{h_{\otimes}\mu}(\mathcal{U})}$ *are equivalent.*

Proof. Similarly to the proof of Proposition 3.2.3, $p(u) \leq C < \infty$ gives $\|v\|^2_{L^2_{h_{\#}\mu}(\mathcal{U})} \leq C\|v\|^2_{L^2_{h_{\otimes}\mu}(\mathcal{U})}$ for $g = v \circ h \in \mathcal{G}(H, V^m, T, r)$. Then, if the functions $\{\phi_i\}_{i\in I}$ are linearly independent on $\mathrm{supp}(h_{\#}\mu)$, the Gram matrix G of the basis is positive definite: $\lambda_{\min}(G) > 0$. This implies

$$\begin{aligned}
\|v\|^2_{L^2_{h_{\#}\mu}(\mathcal{U})} &= \sum_{i\in I}\sum_{j\in I} v_{i_1,\ldots,i_m} v_{j_1,\ldots,j_m} \overbrace{\int_{\mathrm{supp}(h_{\#}\mu)} \phi_i(u)\phi_j(u)\mathrm{d}h_{\#}\mu(u)}^{G_{ij}} \\
&= \sum_{i\in I}\sum_{j\in I} v_{i_1,\ldots,i_m} v_{j_1,\ldots,j_m} G_{ij} \\
&\geq \lambda_{\min}(G) \sum_{i\in I} v^2_{i_1,\ldots,i_m} = \lambda_{\min}(G)\|v\|^2_{L^2_{h_{\otimes}\mu}(\mathcal{U})},
\end{aligned}$$

the last equality being true because the functions $\{\phi_i\}_{i\in I}$ form an orthonormal basis in $L^2_{h_{\otimes}\mu}(\mathcal{U})$. Hence, $c'\|v\|^2_{L^2_{h_{\otimes}\mu}(\mathcal{U})} \leq \|v\|^2_{h_{\#}\mu(\mathcal{U})} \leq C\|v\|^2_{L^2_{h_{\otimes}\mu}(\mathcal{U})}$ with $c' = \lambda_{\min}(G) > 0$, which concludes the proof. □

Remark 3.2.5. *The results of Proposition 3.2.4 can be extended to the case where the functions* $\{\phi_i\}_{i\in I}$ *are orthonormal with respect to* $h_{\otimes}\mu$ *and linearly independent on a subset* $B \subset \mathcal{U}$ *such that* $\inf_{u\in B} p(u) > 0$*. Indeed, for* $g = v \circ h \in \mathcal{G}(H, V^m, T, r)$*,*

$$\begin{aligned}
\|v\|^2_{h_{\#}\mu(\mathcal{U})} &= \int_{\mathcal{U}} v(u)^2 \mathrm{d}h_{\#}\mu(u) \\
&\geq \int_B \left(\sum_{i\in I} v_{i_1,\ldots,i_m}\phi_i(u)\right)^2 \mathrm{d}h_{\#}\mu(u) \\
&\geq \lambda_{\min}(G') \sum_{i\in I} v^2_{i_1,\ldots,i_m} = c''\|v\|^2_{L^2_{h_{\otimes}\mu}(\mathcal{U})}
\end{aligned}$$

with $c'' = \lambda_{\min}(G') > 0$ *and* G' *the Gram matrix with components*

$$G'_{ij} = \int_B \phi_i(u)\phi_j(u)\mathrm{d}h_{\#}\mu(u).$$

The case $B = \mathrm{supp}(h_{\#}\mu)$ *corresponds to Proposition 3.2.4.*

These results are important for the tree adaptation which involves truncation, whose accuracy is in practice controlled by $\|v - \tilde{v}\|_{L^2_{h_{\otimes}\mu}(\mathcal{U})}$, with $\tilde{v}$ an approximation of v. They are also important in the rank adaptation procedure, where the heuristic criterion is based on the truncation error in $L^2_{h_{\otimes}\mu}(\mathcal{U})$-norm.

In the following, we will assume that we meet the requirements of Remark 3.2.5, which means that we allow $p(u)$ to be equal to 0 on a subset of $\mathcal{U}$, but there exists $B \subset U$ such that $\inf_{u \in B} p(u) > 0$ and on which the functions $\{\phi_i\}_{i \in I}$ are linearly independent.

3.3 Learning with tree-based tensor formats combined with changes of variables

In this section, we present algorithms to approximate a function f by a function $g \in \mathcal{G}(H, V^m, T, r)$, where the dimension tree T, the tree-based rank r and the effective dimension m are adapted. We begin by presenting how to compute the coefficients $(C^\alpha)_{\alpha \in T}$ of the tree-based tensor and W of the change of variables, then we describe how to construct an approximation in $\mathcal{G}(H, V^m, T, r)$ for given T, r and m and finally, we introduce the adaptive algorithm mentioned above.

3.3.1 Learning the parameters $(C^\alpha)_{\alpha \in T}$ with fixed W

For a fixed change of variables h (fixed W), the empirical risk minimization problem to solve writes

$$\min_{(C^\alpha)_{\alpha \in T}} \frac{1}{n} \sum_{k=1}^{n} \gamma(\Psi(h(\cdot))((C^\alpha)_{\alpha \in T}), z_k),$$

which is solved with Algorithm 6 in Chapter 2, where the points $\{x_k\}_{k=1}^n$ of the training sample S are replaced by $\{h(x_k)\}_{k=1}^n$. As in Chapter 2, sparsity can be sought in the parameters $(C_\alpha)_{\alpha \in T}$.

3.3.2 Learning the parameter W with fixed $(C^\alpha)_{\alpha \in T}$

For fixed node tensors $(C^\alpha)_{\alpha \in T}$, the empirical risk minimization problem to solve writes

$$\min_{W \in \mathbb{R}^{m \times q}} \frac{1}{n} \sum_{k=1}^{n} \gamma(\Psi^h(\cdot)(W), z_k). \tag{3.2}$$

In practice, given a current value for W, we compute an update by performing one iteration of a nonlinear optimization algorithm, for instance a Gauss-Newton algorithm in least-squares regression, or a gradient descent in density estimation.

Once updated, each W^α is normalized so that $\sum_{j=1}^{q} (W_j^\alpha)^2 = 1$, and the basis $\{\phi_{i_\alpha}^\alpha\}_{i_\alpha \in I^\alpha}$ is constructed to be orthonormal with respect to the measure $h_{\alpha \#} \mu$, $\alpha \in \mathcal{L}(T)$, estimated from the sample $\{h_{\alpha \#} \mu(x_k)\}_{k=1}^n$ (see Appendix B).

Remark 3.3.1 (Least-squares regression). *When using the contrast function $\gamma(g, (x, y)) = (y - g(x))^2$, we perform one iteration of a Gauss-Newton algorithm, which involves the Jacobian matrix $J = (J^1, \ldots, J^m) \in \mathbb{R}^{n \times mq}$. For $g \in \mathcal{G}(H, V^m, T, r)$, the components of*

each $J^\nu \in \mathbb{R}^{n\times q}$, $\nu \in \mathcal{L}(T)$, *write*

$$J^\nu_{kj} = \frac{\partial g}{\partial W^\nu_j}(x_k) = \sum_{\substack{i_\mu \in I^\mu \\ \mu\in\mathcal{L}(T)}} \sum_{\substack{1\le k_\gamma \le r_\gamma \\ \gamma\in T}} \prod_{\alpha\in T\setminus\mathcal{L}(T)} C^\alpha_{(k_\beta)_{\beta\in S(\alpha)},k_\alpha} \prod_{\beta\in\mathcal{L}(T)\setminus\{\nu\}} C^\beta_{i_\beta,k_\beta}\phi^\beta_{i_\beta}(h_\beta(x_k)) C^\nu_{i_\nu,k_\nu}(\phi^\nu_{i_\nu})'(h_\nu(x_k))\varphi_j(x_k),$$

for $k = 1,\dots,n$ *and* $j = 1,\dots,q$, *and with* $(\phi^\nu_{i_\nu})'(h_\nu(x_k))$ *the derivative of the (univariate) function* $\phi^\nu_{i_\nu}$ *evaluated at the point* $h_\nu(x_k)$. *Given a current value for* W, *an update* ΔW *is computed by solving the linear regression problem*

$$\min_{\Delta W\in\mathbb{R}^{m\times q}} \|J\,\mathrm{vec}(\Delta W) - r\|_2^2$$

with $\mathrm{vec}(W) \in \mathbb{R}^{mq}$ *the vectorization of* W *and* $r \in \mathbb{R}^n$ *the residual, such that* $r_k = y_k - g(x_k)$, $k = 1,\dots,n$. *The next iterate then writes* $W + \alpha\Delta W$, *with* α *selected with a golden section line search over* $[0,1]$.

Remark 3.3.2 (Density estimation with a quadratic contrast function)**.** *When using the contrast function* $\gamma(g,x) = \|g\|^2 - 2g(x)$, *the problem to solve writes*

$$\min_{W\in\mathbb{R}^{m\times q}} \|\Psi^h(\cdot)(W)\|^2_{L^2_\mu(\mathcal{X})} - \frac{2}{n}\sum_{k=1}^n \Psi^h(x_k)(W).$$

The gradient descent algorithm involves the gradient of the functional to minimize

$$G = 2\int_{\mathcal{X}} \Psi^h(x)(W)\frac{\partial}{\partial\,\mathrm{vec}(W)}\Psi^h(x)(W)\mathrm{d}\mu(x) - \frac{2}{n}\sum_{k=1}^n \frac{\partial}{\partial\,\mathrm{vec}(W)}\Psi^h(x_k)(W),$$

with $\mathrm{vec}(W) \in \mathbb{R}^{mp}$ *the vectorization of* W *and*

$$\frac{\partial}{\partial\,\mathrm{vec}(W)}\Psi^h(x_k)(W) = (J^1_{k1},\dots,J^1_{kp},\dots,J^m_{k1},\dots,J^m_{kp})$$

with J^α_{kj} *given in Remark 3.3.1,* $\alpha \in \mathcal{L}(T)$, $j = 1,\dots,p$.

It may be interesting to seek sparsity in the parameters $\{W^\alpha\}_{\alpha\in\mathcal{L}(T)}$, for instance to perform input variables screening. While searching for sparsity, one must take care to retain a solution W which is so that there does not exist $\alpha \in \mathcal{L}(T)$ such that $W^\alpha = 0$, because this would mean $U^\alpha = 0$ and would lead to the degeneration of the approximation format.

3.3.3 Algorithm for learning in $\mathcal{G}(H, V^m, T, r)$

The algorithm to compute an approximation in $\mathcal{G}(H, V^m, T, r)$ is similar to Algorithm 6 to compute an approximation in $\mathcal{T}^T_r(\mathcal{H})$, with additional steps after the minimization over $(C^\alpha)_{\alpha\in T}$ in order to compute W and the bases $\{\phi^\alpha_{i_\alpha}\}_{i_\alpha\in I^\alpha}$, orthonormal with respect to $h_{\alpha\#}\mu$, $\alpha \in \mathcal{L}(T)$. Algorithm 10 presents the learning procedure.

Algorithm 10 Learning in $\mathcal{G}(H, V^m, T, r)$.

Inputs: sample $S = \{z_k\}_{k=1}^n$, contrast function γ, tensor format with parametrizations $\Psi(h(\cdot))((C^\alpha)_{\alpha\in T})$ and $\Psi^h(\cdot)(W)$, and initial values for $\{C^\alpha\}_{\alpha\in T}$ and $W = (W^\alpha)_{\alpha\in\mathcal{L}(T)}$

Outputs: approximation $g \in \mathcal{G}(H, V^m, T, r)$

1: **while** not converged **do**
2: set $h_\alpha(x) \leftarrow \sum_{j=1}^q W_j^\alpha \varphi_j(x)$, $\alpha \in \mathcal{L}(T)$
3: create the bases $\{\phi_{i_\alpha}^\alpha\}_{i_\alpha\in I^\alpha}$, orthonormal with respect to the estimated push-forward measure $h_{\alpha\#}\mu$, $\alpha \in \mathcal{L}(T)$
4: **for** $\alpha \in T$ **do**
5: estimate C^α for fixed parameters C^β, $\beta \neq \alpha$ (learning problem with a linear model)
6: **end for**
7: estimate $W = (W^\alpha)_{\alpha\in\mathcal{L}(T)}$ for fixed parameters C^α, $\alpha \in T$, by performing one iteration of a nonlinear optimization algorithm applied to (3.2)
8: normalize W^α such that $\|W^\alpha\|_2 = 1$, $\alpha \in \mathcal{L}(T)$
9: **end while**
10: set

$$g(x) \leftarrow \sum_{\substack{i_\nu\in I^\nu \\ \nu\in\mathcal{L}(T)}} \sum_{\substack{1\le k_\beta\le r_\beta \\ \beta\in T}} \prod_{\alpha\in T\setminus\mathcal{L}(T)} C^\alpha_{(k_\beta)_{\beta\in S(\alpha)},k_\alpha} \prod_{\alpha\in\mathcal{L}(T)} C^\alpha_{i_\alpha,k_\alpha}\phi^\alpha_{i_\alpha}\left(\sum_{j=1}^q W_j^\alpha\varphi_j(x)\right)$$

3.3.4 Effective dimension adaptation algorithm

We propose an algorithm that computes an approximation in $\mathcal{G}(H, V^m, T, r)$, with adaptation of T, r and m, the effective dimension. The idea of this algorithm is as follows: starting from 1, incrementally increase m, each time computing an approximation in tree-based tensor format combined with changes of variables from $\mathbb{R}^d$ to $\mathbb{R}^m$, until a maximum dimension $m_{\max}$ is reached.

At a step m, given a current approximation g^m with m effective variables $U_1, \ldots, U_m$, a new variable U_{m+1} is added by computing a correction of g^m that writes

$$c^{m+1}(x) = \sum_{i\in I^{m+1}} a_i \phi_i^{m+1}\left(\sum_{j=1}^q W_j^{m+1}\varphi_j(x)\right),$$

and by defining

$$U_{m+1} = h_{m+1}(X) = \sum_{j=1}^q W_j^{m+1}\varphi_j(X).$$

In least-squares regression, this correction corresponds to a one-term projection pursuit regression model [16]. Its computation is described in Algorithm 11: after randomly initializing the coefficients W^{m+1} of the new variable U_{m+1}, the correction $c^{m+1}(x)$ is computed by alternatively minimizing on $(a_i)_{i\in I^{m+1}}$ (learning problem with a linear model) and performing

one iteration of a nonlinear optimization algorithm applied to the problem of minimization on $(W_j^{m+1})_{j=1}^q$, for instance a gradient descent algorithm, or a Gauss-Newton algorithm in least-squares regression.

Algorithm 11 Computation of a correction c^{m+1} of an approximation g^m with m effective variables.

Inputs: sample $S = \{z_k\}_{k=1}^n$, contrast function γ, function g^m of m variables, basis $\{\varphi_j\}_{j=1}^q$, construction method for the basis $\{\phi_i^{m+1}\}_{i\in I^{m+1}}$

Outputs: corrected approximation $g^m + c^{m+1}$, new variable $U_{m+1} = h_{m+1}(X)$

1: randomly initialize W^{m+1} such that $\|W^{m+1}\|_2 = 1$, set $h_{m+1} \leftarrow \sum_{j=1}^q W_j^{m+1}\varphi_j$ and $U_{m+1} = h_{m+1}(X)$

2: **while** not converged **do**

3: create the basis $\{\phi_i^{m+1}\}_{i\in I^{m+1}}$, orthonormal with respect to the estimated measure $h_{(m+1)\#}\mu$ of U_{m+1}

4: solve the problem

$$\min_{a\in\mathbb{R}^{N_{m+1}}} \frac{1}{n}\sum_{k=1}^n \gamma(g(\cdot) + \sum_{i\in I^{m+1}} a_i\phi_i^{m+1}(h_{m+1}(\cdot)), z_k)$$

5: perform one iteration of a nonlinear optimization algorithm applied to

$$\min_{W^{m+1}\in\mathbb{R}^q} \frac{1}{n}\sum_{k=1}^n \gamma(g(\cdot) + \sum_{i\in I^{m+1}} a_i\phi_i^{m+1}(\sum_{j=1}^q W_j^{m+1}\varphi_j(\cdot)), z_k)$$

6: set $h_{m+1} \leftarrow \sum_{j=1}^q W_j^{m+1}\varphi_j$ and $U_{m+1} = h_{m+1}(X)$

7: **end while**

8: set $c^{m+1}(\cdot) \leftarrow \sum_{i\in I^{m+1}} a_i\phi_i^{m+1}(h^{m+1}(\cdot))$

Algorithm 12 summarizes the adaptive procedure: starting from 1, it increases the effective dimension m up to $m_{\max}$, each time computing an approximation in $\mathcal{G}(\mathcal{H}, V^m, T^m, r^m)$, with adapted dimension tree T^m and tree-based rank r^m. The selection of the best approximation $g^{m_{\text{opt}}}$ is performed using a validation sample independent of the training sample S.

Remark 3.3.3 (Rank adaptation)**.** *At Step 6 of Algorithm 12, the approximations computed when performing rank adaptation are done without optimizing W.*

3.4 Numerical experiments

We now consider several numerical experiments that aim at illustrating the performances of the proposed algorithms.

Algorithm 12 Learning in $\mathcal{G}(H, V^m, T, r)$ with adaptation of T, r and m.

Inputs: sample $S = \{z_k\}_{k=1}^n$, contrast function γ, maximal number of effective variables $m_{\max}$, basis $\{\varphi_j\}_{j=1}^q$, construction method for the bases $\{\phi_{i_m}^m\}_{i_m \in I^m}$, $m = 1, \ldots, m_{\max}$

Outputs: approximation $g \in \mathcal{G}(H, V^m, T, r)$ with adapted T, r and m

set $g^0 \leftarrow 0$
add a new variable $U_1 = h_1(X)$ and compute a corrected approximation $g^1 = g^0 + c^1$ using Algorithm 11
add a new variable $U_2 = h_2(X)$ and compute a corrected approximation $g^2 = g^1 + c^2$ using Algorithm 11
set $T_{\text{init}}^2 \leftarrow \{\{1,2\},\{1\},\{2\}\}$
for $m = 2, \ldots, m_{\max}$ **do**
    compute an approximation $g^m = v^m \circ h^m \in \mathcal{G}(H, V^m, T^m, r^m)$ with adapted T^m, r^m and change of variables $h^m = (h_1, \ldots, h_m)$ using Algorithm 9 of Chapter 2 (with T_{init}^m as initial tree), where Algorithm 6 in Step 9 is replaced by Algorithm 10
    if $m < m_{\max}$ **then**
        add a new variable U_{m+1} using Algorithm 11
        set $T_{\text{init}}^{m+1} \leftarrow T$
        in T_{init}^{m+1}, replace $\{m\}$ by $\{m, m+1\}$ with children $S(\{m, m+1\}) = \{\{m\},\{m+1\}\}$
    end if
end for
select $m_{\text{opt}} = \text{argmin}_{1 \le m \le m_{\max}} \mathcal{R}_{\tilde{S}}(g^m)$, with $\tilde{S}$ a validation set independent of S
set $m \leftarrow m_{\text{opt}}$, $H \leftarrow H^m$, $T \leftarrow T^m$, $r \leftarrow r^m$ and $g \leftarrow g^m$

Contrast function. We consider a least-squares regression setting by choosing the contrast function $\gamma(g,(x,y)) = (y - g(x))^2$. Then, we use one step of a Gauss-Newton algorithm to compute the coefficients W of the changes of variables.

Approximation spaces. For the spaces V_ν, $\nu = 1, \ldots, m_{\max}$, we choose polynomial spaces $\mathbb{P}_p(\mathcal{U}_\nu)$ of degree p and we use orthonormal polynomial bases $\{\phi_{i_\nu}^\nu\}_{i_\nu \in I^\nu}$ in $L^2_{h_{\nu\#}\mu}(\mathcal{U}_\nu)$, with $I^\nu = \{0, \ldots, p\}$. For a given tree T^m with m leaves, we exploit sparsity in the leaf tensors $(C^\alpha)_{\alpha \in \mathcal{L}(T^m)}$ by using a working set strategy. We use the natural sequence of candidate patterns associated with spaces of polynomials with increasing degree.

For the change of variables h, we choose a basis $\{\varphi_i\}_{i=1}^q$ of multivariate polynomials with total degree bounded by Q, from which we remove the constant function, so that the number of basis functions is $q = (Q+d)!/(Q!d!) - 1$. This basis is taken orthonormal with respect to the measure μ of X.

Parameters of Algorithm (12). We set $N = 10^3$ in Algorithm 5. We also let the maximum number of iterations in Algorithm 9 be sufficiently high. In Algorithm 10, the stopping criterion is the stagnation of g in $L^2_{h\#\mu}(\mathcal{U})$ norm, with a tolerance equal to 10^{-6}

and a maximal number of iterations of 100; in Algorithm 11, the stopping criterion is the stagnation in 2-norm of both W and a, with a tolerance set to 10^{-10} and a maximal number of iterations of 100; finally, in Algorithm 12, $m_{\max}$ is set to 10.

Error measures. The relative test error $\varepsilon(S_{\text{test}}, g)$ associated with the function g and a test sample S_{test} of size 10^5, independent of S, is defined as

$$\varepsilon(S_{\text{test}}, g) = \sqrt{\frac{\sum_{(x,y)\in S_{\text{test}}} (y - g(x))^2}{\sum_{(x,y)\in S_{\text{test}}} y^2}}.$$

Robustness study. For studying the robustness of the proposed algorithm, we run it 10 times for each example, each run using a different training sample S.

Storage complexity. In order to take into account the changes of variables into the storage complexity, we redefine it as

$$\widetilde{C}(V^m, T, r) = C(T, r) + qm,$$

the storage complexity of the tree-based tensor, as defined in Chapter 1, to which we add the number of entries in the matrix W.

3.4.1 Simple function in high dimension

We first study the function

$$f(X) = \frac{1}{(10 + U_1 U_3 + 0.5 U_2)^2}, \tag{III.i}$$

with $U_i = w_i^\intercal X$, $w_i \in \mathbb{R}^d$ with components drawn randomly in $[-1, 1]$ such that $\|w_i\|_2 = 1$, $i = 1, 2, 3$, and with $d = 100$. The random variables $X_1, \ldots, X_d$ are independent and such that $X_i \sim \mathcal{N}(0, 1)$, $i = 1, \ldots, d$, so that $X = (X_1, \ldots, X_d)$ is with values in $\mathcal{X} = \mathbb{R}^d$.

We choose a polynomial degree $p = 10$ for the bases $\{\phi^\nu_{i_\nu}\}_{i_\nu \in I^\nu}$, $\nu = 1, \ldots, m_{\max}$.

Table 3.1 presents the obtained results. We first study, in Table 3.1b, the results obtained without introducing changes of variables, that is to say when approximating f in tree-based tensor format in dimension $d = 100$. We see that, because of the high dimension, the obtained approximations require a large storage complexity to be able to represent the function with a small error. On the other hand, when introducing a change of variables using Algorithm 12, we obtain the results of Table 3.1a where we see that, at an equal number of samples $n \geq 10^3$, we obtain better results, with a smaller storage complexity. We also notice that, with $n = 10^4$, the obtained approximations have on average an effective dimension $m_{\text{opt}} = 4.4$, which is much smaller than the dimension $d = 100$, but is not able to find the expected effective dimension of Function $(III.i)$, equal to 3.

n	m_{opt}	$\varepsilon(S_{\text{test}}, g)$	CV error	$\widetilde{C}(V^{m_{\text{opt}}}, T, r)$
10^2	$[1, 1.7, 5]$	$[1 \cdot 10^{-1}, 2 \cdot 10^{-0}, 1 \cdot 10^{+1}]$	$[2 \cdot 10^{-16}, 8 \cdot 10^{-2}, 1 \cdot 10^{-1}]$	$[111, 189.4, 559]$
10^3	$[3, 5.2, 10]$	$[6 \cdot 10^{-3}, 3 \cdot 10^{-2}, 1 \cdot 10^{-1}]$	$[1 \cdot 10^{-04}, 1 \cdot 10^{-2}, 1 \cdot 10^{-1}]$	$[393, 684.4, 1172]$
10^4	$[3, 4.4, 7]$	$[1 \cdot 10^{-3}, 2 \cdot 10^{-3}, 3 \cdot 10^{-3}]$	$[4 \cdot 10^{-05}, 2 \cdot 10^{-4}, 5 \cdot 10^{-4}]$	$[378, 615.6, 1063]$

(a) With changes of variables.

n	$\varepsilon(S_{\text{test}}, g)$	CV error	$C(T, r)$
10^2	$[1 \cdot 10^{-1}, 2 \cdot 10^{-1}, 2 \cdot 10^{-1}]$	$[1 \cdot 10^{-2}, 6 \cdot 10^{-2}, 1 \cdot 10^{-1}]$	$[1199, 1199.0, 1199]$
10^3	$[1 \cdot 10^{-1}, 1 \cdot 10^{-1}, 1 \cdot 10^{-1}]$	$[1 \cdot 10^{-1}, 1 \cdot 10^{-1}, 1 \cdot 10^{-1}]$	$[1199, 1213.6, 1284]$
10^4	$[3 \cdot 10^{-2}, 4 \cdot 10^{-2}, 8 \cdot 10^{-2}]$	$[1 \cdot 10^{-2}, 2 \cdot 10^{-2}, 4 \cdot 10^{-2}]$	$[3344, 3984.7, 4660]$

(b) Without changes of variables, with $p = 10$.

Table 3.1: [Minimum, mean, maximum] values over 10 trials of the obtained results for the learning of $(III.i)$, for different training samples sizes n, with and without changes of variables.

3.4.2 Borehole function

We consider an 8-dimensional function f that models the water flow through a borehole:

$$f(X) = \frac{2\pi T_u (H_u - H_l)}{\ln(r/r_w)\left(1 + \frac{2LT_u}{\ln(r/r_w) r_w^2 K_w} + \frac{T_u}{T_l}\right)}, \qquad (III.ii)$$

with $X = (r_w, r, T_u, H_u, T_l, H_l, L, K_w)$ whose components have the distributions shown in Table 3.2.

X_i	distribution	$\mathcal{X}_i$	X_i	distribution	$\mathcal{X}_i$
r_w	$\mathcal{N}(0.1, 0.0161812)$	$\mathbb{R}$	T_l	$\mathcal{U}(63.1, 116)$	$[63.1, 116]$
r	$\mathcal{LN}(7.71, 1.0056)$	$]0, +\infty[$	H_l	$\mathcal{U}(700, 820)$	$[700, 820]$
T_u	$\mathcal{U}(63070, 115600)$	$[63070, 115600]$	L	$\mathcal{U}(1120, 1680)$	$[1120, 1680]$
H_u	$\mathcal{U}(990, 1110)$	$[990, 1110]$	K_w	$\mathcal{U}(9855, 12045)$	$[9855, 12045]$

Table 3.2: Distribution and support of the input random variables of the borehole function $(III.ii)$.

For the basis $\{\varphi_i\}_{i=1}^q$ of the changes of variables, we consider maximal total degrees $Q = 1$ and $Q = 3$, and for the polynomial bases $\{\phi_{i_\alpha}^\alpha\}_{i_\alpha \in I^\alpha}$, $\alpha \in \mathcal{L}(T)$, a degree $p = 10$.

When studying the expression $(III.ii)$, no linear change of variables seems obvious, except $H_u - H_l$. With a polynomial basis with maximal total degree $Q = 3$, one could also consider the new variables $T_u(H_u - H_l)$, $r_w^2 K_w$ and LT_u. We then do not expect a large decrease from d to the effective dimension m. Table 3.3 presents the obtained results when approximating f in $\mathcal{G}(H, V^m, T, r)$ (using Algorithm 12 for Tables 3.3a with $Q = 1$ and 3.3b with $Q = 3$) and in $\mathcal{T}_r^T(\mathcal{H})$ (using Algorithm 9 for Table 3.3c).

We see that, as expected, for $Q = 1$, the error decreases when n increases, and that the cross-validation error is a good estimator of $\varepsilon(S_{\text{test}}, g)$. However, for $Q = 3$, the error increases on

n	m_{opt}	$\varepsilon(S_{\text{test}}, g)$	CV error	$\widetilde{C}(V^{m_{\text{opt}}}, T, r)$
10^2	$[4, 4.6, 6\ ]$	$[1 \cdot 10^{-3}, 2 \cdot 10^{-3}, 5 \cdot 10^{-3}]$	$[1 \cdot 10^{-4}, 7 \cdot 10^{-4}, 2 \cdot 10^{-3}]$	$[122, 204.1, 284]$
10^3	$[6, 7.7, 10]$	$[7 \cdot 10^{-5}, 2 \cdot 10^{-4}, 3 \cdot 10^{-4}]$	$[4 \cdot 10^{-5}, 1 \cdot 10^{-4}, 2 \cdot 10^{-4}]$	$[268, 426.0, 601]$
10^4	$[5, 7.4, 10]$	$[3 \cdot 10^{-5}, 1 \cdot 10^{-4}, 2 \cdot 10^{-4}]$	$[3 \cdot 10^{-5}, 7 \cdot 10^{-5}, 2 \cdot 10^{-4}]$	$[380, 540.0, 881]$

(a) With changes of variables, with $Q = 1$.

n	m_{opt}	$\varepsilon(S_{\text{test}}, g)$	CV error	$\widetilde{C}(V^{m_{\text{opt}}}, T, r)$
10^2	$[1, 4.4, 9]$	$[9 \cdot 10^{-2}, 4 \cdot 10^{-1}, 1 \cdot 10^{-0}]$	$[5 \cdot 10^{-16}, 7 \cdot 10^{-7}, 3 \cdot 10^{-6}]$	$[175, 773.4, 1583]$
10^3	$[1, 2.0, 3]$	$[2 \cdot 10^{-4}, 8 \cdot 10^{-0}, 8 \cdot 10^{+1}]$	$[6 \cdot 10^{-05}, 7 \cdot 10^{-4}, 1 \cdot 10^{-3}]$	$[175, 411.2, 670]$
10^4	$[2, 2.8, 6]$	$[5 \cdot 10^{-5}, 1 \cdot 10^{-4}, 5 \cdot 10^{-4}]$	$[2 \cdot 10^{-05}, 7 \cdot 10^{-4}, 1 \cdot 10^{-4}]$	$[376, 593.3, 1164]$

(b) With changes of variables, with $Q = 3$.

n	$\varepsilon(S_{\text{test}}, g)$	CV error	$C(T, r)$
10^2	$[1 \cdot 10^{-5}, 3 \cdot 10^{-5}, 6 \cdot 10^{-5}]$	$[1 \cdot 10^{-6}, 3 \cdot 10^{-6}, 9 \cdot 10^{-6}]$	$[199, 235.3, 290]$
10^3	$[4 \cdot 10^{-7}, 1 \cdot 10^{-6}, 4 \cdot 10^{-6}]$	$[3 \cdot 10^{-7}, 4 \cdot 10^{-7}, 1 \cdot 10^{-6}]$	$[199, 213.5, 224]$
10^4	$[4 \cdot 10^{-7}, 5 \cdot 10^{-7}, 1 \cdot 10^{-6}]$	$[3 \cdot 10^{-7}, 3 \cdot 10^{-7}, 4 \cdot 10^{-7}]$	$[214, 214.0, 214]$

(c) Without changes of variables, with $p = 10$.

Table 3.3: [Minimum, mean, maximum] values over 10 trials of the obtained results for the learning of $(III.ii)$, for different training samples sizes n, with and without changes of variables.

average between $n = 10^2$ and $n = 10^3$, before decreasing for $n = 10^4$. This can be explained by the fact that the changes of variables depend on $q = 164$ parameters, degrading the quality of the approximation when considering small training samples. Using a maximal total degree $Q = 3$ instead of 1 yields smaller values of m_{opt} and of the error, at the price of a larger storage complexity (once again due to the fact that $q = 164$ instead of 8). This shows that, when choosing the degree Q, there is a trade-off between dimension reduction (that can decrease when increasing Q) and storage complexity (that can increase when increasing Q). Introducing sparsity in W could be a way of improving the obtained results in the case of small training samples. Furthermore, it could increase the interpretability of the components of W, by performing input screening for instance.

We also see that the algorithm provides most of the time an approximation with $m_{\text{opt}} < d$, which means a reduction of the dimension, this comes however with a larger error than when approximating in $\mathcal{T}_r^T(\mathcal{H})$, that is to say without changes of variables. The effective dimension, equal on average to $m_{\text{opt}} = 2.8$ with $n = 10^4$ and $Q = 3$, shows that the algorithm is able to reduce the dimension by finding non-trivial changes of variables (as mentioned above, we expected an effective dimension not much smaller than d).

Table 3.4 presents the components of the matrix W of the approximation yielding the smallest error in Table 3.3a for $Q = 1$ and different values of n. We notice that the obtained approximation always features new variables involving $H_u - H_l$, which was expected. We also see that the variables r, T_u and T_l have a much smaller influence on $f(x)$ than the other variables, in the sense that their related coefficients in W are much smaller than the ones of

other variables. Finally, we see with $n = 10^4$, as mentioned above, the effective dimension is greater than 8, which means that the new variables are not linearly independent.

	r_w	r	T_u	H_u	T_l	H_l	L	K_w
U_1	**0.83**	−0.00	−0.00	**0.32**	0.00	**-0.32**	**-0.31**	**0.16**
U_2	−0.01	−0.02	0.01	**-0.48**	0.02	**0.47**	0.09	**0.74**
U_3	**-0.64**	0.01	−0.01	**0.43**	−0.01	**-0.43**	**0.42**	**-0.20**
U_4	**-0.59**	−0.01	−0.00	**0.38**	0.01	**-0.38**	**-0.55**	**0.22**

(a) $n = 10^2$.

	r_w	r	T_u	H_u	T_l	H_l	L	K_w
U_1	**-0.83**	0.00	0.00	**-0.31**	−0.00	**0.31**	**0.31**	**-0.15**
U_2	**0.77**	−0.00	0.00	**-0.45**	−0.00	**0.45**	0.02	0.00
U_3	**0.29**	−0.01	0.00	**0.13**	0.00	**-0.13**	**-0.94**	0.05
U_4	**-0.51**	0.01	−0.00	**-0.20**	0.01	**0.20**	**-0.10**	**0.80**
U_5	**-0.58**	**-0.22**	0.00	0.06	**0.74**	−0.05	**0.15**	**-0.17**
U_6	**0.80**	**0.14**	−0.03	**-0.16**	0.08	**0.15**	**0.33**	**-0.43**
U_7	**0.40**	**-0.79**	0.08	**0.15**	0.04	**-0.13**	**0.33**	**0.25**

(b) $n = 10^3$.

	r_w	r	T_u	H_u	T_l	H_l	L	K_w
U_1	**-0.84**	−0.01	0.00	**-0.30**	0.01	**0.30**	**0.29**	**-0.15**
U_2	**-0.96**	0.00	−0.00	**0.20**	0.00	**-0.20**	0.00	0.07
U_3	**0.10**	0.00	−0.00	**-0.70**	0.01	**0.70**	−0.08	0.03
U_4	**-0.36**	0.00	0.00	**0.10**	−0.03	**-0.10**	**-0.92**	0.00
U_5	0.06	0.01	0.02	**-0.23**	**-0.30**	**0.22**	**-0.50**	**0.75**
U_6	**-0.10**	**0.15**	0.02	**-0.34**	**-0.65**	**0.33**	**-0.37**	**0.43**
U_7	**0.80**	−0.03	−0.00	**-0.12**	0.07	**0.12**	**-0.57**	0.04
U_8	**0.17**	0.03	−0.01	**0.30**	0.01	**-0.29**	**0.35**	**-0.82**
U_9	**0.45**	**-0.12**	0.04	**-0.62**	0.02	**0.60**	**0.15**	0.01

(c) $n = 10^4$.

Table 3.4: Components in each W^i, $i \in M$, involving each variable X_j, $j \in D$, for the smallest error obtained with different sample sizes n in Table 3.3, when approximating $(III.ii)$ with $Q = 1$.

We can then conclude that, even though the algorithm is able to provide a good approximation with a reduction of the dimension, when there are no obvious changes of variables, it may yield worse results than when not introducing a change of variables. We propose several leads as to why this is the case: even though the cross-validation seems like a good estimator of the generalization error, it might not be good enough for the model selection involved in the sparsity exploitation in the leaves of the tree, maybe because of the loss of orthonormal representation (as discussed in Section 3.2.2). Also, one step of the Gauss-Newton algorithm when learning W might not be enough: this would call for an increase of the number of steps or for the search for a better solver.

3.4.3 Function of ten variables with five noninfluential variables

We consider the following function in dimension $d = 10$, originally used as a test case in [45]:

$$f(X) = 10\sin(\pi X_1 X_2) + 20(X_3 - 0.5)^2 + 10X_4 + 5X_5, \qquad (III.iii)$$

where only the five first input variables have an influence on $f(X)$. The random variables $X_1, \ldots, X_{10}$ are uniform on $[0, 1]$, so that $\mathcal{X} = [0, 1]^{10}$.

As for the previous experiment, we choose $p = 10$. When representing the changes of variables h with $Q = 1$, we expect Algorithm 12 to provide an approximation of $(III.iii)$ with $m_{\text{opt}} = 4$; indeed, it can be written

$$f(X) = \tilde{f}(U) = 10\sin\left(\frac{\pi}{4}(U_1^2 - U_2^2)\right) + 20(U_3 - 0.5)^2 + U_4$$

with $U_1 = X_1 + X_2$, $U_2 = X_1 - X_2$, $U_3 = X_3$ and $U_4 = 10X_4 + 5X_5$. Similarly, using a maximal total degree $Q = 2$ excluding the constant function (containing 65 functions), we expect Algorithm 12 to provide an approximation of $(III.iii)$ with $m_{\text{opt}} = 2$, because it can be written

$$f(X) = \tilde{\tilde{f}}(U) = 10\sin(\pi U_1) + U_2$$

with $U_1 = X_1 X_2$ and $U_2 = 20(X_3 - 0.5)^2 + 10X_4 + 5X_5$.

Table 3.5 presents the obtained results with $p = 10$ and $Q = 1, 2$, as well as without introducing changes of variables with $p = 20$. We see that, with $Q = 1$ and enough data points, the algorithm is able to find on average an effective dimension close to the expected one of 4, with a small decrease in the storage complexity, at the price of a larger error. When using $Q = 2$, we notice that the algorithm cannot achieve the error level obtained without changes of variables, and yields larger storage complexity. As in the previous example, this might be due to the use of a cross-validation estimator of the error for model selection, or to a convergence issue in the Gauss-Newton algorithm. Introducing sparsity in W might also improve the obtained results.

Table 3.6 presents the components of the matrix W of the approximation yielding the smallest error in Table 3.5 for $Q = 1$ and different values of n. We notice that the algorithm introduces a new variable that involves (up to a factor) $10X_4 + 5X_5$ as expected, but also the variables X_1 and X_2. We also notice that there is always a new variable almost equal to X_3 and that the coefficients associated with the variables X_i, $i > 5$ are close to 0, which shows that the algorithm is able to perform input variables screening. Finally, for $n = 10^4$, there are, as expected, two variables approximately proportional to $X_1 + X_2$ and $X_1 - X_2$ respectively.

n	Q	m_{opt}	$\varepsilon(S_{\text{test}}, g)$	CV error	$\widetilde{C}(V^{m_{\text{opt}}}, T, r)$
10^2	1	$[4, 5.3, 10]$	$[1 \cdot 10^{-2}, 4 \cdot 10^{-2}, 1 \cdot 10^{-1}]$	$[4 \cdot 10^{-3}, 2 \cdot 10^{-2}, 8 \cdot 10^{-2}]$	$[183, 214.3, 260]$
	2	$[1, 2.5, 5]$	$[1 \cdot 10^{-1}, 1 \cdot 10^{-1}, 1 \cdot 10^{-1}]$	$[3 \cdot 10^{-2}, 4 \cdot 10^{-2}, 5 \cdot 10^{-2}]$	$[76, 198.6, 430]$
10^3	1	$[4, 5.5, 7]$	$[2 \cdot 10^{-5}, 1 \cdot 10^{-4}, 4 \cdot 10^{-4}]$	$[1 \cdot 10^{-5}, 2 \cdot 10^{-4}, 8 \cdot 10^{-4}]$	$[180, 496.8, 778]$
	2	$[3, 3.8, 5]$	$[9 \cdot 10^{-5}, 3 \cdot 10^{-4}, 7 \cdot 10^{-4}]$	$[1 \cdot 10^{-4}, 1 \cdot 10^{-3}, 3 \cdot 10^{-3}]$	$[396, 537.7, 696]$
10^4	1	$[4, 4.4, 5]$	$[2 \cdot 10^{-7}, 5 \cdot 10^{-6}, 1 \cdot 10^{-5}]$	$[2 \cdot 10^{-7}, 5 \cdot 10^{-5}, 3 \cdot 10^{-4}]$	$[209, 496.7, 834]$
	2	$[4, 5.5, 10]$	$[4 \cdot 10^{-5}, 8 \cdot 10^{-5}, 1 \cdot 10^{-4}]$	$[4 \cdot 10^{-5}, 4 \cdot 10^{-4}, 1 \cdot 10^{-3}]$	$[726, 1066.6, 1969]$

(a) With change of variables, with $Q = 1, 2$.

n	$\varepsilon(S_{\text{test}}, g)$	CV error	$C(T, r, m_{\text{opt}})$
10^2	$[1 \cdot 10^{-4}, 2 \cdot 10^{-2}, 1 \cdot 10^{-1}]$	$[2 \cdot 10^{-5}, 1 \cdot 10^{-2}, 8 \cdot 10^{-2}]$	$[219, 385.3, 503]$
10^3	$[3 \cdot 10^{-7}, 6 \cdot 10^{-7}, 1 \cdot 10^{-6}]$	$[2 \cdot 10^{-7}, 4 \cdot 10^{-7}, 6 \cdot 10^{-7}]$	$[516, 516.0, 516]$
10^4	$[2 \cdot 10^{-7}, 3 \cdot 10^{-7}, 7 \cdot 10^{-7}]$	$[2 \cdot 10^{-7}, 3 \cdot 10^{-7}, 5 \cdot 10^{-7}]$	$[516, 516.0, 516]$

(b) Without changes of variables, with degree 20 polynomials for the leaf approximation bases.

Table 3.5: [Minimum, mean, maximum] values over 10 trials of the obtained results for the learning of $(III.iii)$ with and without introducing changes of variables, for different values of n and Q (when using a change of variables).

3.5 Conclusion

We proposed in this chapter learning algorithms in formats combining tree-based tensor formats and changes of variables: $g = v \circ h$, with $v \in \mathcal{T}_r^T(V^m)$ and $h \in H^m$. These algorithms compute a sequence of approximations with increasing so-called effective dimension m, which corresponds to the dimension of the range of h.

The proposed algorithms are sometimes able to compute accurate approximations with a storage complexity much smaller than when directly using the algorithms of Chapter 2, that is to say without introducing changes of variables.

We noticed however that when approximating certain functions, not introducing changes of variables yields better results than when using the algorithms presented in this chapter. We propose several leads as to why this is the case, and how to improve the algorithms. We considered for then changes of variables polynomial bases with total degree bounded by Q, that have a number of basis functions quickly growing with Q and d. This means that, even when considering moderate degrees Q, many parameters have to be estimated to describe the changes of variables, which can be a problem when the training sample sizes are small. Introducing sparsity in the matrix W might improve the performances of the algorithms. It would also enable them to perform input variables screening. This would call for the use or development of efficient sparsity-inducing nonlinear optimization algorithms. Furthermore, there is no guarantee that the cross-validation estimator of the generalization risk is a good estimator, and that it is well suited for model selection, required when seeking sparsity in the C^α, $\alpha \in T$ or a subset of nodes of T. This calls for the development of model selection techniques that do not rely on this cross-validation estimator.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
U_1	0.01	**0.90**	0.02	**0.38**	**0.19**	−0.01	−0.00	0.00	0.00	0.00
U_2	**0.16**	**-0.36**	**0.90**	**0.16**	0.07	0.02	0.00	0.00	0.00	0.01
U_3	**-0.99**	**-0.12**	−0.01	**0.11**	0.05	0.01	−0.00	−0.01	0.00	0.00
U_4	**-0.30**	**-0.67**	**-0.68**	0.01	−0.00	0.01	−0.00	−0.01	−0.00	−0.00

(a) $n = 10^2$.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
U_1	**0.50**	**0.52**	0.06	**0.62**	**0.31**	0.00	0.00	0.00	0.00	0.00
U_2	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00
U_3	**-0.70**	**-0.71**	−0.00	0.00	0.00	−0.00	0.00	0.00	−0.00	−0.00
U_4	**-0.29**	**0.96**	−0.00	0.00	0.00	−0.00	0.00	0.00	−0.00	−0.00
U_5	**0.97**	**-0.24**	−0.00	0.00	0.00	0.00	−0.00	−0.00	0.00	0.00
U_6	−0.07	**-1.00**	0.00	−0.00	−0.00	0.00	0.00	−0.00	0.00	0.00

(b) $n = 10^3$.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
U_1	**0.58**	**0.55**	−0.01	**0.54**	**0.27**	−0.00	0.00	−0.00	−0.00	−0.00
U_2	0.00	0.00	**-1.00**	0.00	0.00	−0.00	0.00	0.00	0.00	−0.00
U_3	**-0.72**	**-0.69**	−0.00	0.00	0.00	−0.00	0.00	0.00	0.00	0.00
U_4	**0.72**	**-0.69**	−0.00	−0.00	−0.00	0.00	0.00	−0.00	−0.00	0.00

(c) $n = 10^4$.

Table 3.6: Components in each W^i, $i \in M$, involving each variable X_j, $j \in D$, for the smallest error obtained with different sample sizes n and $Q = 1$ in Table 3.5 when approximating $(III.iii)$.

Finally, we saw that, by combining tree-based tensor format with changes of variables, we lost the multilinearity of the former, and with it the natural idea to recast the nonlinear risk minimization problem into a series of learning problems with linear model classes. We then understand the need to propose specific algorithms suited for learning with these formats.

Chapter 4

Application to uncertainty quantification in vibroacoustics

Contents

4.1 Introduction

In an underwater vehicle, numerous noise sources exist, such as the vibrations generated by the engine and transmitted to the hull or the noise created by the interaction between the propeller and the wake of the vehicle. Each source can generate in the frequency response function (FRF) of the sound produced by the vehicle discrete pressure peaks, characterized by an identifiable acoustic signature [46]: peaks in the frequency range from 0.1 Hz to 10 Hz are caused by the rotation of the propellers, and are characteristic of sounds that can be heard at several thousand kilometers of distance [47], which makes it an important source

to consider; peaks from several Hertz to several hundred Hertz are caused by the vibration of the hull and engine.

Such a structure may have to meet some discretion requirements that are classically described with a threshold curve for the sound pressure level (SPL): the SPL must take values that stay inferior to this threshold, which means that the vehicle will make less noise than a certain level.

In this study, in the context of the Eval-π project and the Joint Laboratory of Marine Technology between Naval Group, Centrale Nantes and Université de Nantes, the quantity of interest is the sound power level L_X, which is a real scalar quantity depending on the frequency and related to the SPL. An example of frequency response function and associated envelope curve of the sound power level is shown in Figure 4.1. The frequency response of L_X can be really sensitive to the uncertainties on the characteristics of the materials, on the position and direction of the forces, *etc*, and the numerical models able to simulate it are costly, both in computation time and in memory requirements.

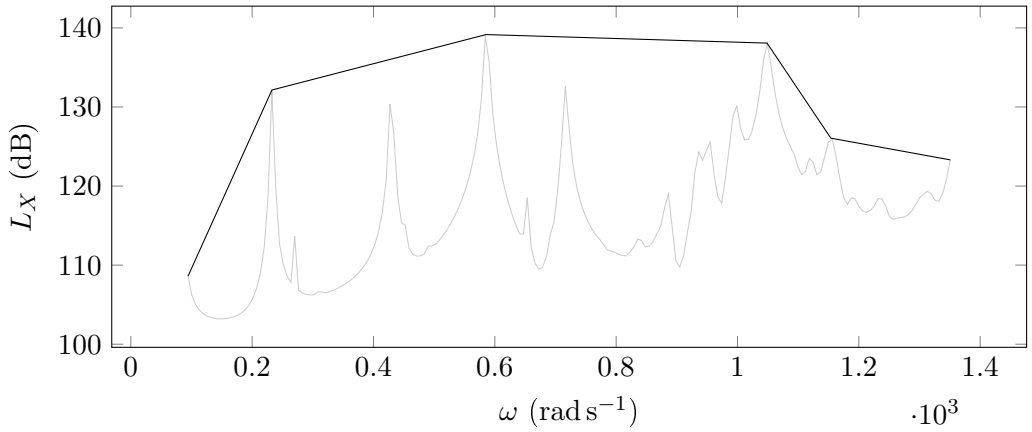


Figure 4.1: Representation of the sound power level L_X and its envelope curve as a function of the circular frequency ω.

The objective of this chapter is to propose a methodology to predict the impact of uncertainties on quantities of interest derived from the sound power level on a wide frequency band, based only on a limited number of direct computations and using the learning algorithms proposed in Chapters 1 and 2. Its outline is the following. We first present in Section 4.2 the sound power level of a vibrating underwater vehicle: how to compute it, the uncertainties on its input parameters, and some quantities of interest that can be deduced from it. Section 4.3 presents a parameter-dependent frequency transformation that aims at maximizing the similarity between the realizations of the frequency response of interest by aligning the resonance peaks. It acts as a pre-treatment for the learning of the frequency response of the quantity of interest in the transformed frequency space, presented in Section 4.4. Finally, Section 4.5 presents first results of the application of the proposed methodology to the learning of the envelope curve of the sound power level of a vibrating underwater vehicle.

4.2 Sound power level of an underwater vehicle

We consider a spherical domain S around an underwater vehicle, illustrated in Figure 4.2, which is discretized with a mesh with N_S nodes. The radius of the sphere is denoted by R.

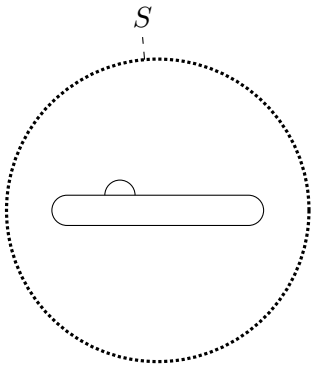
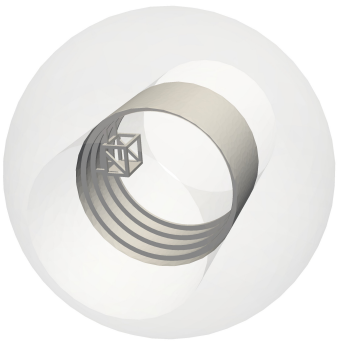
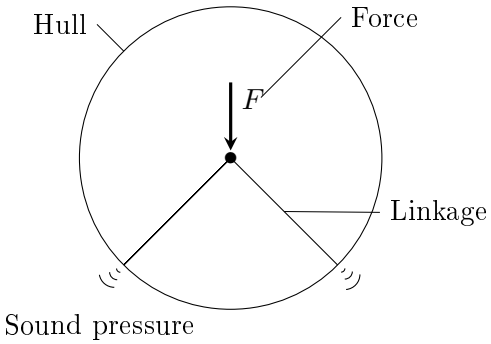


Figure 4.2: Representation of the integration domain around the vehicle.

We are here interested in the quantification of the noise due to the transmission of the structural vibrations through the hull of the vehicle to the fluid medium. To this end, a configuration close to the reality is used: a keelson attached to a section of hull, submerged in water (represented in Figure 4.3a), and to which a force is applied (see Figure 4.3b) [48]. This model allows the consideration of numerous configurations: different characteristics and thicknesses of the materials and different frequencies and positions for the vibrating force for instance.



(a) Geometry of the studied case.



(b) Representation of a noise source.

Figure 4.3: Studied case: a keelson attached to a section of hull submerged in water. An oscillating force is applied to the keelson and its energy is transmitted to the water through the hull.

The (u, p, ϕ) formulation (with u the displacement, p the pressure and ϕ the displacement potential) and finite-element discretization [49] yield the system

$$(-i\omega^3 I - \omega^2 M(x) + i\omega C + K(x))V(\omega, x) = B$$

with i the imaginary unit, I the impedance matrix, $M(x)$ the parameter-dependent mass matrix

$$M(x) = M_{\text{fluid}} + \sum_{j=1}^{3} x_j M_j,$$

C the damping matrix, $K(x)$ the parameter-dependent stiffness matrix

$$K(x) = x_4 K_0,$$

and B the vector associated with the source term. The number of degrees of freedom is $N_{\text{dof}} = 32970$. The solution vector V gathers the values of u_j, p_j, ϕ_j for all the nodes j of the mesh. The densities of the different materials as well as their stiffness are uncertain: x is a realization of the random variables with independent components $X = (X_1, \ldots, X_4)$, where X_i is with values in $\mathcal{X}_i$ and with probability law μ_i, $i = 1, \ldots, 4$, such that

- $X_1 \sim \mathcal{U}(7065, 8635)$,
- $X_2 \sim \mathcal{U}(5887.5, 9812.5)$,
- $X_3 \sim \mathcal{U}(7065, 8635)$,
- $X_4 \sim \mathcal{U}(0.9, 1.1)$.

Remark 4.2.1. *In dimension $d = 3$, all the representations in tree-based tensor format are equivalent, whatever the dimension tree over $\{1, \ldots, d\}$ is. The random variable X_4 models uncertainties that arguably are not realistic (the uncertainties on the stiffness are the same for all the different materials): it was artificially introduced in the model as a way to increase the dimension of the problem from 3 to 4, to propose first results that represent what can be expected in higher dimensions, where the dimension tree must be optimized, for instance with the proposed tree optimization Algorithm 5 of Chapter 1.*

The sound power level L_X, which is the quantity of interest for Naval Group, can be written

$$L_X = 10 \log_{10}\left(\frac{P}{P_0}\right)$$

with $P_0 = 6.414 \cdot 10^{-19}$ W a reference sound power, and

$$P = \frac{1}{2}\int_S \Re(p v_n^\star)\mathrm{d}S = \frac{1}{2}\int_S \frac{|p|^2}{Z_\Re}\mathrm{d}S \approx \frac{4\pi R^2}{2N_S}\sum_{i=1}^{N_S}\frac{|p_i|^2}{Z_\Re},$$

where $\Re(\cdot)$ denotes the real part of a complex number, p is the pressure field, approximated by p_i at the N_S nodes of the mesh of the spherical domain S, $v_n^\star$ is the particle normal velocity and $Z_\Re$ is the real part of the impedance.

Several quantities of interest, based on the sound power level L_X, can be studied. Its frequency response, for all values of ω and x in their respective domains, is denoted by q_1. It is the most accurate way of describing the sound power level, but also the most complex, because of the difficulty to represent its circular frequency dependency. An example of full

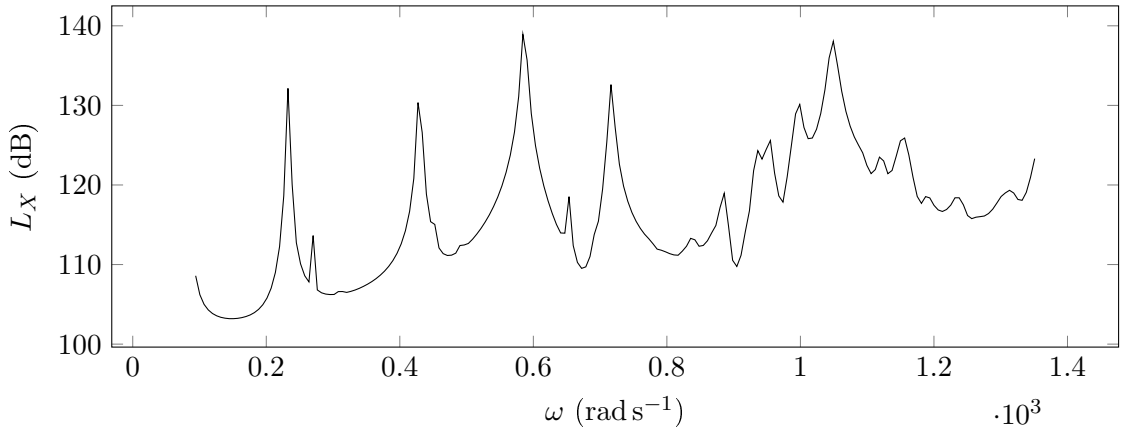
frequency response is depicted in Figure 4.4a. Furthermore, for many industrial applications, knowing the FRF for all circular frequencies is not necessary. Taking this into account, the one-third octave band[1] FRF of the sound power level, denoted by q_2, is often studied. It offers a frequency resolution suited for numerous applications, and is constant on each one-third octave band. Figure 4.4b shows an example of one-third octave band FRF, computed from the full FRF of Figure 4.4a. Finally, as mentioned at the beginning of the chapter, a particularly interesting quantity of interest for Naval Group is the envelope curve of the FRF of the sound power level, that we denote by q_3. This function, which is piece-wise linear in ω, connects (some of) the peaks of the FRF, so that the latter never has values above the former. These peaks can be obtained, for instance, with software usually available in the industry. This envelope function then guaranties that the sound power level never goes above it, for any value of ω and x. An envelope curve of the FRF of Figure 4.4a is displayed in Figure 4.4c. One can notice that the proposed envelope curve passes only through some of the peaks of the FRF: the choice of these peaks is free and depends on the use of the envelope. In this work, we choose to select only some of the peaks of $q_1(\omega, x_j)$ for a given x_j, with the following rule: beginning by selecting the circular frequency ω_0^j of the maximum of the FRF on the set of peaks frequencies denoted by Ω_p^j, the algorithm retains the next largest value both on the left and on the right of ω_0, that we call ω_{-1}^j and ω_1^j respectively. Then, the next largest values on the left of ω_{-1}^j and on the right of ω_1^j are selected and so on, until no points are left in Ω_p^j to be selected. The bounds of the circular frequency domain are also retained. The envelope in Figure 4.4c is obtained by computing a piece-wise linear interpolation of the points $(\omega_i^j, q_1(\omega_i^j, x_j))$. This procedure is described in Algorithm 13.

In the following, we present methodologies to compute approximations of a FRF of interest depending on ω and x, which could be either q_1, q_2 or q_3, and we focus on the approximation of the envelope curve q_3, being the quantity of greatest interest for Naval Group. But first, we introduce a problem-dependent peaks aligning procedure that acts as a pre-treatment for the learning, by aligning the frequencies of the peaks of the FRF for all values of x.

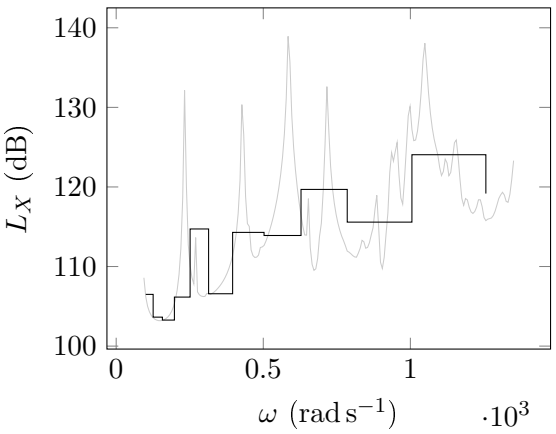
4.3 Peaks aligning frequency transformation

We can see in Figure 4.5a the FRFs of the sound power level $q_1(\omega, X)$ associated with different realizations of X. We notice that the circular frequencies of the peaks of the FRFs change from one realization of X to another. In this section, we propose a parameter-dependent frequency transformation inspired from [50], that aims at aligning the peaks of the FRF (or a selection of them, for instance by applying Algorithm 13) in order to increase the similarity between the FRFs for different values of x, with the hope of reducing the complexity of the approximations to compute. For instance, applying to the FRFs $q_1(\omega, x)$ of Figure 4.5a the piece-wise linear frequency transformations of Figure 4.5b, we obtain the

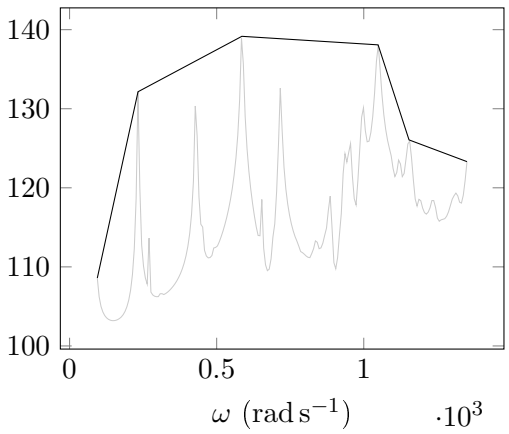
[1] See `https://law.resource.org/pub/us/cfr/ibr/002/ansi.s1.11.2004.pdf`. Accessed on September 27th, 2019.



(a) Narrow band FRF.



(b) One-third octave band FRF.



(c) Envelope curve of the FRF.

Figure 4.4: Example of frequency response functions of interest: narrow band, one-third octave band and envelope curve.

FRFs $\tilde{q}_1(\nu, x)$ of Figure 4.5c, where a selection of peaks are aligned with the peaks of one common reference realization, arbitrarily chosen.

Let us assume that we know the circular frequencies of (a selection of) peaks of $q_1(\omega, x)$ in the range $\mathcal{W} = [\omega_{\min}, \omega_{\max}]$ (the circular frequency domain of interest) for n realizations of X, stored in $\pi^j = (\omega_1^j, \ldots, \omega_{n_\omega^j}^j)$, $j = 1, \ldots, n$.

We choose one realization of X to be the reference with which the peaks for other realizations will be aligned. If the number of peaks changes from one realization of X to another, we select a reference index in the set of realizations with the largest number of peaks $\{k : n_\omega^k = \max_{j=1,\ldots,n} n_\omega^j\}$. We denote the reference peaks frequencies by $\pi^{\text{ref}} = (\omega_1^{\text{ref}}, \ldots, \omega_{n_\omega^{\text{ref}}}^{\text{ref}})$.

The goal is to build a function $\nu(\omega, x; \pi^{\text{ref}})$ which is such that, for a given x and a circular frequency ω_i corresponding to the i-th peak of the function $q_1(\cdot, x)$, we have

$$\nu(\omega_i, x; \pi^{\text{ref}}) = \underset{\omega \in \pi^{\text{ref}}}{\operatorname{argmin}} |\omega_i - \omega|,$$

Algorithm 13 Filtering of the peaks of the function $q_1(\cdot, x_j)$ for a given x_j.

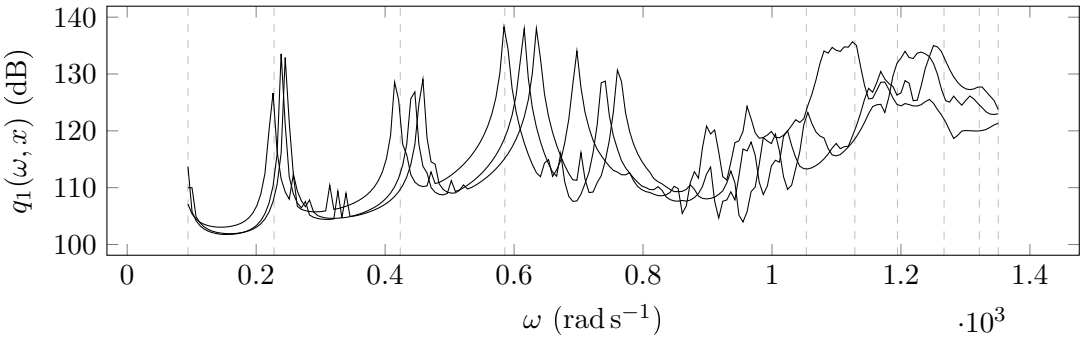
Inputs: input parameter value x_j, frequencies of the peaks of $q_1(\cdot, x_j)$ stored in Ω_p^j, associated values of $q_1(\omega, x_j)$ for $\omega \in \Omega_p^j$

Outputs: selected peaks frequencies, stored in $\pi^j = (\omega_1^j, \ldots, \omega_{n_\omega^j}^j)$

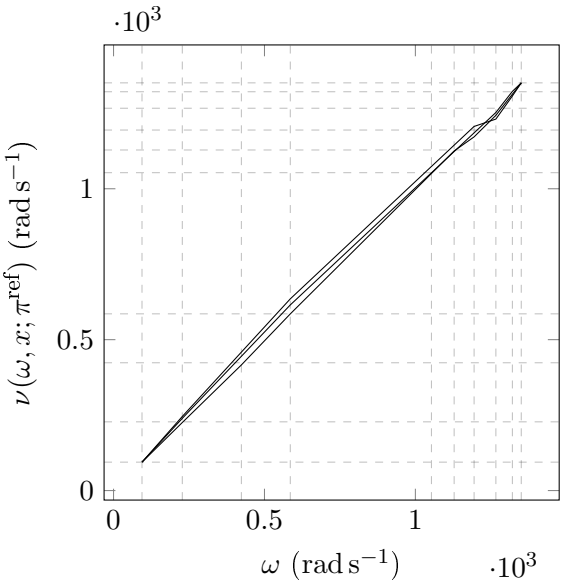
$\omega_0^j = \operatorname{argmax}_{\omega \in \Omega_p^j} q_1(\omega, x_j)$
$i = 0$, $n_\omega^+ = 0$
while $\{\zeta \in \Omega_p^j \colon \zeta > \omega_{i-1}^j\} \neq \emptyset$ **do**
  $i \leftarrow i + 1$, $n_\omega^+ \leftarrow n_\omega^+ + 1$
  $\omega_i^j = \operatorname{argmax}_{\omega \in \{\zeta \in \Omega_p^j \colon \zeta > \omega_{i-1}^j\}} q_1(\omega, x_j)$
end while
$\omega_{i+1}^j = \max_{\omega \in \Omega_p^j} \omega$, $n_\omega^+ \leftarrow n_\omega^+ + 1$
$i = 0$, $n_\omega^- = 0$
while $\{\zeta \in \Omega_p^j \colon \zeta < \omega_{i+1}^j\} \neq \emptyset$ **do**
  $i \leftarrow i - 1$, $n_\omega^- \leftarrow n_\omega^- + 1$
  $\omega_i^j = \operatorname{argmax}_{\omega \in \{\zeta \in \Omega_p^j \colon \zeta < \omega_{i+1}^j\}} q_1(\omega, x_j)$
end while
$\omega_{i-1}^j = \min_{\omega \in \Omega_p^j} \omega$, $n_\omega^- \leftarrow n_\omega^- + 1$
$n_\omega^j = n_\omega^+ + n_\omega^-$
$\omega_i^j \leftarrow \omega_{i+n_\omega^-+1}^j$, $i = 1, \ldots, n_\omega^+$ (change of indices of the frequencies)
return $\pi^j \leftarrow (\omega_1^j, \ldots, \omega_{n_\omega}^j)$

that is to say, the peak of frequency ω_i must be aligned with its closest reference peak.

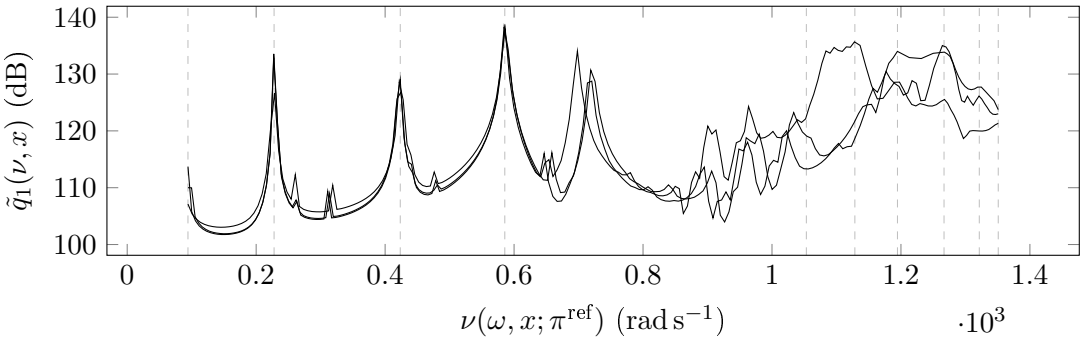
Algorithm 14 proposes a methodology to compute the peaks aligning parameter-dependent frequency transformation, which uses the empirical interpolation method [51] and requires the realizations $(\omega_i^j, q_1(\omega_i^j, x_j))$, $i = 1, \ldots, n_\omega^j$ and $j = 1, \ldots, n$. This algorithm involves the computation of approximations in tree-based tensor format, using the learning algorithms proposed in Chapters 1 and 2. We choose for the approximation space $\mathcal{H}_\alpha$, $\alpha \in \{1, \ldots, d\}$, a polynomial basis $\{\phi_{i_\alpha}^\alpha\}_{i_\alpha \in I^\alpha}$, orthogonal with respect to the probability measure μ_α of X_α.



(a) FRFs $q_1(\omega, x)$ for several realizations x of X.



(b) Frequency transformations associated with the FRFs depicted in Figure 4.5a, computed with Algorithm 14.



(c) FRFs of Figure 4.5a after applying the parameter-dependent frequency transformation of Figure 4.5b, for different realizations x of X.

Figure 4.5: Example of the use of a parameter-dependent frequency transformation to align some peaks of FRFs. The reference circular frequencies are displayed in dashed lines.

Algorithm 14 Computation of the peaks aligning frequency transformation $\nu(\omega, x; \pi^{\text{ref}})$.

Inputs: $\{x_j\}_{j=j}^n$, $\pi^j = (\omega_1^j, \ldots, \omega_{n_\omega^j}^j)$, $j = 1, \ldots, n$, reference peaks frequencies $\pi^{\text{ref}} = (\omega_1^{\text{ref}}, \ldots, \omega_{n_\omega^{\text{ref}}}^{\text{ref}})$, bases $\{\phi_{i_\alpha}^\alpha\}_{i_\alpha \in I^\alpha}$, orthogonal with respect to μ_α the probability measure of X_α, $\alpha \in \{1, \ldots, d\}$, number N of interpolation points for the empirical interpolation method

Outputs: peaks-aligning frequency transformation $\nu(\omega, x; \pi^{\text{ref}})$

1: **for** $j = 1, \ldots, n$ **do**

2: build the frequency transformation $\omega \mapsto \nu^{\text{true}}(\omega, x_j; \pi^{\text{ref}})$, a piece-wise linear interpolation on the set $\{(\omega_i^j, \operatorname{argmin}_{\omega \in \pi^{\text{ref}}} |\omega_i^j - \omega|)\}_{i=1}^{n_\omega^j}$, so that

$$\nu^{\text{true}}(\omega_i^j, x_j; \pi^{\text{ref}}) = \operatorname*{argmin}_{\omega \in \pi^{\text{ref}}} |\omega_i^j - \omega|, \quad i = 1, \ldots, n_\omega^j$$

3: **end for**

4: perform an empirical interpolation [51] of the set of functions $\{\nu^{\text{true}}(\cdot, x_j; \pi^{\text{ref}})\}_{j=1}^n$, to obtain N magic points ζ_k and their associated interpolation functions $\varphi_k(\omega)$, $k = 1, \ldots, N$

5: **for** $k = 1, \ldots, N$ **do**

6: build an approximation $\nu_k(x)$ of $\nu_k^{\text{true}}(x) := \nu^{\text{true}}(\zeta_k, x)$ in tree-based tensor format using Algorithm 9 with the training sample $S = \{(x_j, \nu_k^{\text{true}}(x_j))\}_{j=1}^n$

7: **end for**

8: define the parameter-dependent frequency transformation as

$$\nu(\omega, x; \pi^{\text{ref}}) = \sum_{k=1}^N \alpha_k(x) \varphi_k(\omega)$$

with the $\alpha_k(x)$, $k = 1, \ldots, N$, satisfying the interpolation properties

$$\sum_{k=1}^N \alpha_k(x) \varphi_k(\zeta_l) = \nu_l(x) \approx \nu_l^{\text{true}}(x), \quad l = 1, \ldots, N$$

4.4 Learning the frequency response of the quantity of interest

In this section, we first present how to compute an approximation of a FRF of interest q in tree-based tensor format. Then, we focus on the learning of the envelope curve of the FRF, denoted by q_3, by computing approximations in tree-based tensor format of the magnitude of q at the reference circular frequencies.

In what follows, the approximation, denoted by $g(\nu, x)$, is computed in the transformed frequency space $\nu(\omega, x; \pi^{\text{ref}})$, where (a selection of) the peaks of the FRF are aligned with the reference peaks (with circular frequencies π^{ref}), for any $x \in \mathcal{X}$. Then, the approximation of q at a point (ω, x) can be obtained by evaluating g at the point $(\nu(\omega, x; \pi^{\text{ref}}), x)$.

4.4.1 Approximation of the frequency response function of interest q

We consider here that the transformed circular frequency ν is a realization of a uniform random variable $\widetilde{\Omega} \sim \mathcal{U}(\omega_{\min}, \omega_{\max})$, with values in $\mathcal{W} = [\omega_{\min}, \omega_{\max}]$, independent of X_i, $i = 1, \ldots, d$. An approximation of the $(d+1)$-dimensional function $f(X') = \tilde{q}(\widetilde{\Omega}, X)$, with $X' = (\widetilde{\Omega}, X)$ and $\tilde{q}(\nu, x) = q(\omega, x)$, can be constructed in tree-based tensor format, using the learning algorithms described in Chapter 2.

For the functions $\{\phi^{\eta}_{i_\eta}\}_{i_\eta \in I^\eta}$ associated with the random variable $X'_\eta = X_{\eta-1}$, $\eta = 2, \ldots, d+1$, we can for instance use polynomial bases, orthonormal with respect to the measure $\mu_{\eta-1}$ of $X_{\eta-1}$.

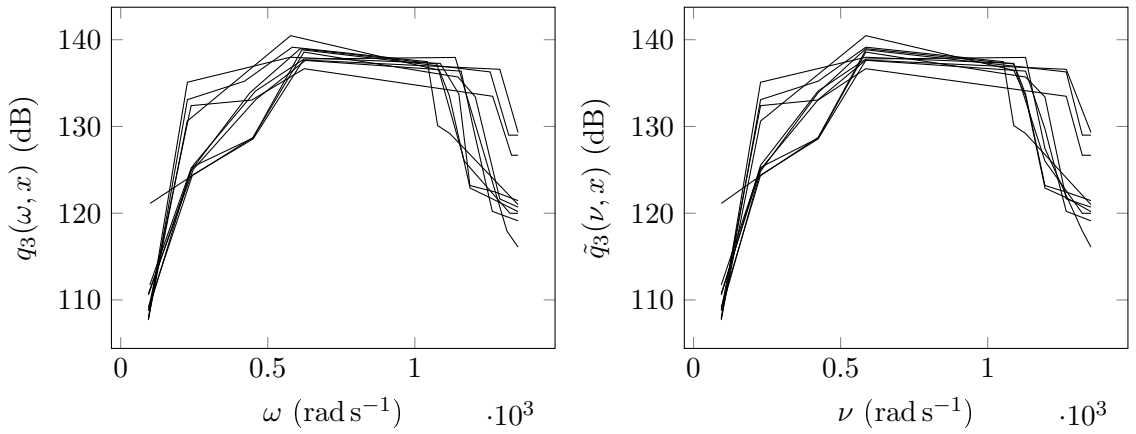
The main issue with this approach is the choice of the functions $\{\phi^1_{i_1}\}_{i_1 \in I^1}$ associated with the random variable $\widetilde{\Omega}$. Indeed, it might be difficult to represent the frequency dependency of the FRF with a good accuracy. This can be seen for example in Figure 4.5a, where the function $q_1(\cdot, x)$ is displayed for several realizations x of X: we understand that, for a given x, representing $q_1(\cdot, x)$ on a polynomial basis for instance would require a very high polynomial degree. One then ought to find other ways of representing such functions, for instance by using bases of rational polynomials [52, 53], or in tree-based tensor format using tensorization, as described in Chapter 5.

The full FRF q_1 not being of major interest for Naval Group, another way to circumvent this problem is to consider other quantities, derived from q_1, with a dependency on ω less complex to represent. This is the case for the one-third octave band FRF q_2, which is piece-wise constant in ω, and the envelope curve q_3, which is piece-wise linear in ω, and thus, in the transformed circular frequency space, for $\tilde{q}_2(\nu, x) = q_2(\omega, x)$ which is piece-wise constant in ν and for $\tilde{q}_3(\nu, x) = q_3(\omega, x)$ which is piece-wise linear in ν. For $\tilde{q}_2$, one could for instance consider that $\widetilde{\Omega}$ is a discrete uniform random variable on the set of midpoints of the one-third octave bands, and choose for the functions $\{\phi^1_{i_1}\}_{i_1 \in I^1}$ discrete polynomials (see Remark 1.2.2).

4.4.2 Approximation of the envelope curve $\tilde{q}_3(\nu, x)$ of the frequency response function of interest

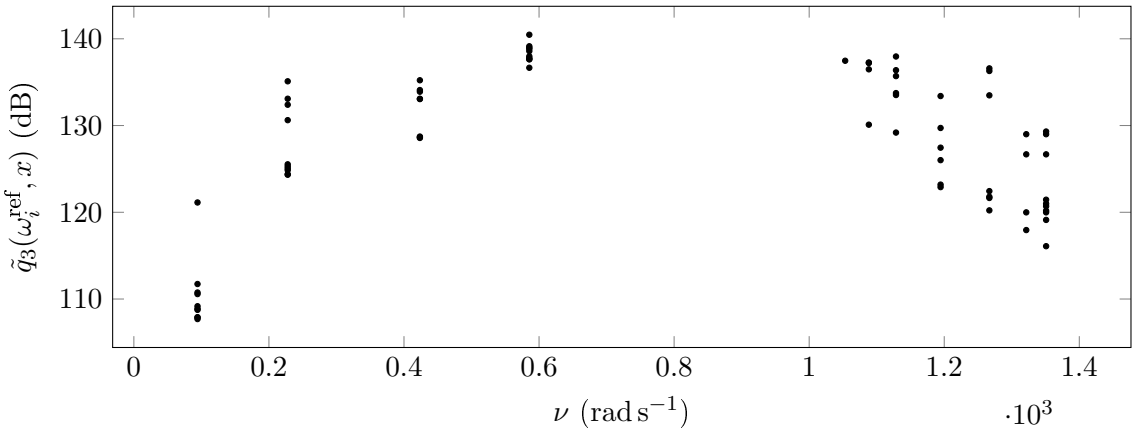
We focus in this section on the problem of the approximation of the envelope curve of a FRF of interest in the transformed circular frequency space, denoted by $\tilde{q}_3(\nu, x)$.

As it can be seen in Figure 4.6a, the envelope curves of interest are piece-wise linear in ν, with breakpoints corresponding to a selection of peaks of the FRF. One can see in Figure 4.6b that, in the transformed frequency space $\nu(\omega, x; \pi^{\text{ref}})$, the breakpoints of the envelope curve share the same circular frequency, for any value of x, contrary to the envelopes in the original circular frequency space in Figure 4.6a.



(a) Envelope curves in the original circular frequency space.

(b) Envelope curves in the transformed circular frequency space.



(c) Breakpoints of the envelope curves in the transformed circular frequency space. Notice the variable number of points associated with each reference circular frequency.

Figure 4.6: Realizations of the envelope curve before and after transformation of the circular frequency space, and training samples for the learning of the magnitude of the breakpoints of the envelope.

Then, in order to approximate the envelope curve in the transformed frequency space, one needs to learn the n_ω^{ref} magnitudes of the breakpoints, as functions of the random input

parameters x. More formally, the envelope curve writes

$$f(\nu, x) = \sum_{i=1}^{n_\omega^{\text{ref}}-1} \left(\frac{\nu - \omega_i^{\text{ref}}}{\omega_{i+1}^{\text{ref}} - \omega_i^{\text{ref}}} (f_{i+1}(x) - f_i(x)) + f_i(x) \right) \mathbf{1}_{\nu \in [\omega_i^{\text{ref}}, \omega_{i+1}^{\text{ref}}]},$$

where the functions $f_1, \ldots, f_{n_\omega^{\text{ref}}}$ give the magnitudes of the reference peaks as a function of x, and are to be approximated in tree-based tensor format using the algorithms proposed in Chapters 1 and 2.

These functions being approximated by the functions $g_1, \ldots, g_{n_\omega^{\text{ref}}}$ respectively, the approximated envelope curve writes

$$g(\nu, x) = \sum_{i=1}^{n_\omega^{\text{ref}}-1} \left(\frac{\nu - \omega_i^{\text{ref}}}{\omega_{i+1}^{\text{ref}} - \omega_i^{\text{ref}}} (g_{i+1}(x) - g_i(x)) + g_i(x) \right) \mathbf{1}_{\nu \in [\omega_i^{\text{ref}}, \omega_{i+1}^{\text{ref}}]}.$$

The approximation of f_i, $1 \leq i \leq n_\omega^{\text{ref}}$ is constructed using the training sample $S_i = \{(x_k, q_1(\omega_k, x_k))\}_{k \in K_i}$, where K_i gathers the indices of the available realizations of X that are such that there exists a circular frequency ω_k of a peak of $q_1(\cdot, x_k)$ which is such that $\nu(\omega_k, x_k; \pi^{\text{ref}}) = \omega_i^{\text{ref}}$. In other words, the training sample to learn the magnitude of the i-th reference peak is constituted of values of the magnitude of this peak for several realizations of X. Figure 4.6c shows, for each reference circular frequency ω_i^{ref}, the training sample S_i.

Remark 4.4.1. *If, for a given realization x of X, the function $\nu(\omega, x; \pi^{ref})$ maps two or more circular frequencies to the same reference circular frequency ω_i^{ref}, only the one associated with the highest magnitude of the FRF of interest q_1 is retained for the training sample S_i.*

4.5 Application to envelope learning: first results

We apply the proposed methodology to the problem of the approximation of the envelope curve of the frequency response function of the sound pressure level of an underwater vehicle, presented in Section 4.2.

In order to compute this approximated envelope curve, denoted by g, we need the circular frequencies of the peaks and associated evaluations of the FRF for n realizations $x_1, \ldots, x_n$ of X. As described in Section 4.2, these peaks circular frequencies are stored in π^j, $j = 1, \ldots, n$. In this first study, we consider $n = 190$, with 10 other realizations serving as a validation sample.

For a given realization x_j, $1 \leq j \leq n$, the peaks circular frequencies π^j are obtained using Algorithm 13, with the candidate peaks circular frequencies Ω_p^j computed with a piece-wise quadratic interpolation of the FRF on a uniform circular frequency grid of 200 points (1 per Hertz). In an industrial context, this approach would be intractable, the expensive quantity of interest needing to be evaluated on a fine circular frequency grid for many realizations

of the parameters; instead, one could solve for the signed eigenfrequencies of the FRF, that can be obtained with software generally available in the industry.

4.5.1 Computation of the peaks aligning circular frequency transformation

We begin by computing the peaks aligning circular frequency transformation $\nu(\omega, x; \pi^{\text{ref}})$, by applying the methodology described in Section 4.3. We choose $N = 50$ magic points, and orthonormal polynomial bases of degree 10 for the approximation $\nu_l(x)$ of $\nu_l^{\text{true}}(x)$, $l = 1, \ldots, N$ in Algorithm 14.

Table 4.1 presents the coefficient of variation of the frequencies of the peaks of the FRFs of the test sample associated with the reference peaks before and after the application of the constructed circular frequency transformation. We see that, in most cases, the computed transformation is able to reduce the dispersion around each reference frequency. However, this decrease is often quite small, as it can be seen in Figure 4.7, where the reduction of the dispersion is not distinguishable. Note that the 0 value for the dispersion around ω_6^{ref} is due to the fact that only one point was available for the validation sample associated with this reference circular frequency.

We then see that we were able to compute a circular frequency transformation $\nu(\omega, x; \pi^{\text{ref}})$ using the proposed methodology, that offered an unsatisfactory precision, certainly due to the small size of the training samples.

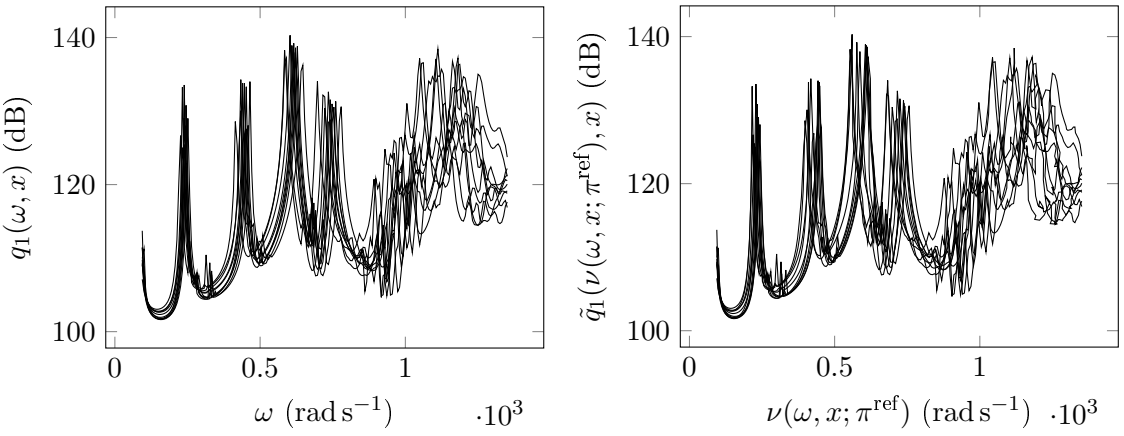
	ω_1^{ref}	ω_2^{ref}	ω_3^{ref}	ω_4^{ref}	ω_5^{ref}	ω_6^{ref}	ω_7^{ref}	ω_8^{ref}	ω_9^{ref}	ω_{10}^{ref}	ω_{11}^{ref}
Before	1.39	3.07	3.48	2.93	0.68	0	1.55	1.53	1.52	1.28	0.00
After	3.10	1.34	3.42	1.00	0.68	0	1.67	1.31	1.29	1.29	0.00

Table 4.1: Coefficient of variation (in %) of the frequencies of the peaks associated with the reference peaks, before and after the circular frequency transformation.

4.5.2 Approximation of the envelope curve in the transformed circular frequency space

We now approximate, in the transformed circular frequency space, the magnitude of the FRF at the reference circular frequencies, by applying the methodology of Section 4.4.2. We choose orthonormal polynomial bases of degree 10 for the approximation g_i of f_i, $i = 1, \ldots, n_\omega^{\text{ref}}$, the magnitude of the i-th reference peak.

Table 4.2 shows, for each reference circular frequency, the error on the magnitude of the FRF, obtained on a validation sample. We see that we are able to predict, with a good accuracy, these magnitudes, and hence the envelope curve of the FRF in the transformed circular frequency space. This satisfactory match can also be seen in Figure 4.8 that depicts, for each realization of the validation set, the true and approximated envelopes.



(a) FRFs in the original circular frequency space. (b) FRFs in the transformed circular frequency space.

Figure 4.7: Effect of the computed circular frequency transformation on 10 validation FRFs.

ω_1^{ref}	ω_2^{ref}	ω_3^{ref}	ω_4^{ref}	ω_5^{ref}	ω_6^{ref}	ω_7^{ref}	ω_8^{ref}	ω_9^{ref}	ω_{10}^{ref}	ω_{11}^{ref}
1.83	1.86	1.26	0.55	0.87	1.58	2.02	2.34	0.73	0.95	1.10

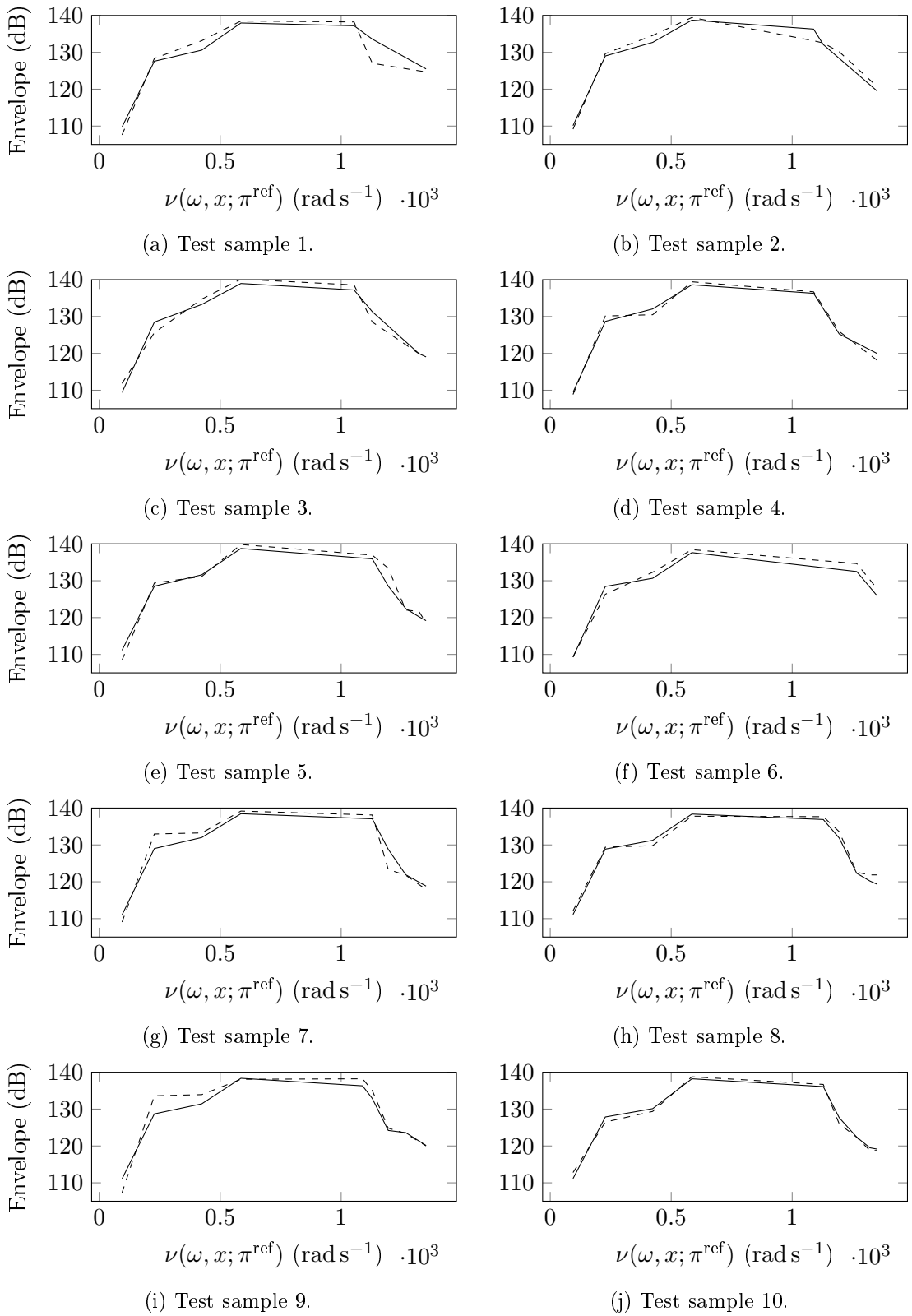
Table 4.2: Errors (in %) of the magnitude of the FRF at the reference peaks, on a test sample of 10 realizations.

4.6 Conclusion

We proposed in this chapter a method dedicated to the uncertainty quantification in vibroacoustics, that involves learning functions of random parameters using the algorithms proposed in Chapters 1 and 2. The parameter-dependent circular frequency transformation $\nu(\omega, x; \pi^{\text{ref}})$, by aligning a selection of signed peaks of the FRF of interest with chosen reference peaks, maximizes the similarity between its different realizations. Then, in the transformed circular frequency space, the magnitudes of the FRF at the reference circular frequencies are approximated, to yield an approximation of the envelope curve of the FRF $\tilde{q}_3(\nu, x)$.

The proposed first results, obtained with a small training sample, are promising. Indeed, we saw that the learned circular frequency transformation was able to reduce the dispersion of most of the selected peaks of the FRF, and we obtained a satisfactory accuracy of the approximation of the envelope curve of the FRF in the transformed space, given the small size of the training sample.

This study should be continued, by analyzing the impact of all the steps of the proposed method on the accuracy (selection of the peaks to align, number of magic points, *etc.*). Furthermore, it would be interesting to perform this study with the real frequencies of the peaks, and not with the frequencies interpolated from a grid of evaluations of the FRF, as it was performed here. Finally, more training data should be generated, in order to increase the accuracy of the computed approximations.



(a) Test sample 1.

(b) Test sample 2.

(c) Test sample 3.

(d) Test sample 4.

(e) Test sample 5.

(f) Test sample 6.

(g) Test sample 7.

(h) Test sample 8.

(i) Test sample 9.

(j) Test sample 10.

Figure 4.8: True (dashed line) and approximated (solid line) envelope curves of the FRF $\tilde{q}_1$ in the transformed frequency space, on a test sample of 10 realizations.

Chapter 5

Tensorization of functions

Contents

5.1 Introduction

We present, in this chapter, an exploratory work to study the potential of the identification of a univariate function with a multivariate function, called tensorization, and of its learning in tree-based tensor format.

The quantized tensor-train (QTT) format [54, 35], a particular case of tensorization, has been successfully applied to the identification of vectors with 2^d entries with $2 \times \cdots \times 2$ order-d tensors in tensor-train format to solve elliptic partial differential equations with a very high spatial resolution [55, 56].

In Section 5.2, we propose a more general tensorization, where the base, denoted by b, is not set to 2 (the case of quantization) but can be any integer greater than 1, and which can encode functions defined on $[0, 1[$ instead of vectors or matrices. Tensorization enables the use of the algorithms proposed in Chapters 1 and 2 for the learning of univariate functions with tree-based tensor formats. In Section 5.3, we illustrate the potential of tensorization and low-rank approximations, using as a test case a non-smooth frequency response function, and study the influence of the base b and resolution d on the quality and storage complexity of the obtained approximations. Finally, we study in Section 5.4 the performances of the learning algorithms proposed in Chapters 1 and 2 applied to the approximation of the same

non-smooth frequency response function for different values of b, d and n, the training sample size.

5.2 Tensorization of univariate functions and representation in tree-based tensor format

We consider, without loss of generality, functions defined on the interval $I = [0,1[$. An element $x \in I$ can be identified with the tuple $(i_1, \ldots, i_d, y)$, such that

$$x = t_{b,d}(i_1, \ldots, i_d, y) = \sum_{k=1}^{d} i_k b^{-k} + b^{-d} y$$

with $i_k \in I_b = \{0, \ldots, b-1\}$, $k = 1, \ldots, d$, and $y = b^d x - \lfloor b^d x \rfloor \in [0,1[$. The tuple $(i_1, \ldots, i_d)$ is the representation in base b of $\lfloor b^d x \rfloor$.

We then introduce a tensorization map $T_{b,d}$, which associates to a univariate function F defined on I the multivariate function

$$f(i_1, \ldots, i_d, y) = F(t_{b,d}(i_1, \ldots, i_d, y)),$$

defined on $I_b^d \times I$.

It can be proved that $T_{b,d}$ defines a linear isometry from $L^2(0,1)$ to $(\mathbb{R}^{I_b})^{\otimes d} \otimes L^2(0,1)$, the latter tensor space being equipped with the canonical inner product.

The tensor space

$$V_{b,d,S} = V_{b,d} \otimes S = (\mathbb{R}^{I_b})^{\otimes d} \otimes S,$$

with $V_{b,d} = \mathbb{R}^{I_b^d} = (\mathbb{R}^{I_b})^{\otimes d}$ and S a subspace of functions defined on $[0,1[$, is the set of linear combinations of the elementary tensors

$$(f_1 \otimes \cdots \otimes f_d \otimes f_{d+1})(i_1, \ldots, i_d, y) = f_1(i_1) \cdots f_d(i_d) f_{d+1}(y)$$

with $f_k \in \mathbb{R}^{I_b}$, $k = 1, \ldots, d$, and $f_{d+1} \in S$. By convention, $V_{b,0,S} = S$. $V_{b,d,S}$ therefore defines a subspace in $L^2(0,1)$. For $S = \mathbb{P}_p$ (polynomial space), $V_{b,d,S}$ corresponds to piecewise polynomials on a uniform partition of I into b^d elements.

Considering a dimension tree T over $\{1, \ldots, d+1\}$ and a tree-based rank $r = (r_\alpha)_{\alpha \in T}$, the set of functions in tree-based tensor format in $V_{b,d,S}$ writes

$$\mathcal{T}_r^T(V_{b,d,S}) = \{g \in V_{b,d,S} \colon \operatorname{rank}_\alpha(g) \leq r_\alpha, \alpha \in T\}.$$

Using this identification, one can apply all the procedures detailed in Chapters 1 and 2 for the representation, approximation and learning of univariate functions in tree-based tensor format.

5.3 Illustration of the approximation of a tensorized frequency response function

We are here interested in the tensorization and approximation in tree-based tensor format of a univariate frequency response function, which has similar features to the frequency response function of the sound power level of the underwater structure presented in Chapter 4, for a given realization of the random input parameters.

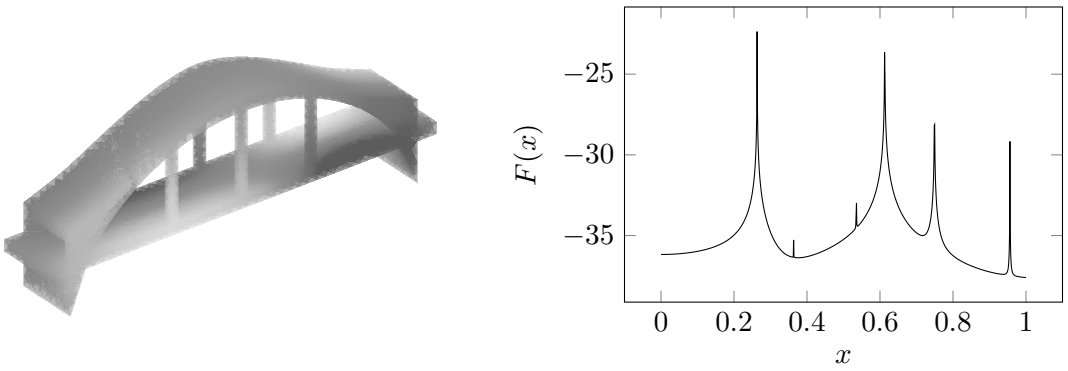
We study the vibrations of a bridge, depicted in Figure 5.1a. The finite-element discretization yields the following system to solve:

$$(-(x\Delta x + x_{\min})^2 M + (1 + i\eta)K)U(x) = B$$

with $x \in [0, 1[$ the standardized circular frequency, $x_{\min}$ and $x_{\max} = x_{\min} + \Delta x$ the bounds of the circular frequency domain, i the imaginary unit, M the mass matrix, K the stiffness matrix, η the loss factor characterizing the damping, and B the source vector. The number of degrees of freedom is $N_{\text{dof}} = 8952$. The solution $U(x)$ contains the values of the displacement $u_j(x)$ for all the nodes j of the mesh. The function of interest to approximate, denoted by F, writes

$$F(x) = \log\left(\sum_{j=1}^{N_{\text{dof}}} u_j(x)^2\right),$$

and is represented in Figure 5.1b.



(a) Geometry of the bridge.

(b) Frequency response function F.

Figure 5.1: Description of the considered function.

We aim at approximating in tree-based tensor format the function f such that

$$f(i_1, \ldots, i_d, y) = F(t_{b,d}(i_1, \ldots, i_d, y)),$$

with b the base and d the resolution. The dimension of $V_{b,d}$ being equal to b^d, a function $f \in V_{b,d,S}$ admits a representation

$$f(i_1, \ldots, i_d, y) = \sum_{i=1}^{b^d} c_i \phi_i(y),$$

with some coefficients $c_1, \ldots, c_{b^d}$ and functions $\{\phi_i\}_{i=1}^{b^d}$ in $L^2(0,1)$. We choose to compute a piece-wise interpolation $\mathcal{I}_{b,d,p} f$ of f:

$$\mathcal{I}_{b,d,p} f(i_1, \ldots, i_d, y) = \sum_{i=1}^{b^d} c_i \mathcal{I}_p \phi_i(y) \approx f(i_1, \ldots, i_d, y),$$

where $\mathcal{I}_p$ denotes an interpolation operator on a polynomial space of degree p. The $p+1$ interpolation points are chosen as the roots of the Chebyshev polynomial of first kind of degree $p+1$.

The function $\mathcal{I}_{b,d,p} f$ is then approximated by a function g in tree-based tensor format using a higher-order singular value decomposition (HOSVD) with a linear tree (denoted by T_0) and a tolerance of $\tau = 10^{-10}$, which is finally optimized using Algorithm 5 of Chapter 1 (also with a tolerance of τ) to yield a tree T possibly reducing the storage complexity of the approximation. The aim of this illustration is to study the capacities of tree-based tensor formats to find a representation of $\mathcal{I}_{b,d,p} f$, if it exists, with a storage complexity smaller than $(p+1)b^d$ (the number of coefficients of the interpolation), as a function of (b, d, p), the base, resolution and polynomial degree respectively.

The obtained results are presented in Table 5.1 and Figure 5.2. The error ε is the relative L^2 error between the approximation g in tree-based tensor format and the function f, estimated with an independent test sample.

We can see that, as one could expect, increasing either b, d or p decreases the obtained error ε while increasing the storage complexity $C(T, r)$. We also see that the representation in tree-based tensor format enables a decrease in storage complexity only below a certain error level, or equivalently above a certain complexity. Above this error level, the representation in tree-based tensor format yields a storage complexity greater than the complexity of the interpolation, namely $(p+1)b^d$, as one can see in the rows of Tables 5.1a and 5.1b corresponding to the smallest values of d considered for each b and p. This can also be observed in Figure 5.2 that depicts, for all the considered tuples (b, d, p), the error versus storage complexity of the original interpolation, of its approximation in tree-based tensor format with the linear tree T_0, and of the approximation with an optimized tree T. We see that for small errors (ε less than approximately $4 \cdot 10^{-3}$), the complexity $(p+1)b^d$ is above both $C(T_0, r_0)$ and $C(T, r)$, whereas it is the opposite for errors greater than approximately $4 \cdot 10^{-3}$. We notice a trend in the power of compression of the tree-based tensor format: we obtain increasingly larger gains in storage complexity with the increase of $(p+1)b^d$, which means that this approach becomes more and more interesting with larger values of the base,

the resolution and the degree.

An interesting result is the following: classically, the base b is chosen equal to 2 (the case of quantization). Then, in order to decrease the error, one can either increase the resolution d or the polynomial degree p. We see in Table 5.1 that considering $(b, d, p) = (4, 5, 2)$ yields both a smaller error and storage complexity than when considering $(b, d, p) = (2, 12, 0)$ (which corresponds to quantization). This shows that it is relevant not to consider b fixed to 2, but on the contrary to search for the tuple (b, d, p) yielding the best performance (error versus complexity).

We finally notice that the tree adaptation algorithm is able, in some cases, to reduce the storage complexity of the approximation by finding a tree which is not the linear tree T_0. This can be seen for instance for $(b, d, p) = (2, 12, 0)$, where the storage complexity is reduced by approximately 12% with T instead of T_0. This shows that it can be useful to consider other trees than the linear tree, which is the one classically used in the literature, for instance by performing tree adaptation with Algorithm 5 of Chapter 1.

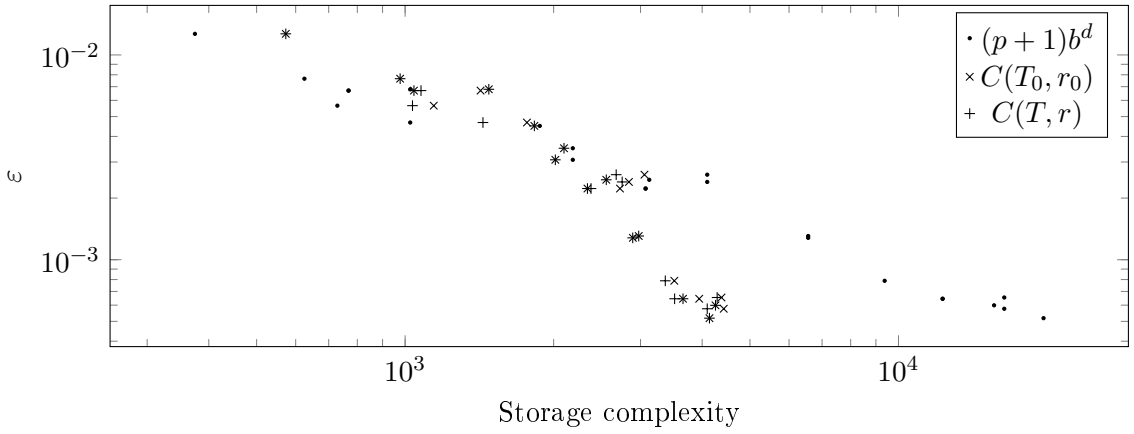


Figure 5.2: Error versus storage complexity of $\mathcal{I}_{b,d,p}f$ and of its approximation in tree-based tensor format, before and after tree adaptation, for several values of (b, d, p).

5.4 Learning tensorized functions in tree-based tensor format

In this section, we study the problem of the approximation of a tensorized function in a statistical learning context, where, contrary to the previous section, one does not choose the points at which the function to approximate is evaluated.

We seek to construct an approximation of the univariate frequency response function F described in Section 5.3, and displayed in Figure 5.1b, in a supervised learning context.

We consider the contrast function $\gamma(g, (x, y)) = (y - g(x))^2$, and we compute approximations g of $f(i_1, \ldots, i_d, y) = F(t_{b,d}(i_1, \ldots, i_d, y))$ for different tuples (b, d), with b the base and d the resolution. We choose $S = \mathbb{P}_p$ with a polynomial degree $p = 2$, and we exploit a possible sparsity in $C^{\{d+1\}}$, the parameters associated with the functions of the variable y. The approximations are obtained with Algorithm 9 of Chapter 2, with rank and tree adaptation.

b	d	$\varepsilon \cdot 10^3$	$(p+1)b^d$	$C(T_0, r_0)$	$C(T, r)$
2	10	4.67	1024	1766	1438
	12	2.60	4096	3056	2678
	14	0.57	16384	4424	4096
3	7	3.50	2187	2099	2099
	8	1.30	6561	2975	2975
	9	0.51	19683	4139	4139
4	5	6.79	1024	1478	1478
	6	2.39	4096	2842	2754
	7	0.65	16384	4378	4290
5	4	7.65	625	977	977
	5	2.45	3125	2557	2557
	6	0.59	15625	4257	4257

(a) $p = 0$.

b	d	$\varepsilon \cdot 10^3$	$(p+1)b^d$	$C(T_0, r_0)$	$C(T, r)$
2	8	6.69	768	1422	1077
	10	2.22	3072	2726	2381
	12	0.64	12288	3946	3518
3	5	5.65	729	1143	1035
	6	3.07	2187	2016	2016
	7	1.27	6561	2895	2893
4	4	6.69	768	1042	1042
	5	2.22	3072	2342	2342
	6	0.64	12288	3658	3658
5	3	12.6	375	573	573
	4	4.50	1875	1828	1828
	5	0.78	9375	3513	3368

(b) $p = 2$.

Table 5.1: Obtained results for the approximation of the tensorization of the frequency response function F, for different values of b, d and p.

The training sample is

$$S = \left\{ \left(t_{b,d}^{-1}(x_i), F(x_i) \right) \right\}_{i=1}^{n},$$

with $x_1, \ldots, x_n$ independent realizations of a uniform random variable on $[0, 1]$, and with $t_{b,d}$ defined in Equation 5.2. We consider three training sample sizes: $n = 10^2$, 10^3 and 10^4.

The obtained results are displayed in Table 5.2. We see that, with these training samples, the proposed learning algorithm is able to construct approximations in tree-based tensor format with small errors and storage complexities. We also see that the choice of the tuple (b, d) has a great impact on the obtained error and storage complexity of the approximation. Considering a resolution $d = 12$, which translates in the computation of a tensor of order $d + 1 = 13$, yields bad results compared to the other tuples (b, d) in terms of error. This

shows, once again, the importance of selecting the tuple (b, d) yielding the best results for the amount of training data available.

Figure 5.3 shows the error versus storage complexity for the results of Table 5.2, as well as for the HOSVD of $\mathcal{I}_{b,d,2}f$ for different tuples (b, d) and tolerances τ. We know that the approximation obtained by the HOSVD is a quasi-best approximation of $\mathcal{I}_{b,d,2}f$ in $\mathcal{T}_r^T(V_{b,d,\mathbb{P}_2})$ (see Section 1.6 of Chapter 1), so that the dots of Figure 5.3 give upper bounds of the best approximation error for several values of (b, d). We notice that the error as a function of the storage complexity of the approximations obtained with the proposed learning algorithms behaves similarly to this upper bound, which would indicate that the performances of the proposed learning algorithms are comparable with the ones of the HOSVD in that case.

We see in Table 5.2 that cross validation provides a good estimation of the error. However, it might not be accurate enough to select the optimal tuple (b, d) for a given n, as one can see with $n = 10^4$, $(b, d) = (4, 5)$ and $(b, d) = (4, 6)$: the approximation with the smallest error would be $(b, d) = (4, 6)$ if we used the cross-validation error estimator of the error, whereas we see that, in fact, the approximation with $(b, d) = (4, 5)$ yields the smallest error on an independent validation set. Once again, these results show the need for a robust method to select a good tuple (b, d).

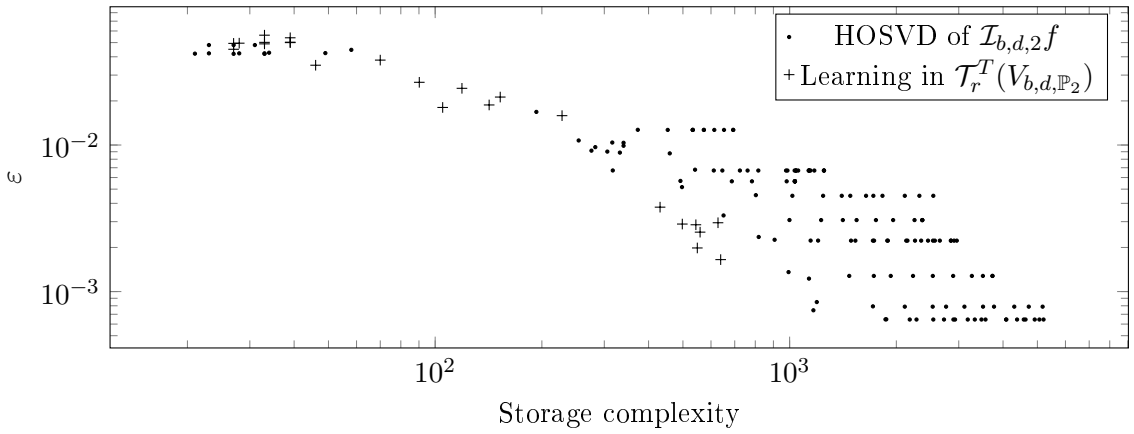


Figure 5.3: Error versus storage complexity of the approximation obtained with a HOSVD of $\mathcal{I}_{b,d,2}f$ for several values of (b, d) and truncation tolerances τ, and with the proposed learning algorithm for several values of (b, d) and n, with $p = 2$.

5.5 Conclusion

We studied, in this exploratory chapter, the possibility to apply the algorithms proposed in Chapters 1 and 2 for learning univariate functions in tree-based tensor format, using an identification of a function of the variable x with a function of the tuple $(i_1, \ldots, i_d, y)$ called tensorization.

b	d	$\varepsilon \cdot 10^2$	CV err$\cdot 10^2$	$C(T, r)$
2	10	5.61	4.53	33
	12	5.39	4.74	39
3	6	4.93	4.85	27
	7	3.79	2.34	70
4	5	4.95	4.01	28
	6	4.90	4.07	33
5	4	4.50	4.21	27
	5	3.50	3.54	46

(a) $n = 10^2$.

b	d	$\varepsilon \cdot 10^2$	CV err$\cdot 10^2$	$C(T, r)$
2	10	5.01	4.89	33
	12	5.03	4.87	39
3	6	1.58	0.71	228
	7	1.68	1.02	128
4	5	1.87	1.58	142
	6	2.43	1.97	119
5	4	1.80	1.56	105
	5	1.82	1.22	46

(b) $n = 10^3$.

b	d	$\varepsilon \cdot 10^3$	CV err$\cdot 10^3$	$C(T, r)$
2	10	2.68	2.96	521
	12	49.9	49.6	39
3	6	1.98	2.31	549
	7	1.65	1.92	639
4	5	2.88	3.34	498
	6	2.94	2.47	628
5	4	3.76	4.13	431
	5	2.54	2.42	559

(c) $n = 10^4$.

Table 5.2: Obtained results for the learning of the tensorization of the frequency response function F, for different values of b, d and n, with $p = 2$.

We highlighted in an illustration the importance of the proper choice of the tuple (b, d, p), namely the base, resolution of the tensorization and the polynomial degree, which can have a great impact on the error and storage complexity of the approximation. We also showed that the learning algorithms proposed in Chapters 1 and 2 can be applied to the approximation of tensorized functions, and can yield approximations with a good convergence of the error as a function of the storage complexity. It would then be interesting to derive from these algorithms new adaptive algorithms to learn univariate functions in tree-based tensor format, with adaptation of b and d as well as of the dimension tree T and tree-based rank r. Furthermore, robust model selection strategies ought to be developed to be able to select the best tuple (b, d) without relying on an independent validation set.

Finally, we showed, thanks to the tree adaptation Algorithm 5 of Chapter 1, that the linear tree, which is the natural tree used in the quantized tensor-train format, is not always optimal. Indeed, we found that the algorithm was sometimes able to find a representation of the approximation with a tree yielding a smaller storage complexity than with a linear tree. A more thorough analysis of the complexity reduction enabled by the tree adaptation with respect to the base b and the resolution d should give more insights about what can be expected from it.

Conclusion and future work

The main objectives of this thesis were the development of adaptive algorithms for the learning of multivariate functions in tree-based tensor format, and their application to uncertainty quantification in vibroacoustics.

In Chapter 1, after a presentation of the model class of functions in tree-based tensor format and of some useful representations and algorithms, we proposed a novel algorithm that performs dimension tree adaptation. This stochastic algorithm, which constitutes the main contribution of this chapter, optimizes the dimension tree of the representation of a function in tree-based tensor format, by randomly modifying the tree and retaining the one yielding the smallest storage complexity. In some cases, we observed that the tree thus obtained, and associated with a small storage complexity, gave information on the structure of the function, for instance the dependency structure of a probabilistic model.

Chapter 2 was devoted to the development of adaptive learning algorithms with tree-based tensor formats. The main contribution of this chapter was a learning algorithm adapting both the dimension tree (as presented in the previous chapter) and the tree-based rank, using a heuristic algorithm that incrementally increases subsets of α-ranks associated with the highest truncation errors. This adaptive learning algorithm was presented in the general context of statistical learning, and specified for the case of least-squares regression and least-squares density estimation, cases of supervised and unsupervised learning respectively. We illustrated the performances of the proposed algorithm in these two settings with several numerical experiments.

The main contribution of Chapter 3 was a learning algorithm with formats combining tree-based tensors and changes of variables. This algorithm returns a sequence of approximations with increasing effective dimension (the dimension of the range of the change of variables). The loss of multilinearity of the format caused by the introduction of the change of variables prevented from recasting the nonlinear minimization problem into a series of linear learning problems, as done in the algorithms of Chapter 2. The proposed algorithm then involved the solution of both linear and nonlinear minimization problems. We saw, by applying it on several test cases, that this algorithm was sometimes able to compute accurate approximations with a storage complexity much smaller than without considering changes of variables.

We developed in Chapter 4 a method dedicated to the uncertainty quantification in vibroacoustics. The problem to tackle was the approximation of a frequency response function, namely the sound power level of an underwater structure, which is a quantity of interest

for Naval Group, and which depends on the circular frequency ω and on random parameters x. The main contribution of this chapter was the development of a methodology to compute the envelope curve of this frequency response function, which depends on ω and x, in a transformed circular frequency space $\nu(\omega, x)$ where some signed peaks of the frequency response function are aligned. We used the algorithms proposed in Chapters 1 and 2 to compute the circular frequency transformation $\nu(\omega, x)$ and the approximation of the envelope of the frequency response function.

Finally, we showed in Chapter 5 that the algorithms proposed in Chapters 1 and 2 can be applied to the learning of univariate functions, using an identification of a univariate function with a multivariate function known as tensorization. This constitutes the main contribution of this exploratory chapter, along with the observation of the interest to consider bases b different from 2 (known as quantization) and trees different from the linear tree (which are classically considered in the literature). This approach can be used to represent non-smooth functions, such as the frequency response functions studied in Chapter 4.

Some aspects of this thesis would deserve further analyses and developments. It would be interesting for the tree adaptation algorithm of Chapter 1 to allow the transition between trees of different arities. Furthermore, the influence of the parameters of the probability distributions involved in this stochastic algorithm should be studied. Similarly, the tree-based rank adaptation proposed in Chapter 2 relies on a heuristic that would deserve a deeper analysis. This algorithm providing a sequence of approximations with increasing complexity, the development of robust model selection methods that do not rely on an independent validation sample is crucial to select the best approximation in this sequence. The problem of being able to well select a model also arises when introducing changes of variables, as done in Chapter 3. The loss of the multilinearity of the format combining tree-based tensors and changes of variables calls for the study and development of more robust algorithms able to handle the nonlinearity induced by the change of variable. The first results presented in Chapter 4, although encouraging, should be further studied, by analyzing the influence of each step of the proposed method on its accuracy. Furthermore, more data should be generated, in order to improve its global performances. We focused in this chapter on the approximation of the envelope curve of the frequency response function; however, it might be of industrial interest to be able to approximate the frequency response function itself, which is, for one realization of its random parameters, a non-smooth univariate function of the circular frequency. Following the first results of Chapter 5, it would be interesting to propose learning algorithms of univariate functions with tree-based tensor formats, with the adaptation of the dimension tree T and tree-based rank r, but also of the base b and resolution d of the tensorization.

Appendix A

Representation of probabilistic models

Contents

In this appendix, we discuss different types of representation of probability distributions using tree-based tensor formats, and provide some results on the relations between the ranks of these representations. We then provide several examples of standard probabilistic models, and discuss their representation in tree-based tensor format.

A.1 Representation of a probability distribution

The probability distribution of the random variable $X = (X_1, \dots, X_d)$ is characterized by its cumulative distribution function $F(x) = \mathbb{P}(X \leq x)$. In the following, we assume that the distribution admits a density $f(x)$ with respect to a product measure $\mu = \mu_1 \otimes \cdots \otimes \mu_d$ on $\mathbb{R}^d$ (*e.g.* the Lebesgue measure), such that

$$F(x) = \int_{\{t \leq x\}} f(t) d\mu(t).$$

This includes the case of a discrete random variable taking values in a finite or countable set $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$, with measure $\rho = \sum_{x \in \mathcal{X}} \mathbb{P}(X = x)\delta_x$, by letting $f(x) := \mathbb{P}(X = x)$ and $\mu := \sum_{x \in \mathcal{X}} \delta_x$. In this case, f is identified with an element of $\mathbb{R}^{\mathcal{X}} = \mathbb{R}^{\mathcal{X}_1 \times \cdots \times \mathcal{X}_d}$.

Proposition A.1.1. *Assume that the distribution F admits a density f with respect to a product measure μ. Then for any $\alpha \subset D$,*

$$\text{rank}_\alpha(F) \leq \text{rank}_\alpha(f).$$

Moreover, if μ is the Lebesgue measure,

$$\text{rank}_\alpha(F) = \text{rank}_\alpha(f).$$

Proof. If $f(x) = \sum_{k=1}^r f_k^\alpha(x_\alpha) f_k^{\alpha^c}(x_{\alpha^c})$, then $F(x) = \sum_{k=1}^r F_k^\alpha(x_\alpha) F_k^{\alpha^c}(x_{\alpha^c})$, $F_k^\beta(x_\beta) = \int_{\{t_\beta \leq x_\beta\}} f_k^\beta(t_\beta) d\mu_\beta(t_\beta)$ for $\beta = \alpha$ and α^c. This implies $\text{rank}_\alpha(F) \leq \text{rank}_\alpha(f)$. If μ is the Lebesgue measure and $F(x) = \sum_{k=1}^r F_k^\alpha(x_\alpha) F_k^{\alpha^c}(x_{\alpha^c})$, then almost everywhere, $f(x) = \sum_{k=1}^r f_k^\alpha(x_\alpha) f_k^{\alpha^c}(x_{\alpha^c})$ with $f_k^\beta(x_\beta) = \partial_{x_{\nu_1}} \cdots \partial_{x_{\nu_{\#\beta}}} F_k^\beta(x_\beta)$, $\beta = \{\nu_1, \ldots, \nu_{\#\beta}\}$. This implies $\text{rank}_\alpha(F) \geq \text{rank}_\alpha(f)$. □

Remark A.1.2. *Note that the above framework and results can be extended to the case where a random variable X_ν is either continuous or discrete, by letting μ_ν be either the Lebesgue measure or a discrete measure.*

For $1 \leq \nu \leq d$, let us denote by $F_\nu : \mathcal{X}_\nu \to [0,1]$ the marginal cumulative distribution function of X_ν. By Sklar's theorem, there exists a copula $C : [0,1]^d \to [0,1]$ such that

$$F(x) = C(F_1(x_1), \ldots, F_d(x_d)).$$

Proposition A.1.3. *For all $\alpha \subset D$, if C is a copula of X,*

$$\text{rank}_\alpha(F) \leq \text{rank}_\alpha(C).$$

If F admits a density f with respect to the Lebesgue measure, then X admits a unique copula C with density c and

$$\text{rank}_\alpha(F) = \text{rank}_\alpha(C) = \text{rank}_\alpha(f) = \text{rank}_\alpha(c).$$

Proof. If $C(u) = \sum_{k=1}^r C_k^\alpha(u_\alpha) C_k^{\alpha^c}(u_{\alpha^c})$, then $F(x) = \sum_{k=1}^r C_k^\alpha(u_\alpha) C_k^{\alpha^c}(u_{\alpha^c})$ with $u_\nu = F_\nu(x_\nu)$. This implies $\text{rank}_\alpha(F) \leq \text{rank}_\alpha(C)$. If F admits a density with respect to the Lebesgue measure, then $C(u) = F(F_1^{-1}(u_1), \ldots, F_d^{-1}(u_d))$, and if it writes $F(x) = \sum_{k=1}^r F_k^\alpha(x_\alpha) F_k^{\alpha^c}(x_{\alpha^c})$, then $C(u) = \sum_{k=1}^r F_k^\alpha(x_\alpha) F_k^{\alpha^c}(x_{\alpha^c})$ with $x_\nu = F_\nu^{-1}(u_\nu)$. This implies $\text{rank}_\alpha(C) \leq \text{rank}_\alpha(F)$, and therefore $\text{rank}_\alpha(F) = \text{rank}_\alpha(C)$. The other equalities are deduced from proposition A.1.1. □

A.2 Mixtures

Consider a random variable $X = (X_1, \ldots, X_d)$ which is a mixture of m random variables $Z^i = (Z_1^i, \ldots, Z_d^i)$ with weights γ_i, $1 \leq i \leq m$, such that $\sum_{i=1}^m \gamma_i = 1$. Let f and f^i denote

the densities with respect to a product measure μ of the probability distributions of X and Z^i respectively. We have

$$f(x) = \sum_{i=1}^{m} \gamma_i f^i(x).$$

From Proposition 1.3.2, we know that for any $\alpha \subset D$, $\text{rank}_\alpha(f) \leq \sum_{i=1}^m \text{rank}_\alpha(f^i)$, therefore, for any tree T,

$$\text{rank}_T(f) \leq \sum_{i=1}^{m} \text{rank}_T(f^i).$$

Assuming that Z^i has independent components Z^i_k with densities f^i_k, we have $f^i(x) = f^i_1(x_1)\cdots f^i_d(x_d)$ with $\text{rank}_\alpha(f^i) = 1$ for any α, and therefore, for any tree T, $\text{rank}_T(f) \leq m$. Assume now that the function f^i is represented in a tree based format with tree T^i. For any $\alpha \subset D$, there exists a subset T^i_α of T^i which forms a partition of α, and from Proposition 1.3.1, we have $\text{rank}_\alpha(f^i) \leq \prod_{\beta \in T^i_\alpha} \text{rank}_\beta(f^i)$, and therefore

$$\text{rank}_\alpha(f) \leq \sum_{i=1}^{m} \prod_{\beta \in T^i_\alpha} \text{rank}_\beta(f^i) := R_\alpha.$$

Then a dimension tree for the representation of f could be chosen to minimize the complexity $C(T, R)$ using the above upper bound $R = (R_\alpha)_{\alpha \in T}$ of the T-rank of f.

A.3 Markov processes

Consider a discrete time Markov process $X = (X_1, \ldots, X_d)$ whose density is given by

$$f(x) = f_{d|d-1}(x_d|x_{d-1}) \cdots f_{2|1}(x_2|x_1) f_1(x_1),$$

where f_1 is the density of X_1 and $f_{i|i-1}(\cdot|x_{i-1})$ is the density of X_i knowing $X_{i-1} = x_{i-1}$. Let m_i be the rank of the bivariate function $(t, s) \mapsto f_{i|i-1}(t|s)$, $i = 2, \ldots, d$.

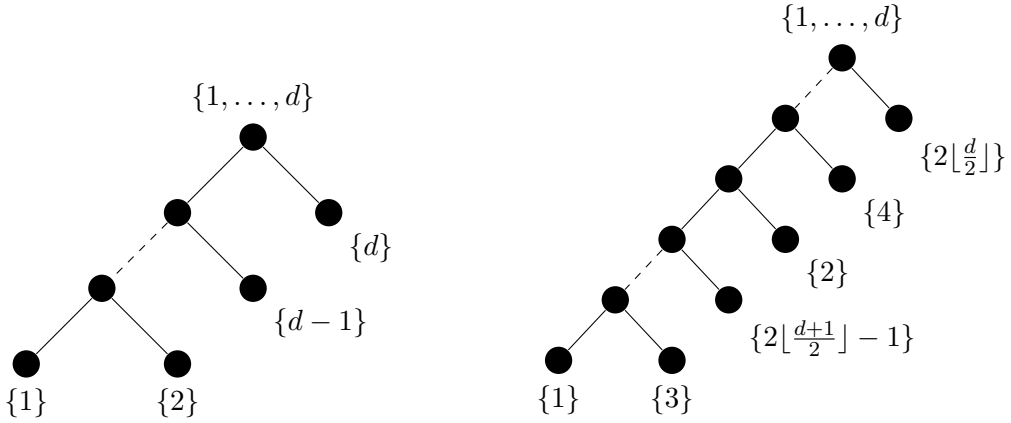
Let

$$T = \{\{1, \ldots, d\}, \{1\}, \ldots, \{d\}, \{1, 2\}, \ldots, \{1, \ldots, d-1\}\}$$

be the linear tree of Figure A.1a. We note that $\text{rank}_{\{1\}}(f) = \text{rank}(f_{2|1}) = m_2$, $\text{rank}_{\{d\}}(f) = \text{rank}(f_{d|d-1}) = m_d$ and for $2 \leq \nu \leq d-1$, $\text{rank}_{\{\nu\}}(f) \leq \text{rank}(f_{\nu|\nu-1})\text{rank}(f_{\nu+1|\nu}) = m_\nu m_{\nu+1}$. Also, for $1 < \nu < d$, we have that $\text{rank}_{\{1,\ldots,\nu\}}(f) = \text{rank}(f_{\nu+1|\nu})$. Letting $m = \max_i m_i$, we deduce that f has a representation in tree-based format with complexity in $O(m^4)$. Note that the choice of tree is here crucial. Indeed, a different ordering of variables may lead to ranks growing exponentially with the dimension d. For instance, consider the tree $\widetilde{T}$ represented in Figure A.1b, with $\widetilde{T} = \{\sigma(\alpha) : \alpha \in T\}$ with the permutation

$$\sigma = (1, 3, \ldots, 2\left\lfloor \frac{d+1}{2} \right\rfloor - 1, 2, 4, \ldots, 2\left\lfloor \frac{d}{2} \right\rfloor).$$

For $\alpha = \{1, 3, \dots, 2k+1\}$, with $k \leq \lfloor \frac{d+1}{2} \rfloor - 1$, we have $\operatorname{rank}_\alpha(f) \leq m_2 m_3 \cdots m_{2k+2} \leq m^{2k+1}$ if $2k+1 < d$, and $\operatorname{rank}_\alpha(f) \leq m_2 m_3 \cdots m_{2k+1} \leq m^{2k}$ if $2k+1 = d$. Therefore, the representation in the corresponding tree-based format has a complexity in $O(m^{2d-2})$. Example A.3.1 presents a Markov process for which the tree-based rank exhibits such a behavior. Therefore, when the structure of the Markov process is not known, a procedure for finding a suitable tree should be used (see Section 1.8 for the description of the proposed tree optimization algorithm).



(a) Linear dimension tree T.

(b) Linear dimension tree $\widetilde{T}$.

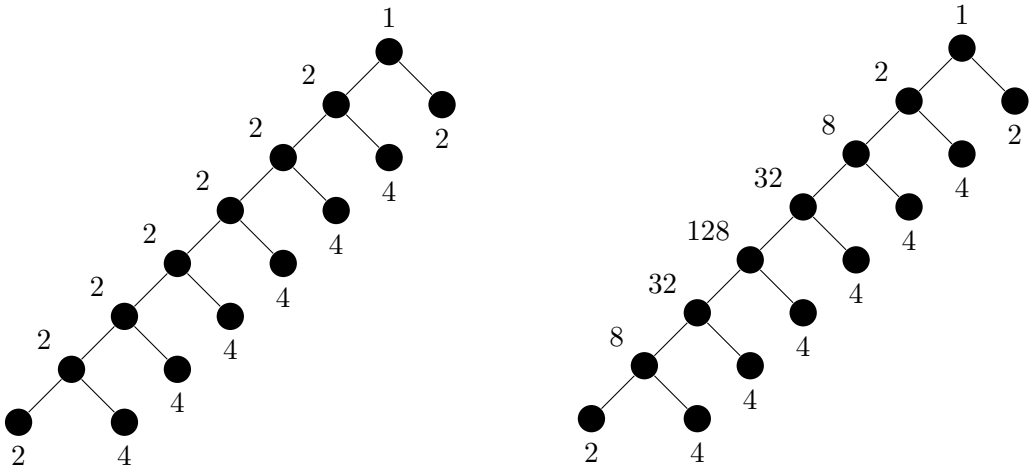
Figure A.1: Examples of linear dimension trees.

Example A.3.1 (Discrete state space Markov process)**.** *We consider the discrete time discrete state space Markov process $X = (X_1, \dots, X_8)$, where each random variable X_ν takes values in $\mathcal{X}_\nu = \{1, \dots, 5\}$. The distribution of X writes*

$$f(i_1, \dots, i_8) \coloneqq \mathbb{P}(X_1 = i_1, \dots, X_8 = i_8) = f_{8|7}(i_8|i_7) \cdots f_{2|1}(i_2|i_1) f_1(i_1)$$

with $f_1(i_1) = 1/5$ for all $i_1 \in \mathcal{X}_1$, and for $\nu = 1, \dots, d-1$, $f_{\nu+1|\nu}(i_{\nu+1}|i_\nu) = P^\nu_{i_\nu, i_{\nu+1}}$ the $(i_\nu, i_{\nu+1})$ component of a randomly chosen rank-2 transition matrix P^ν. We then have $\operatorname{rank}(f_{\nu+1|\nu}) = m = 2$ *for $\nu = 1, \dots, d-1$.*

We first compute a representation of f in tree-based format with the tree T depicted in Figure A.1a (using a truncation algorithm at precision 10^{-13}, as described in Section 1.6), the obtained α-ranks are shown in Figure A.2a. We then compute a representation of f in tree-based format with the tree $\widetilde{T}$ depicted in Figure A.1b (with the same precision 10^{-13}), to obtain the α-ranks shown in Figure A.2b. We see that $\max_{\alpha \in T} \operatorname{rank}_\alpha(f) = 4 = m^2$ *whereas* $\max_{\alpha \in \widetilde{T}} \operatorname{rank}_\alpha(f) = 128 = 2^7 = m^{d-1}$. *As a consequence, the storage complexity of the representation is equal to* 240 *with T, and to* 35088 *with $\widetilde{T}$, more than* 146 *times larger.*



(a) α-ranks when using T.

(b) α-ranks when using $\widetilde{T}$.

Figure A.2: Obtained α-ranks when representing the Markov process of Example A.3.1 in tree-based format with two different linear dimension trees.

A.4 Graphical models

Let us consider a graphical model with a density of the form

$$f(x) = \prod_{\beta \in \mathcal{C}} g_\beta(x_\beta)$$

where $\mathcal{C} \subset 2^D$ represents the cliques of a graph G with nodes $\{1\}, \dots, \{d\}$.

Consider $\alpha \subset D$. First note that if $\alpha \in \mathcal{C}$, then $\text{rank}_\alpha(g_\alpha) = 1$. Also, for a clique β such that either $\beta \subset \alpha$ or $\beta \subset \alpha^c$, $\text{rank}_\alpha(g_\beta) = 1$. Then let $\mathcal{C}_\alpha$ be the set of cliques that intersect both α and α^c,

$$\mathcal{C}_\alpha = \{\beta \in \mathcal{C} : \beta \cap \alpha \neq \emptyset, \beta \cap \alpha^c \neq \emptyset\}.$$

Since $\mathcal{C} \setminus \mathcal{C}_\alpha = \{\beta \in \mathcal{C} : \beta \subset \alpha^c \text{ or } \beta \subset \alpha\}$, and from Proposition 1.3.2, we have

$$\text{rank}_\alpha(f) = \text{rank}_\alpha(\prod_{\beta \in \mathcal{C}_\alpha} g_\beta) \leq \prod_{\beta \in \mathcal{C}_\alpha} \text{rank}_\alpha(g_\beta).$$

Assuming that the α-ranks of all functions g_β are bounded by m, we have

$$\text{rank}_\alpha(f) \leq m^{\#\mathcal{C}_\alpha} = R_\alpha.$$

For the representation of f in tree-based tensor format, a tree T could be chosen such that it minimizes the complexity $C(T, R)$, with $R = (R_\alpha)_{\alpha \in T}$ the above upper bound of the T-rank of f.

Example A.4.1. *The motivating example of Section 1.8.1 corresponds to a graphical model (whose graph is displayed in Figure 2.15), in dimension $d = 10$ with $f(x) = \mathbb{P}(X = x)$. This example shows the importance of the choice of the dimension tree on the storage complexity*

of the representation, as well as that the dimension tree yielding the smallest storage complexity can carry information about the dependence structure of the probabilistic model, by containing nodes associated with cliques of the graph.

Appendix B

Orthonormal polynomials

Contents

We begin by presenting in this appendix some properties of orthonormal polynomials, then we introduce the three-term recurrence relation used to construct them and some usual families of orthonormal polynomials. We then propose a methodology to construct a family of polynomials orthonormal with respect to a measure estimated from a sample. Finally, we show how to deduce from the family of polynomials orthonormal with respect to the measure of a random variable Z, the family of polynomials orthonormal with respect to the measure of the random variable $X = sZ + b$, with b and s real numbers, that we call shifted orthonormal polynomials.

B.1 Properties of orthonormal polynomials

A family of polynomials $\{p_i\}_{i\geq 0}$ is said to be orthonormal with respect to the measure μ if

$$\langle p_i, p_j \rangle \coloneqq \int_{\mathbb{R}} p_i(x)p_j(x)\mathrm{d}\mu(x) = \delta_{ij},$$

with δ_{ij} the Kronecker delta.

We denote by X the random variable with probability measure μ and with values in $\mathcal{X} \subset \mathbb{R}$, and by m_n the n-th moment of X, such that

$$m_n = \int_{\mathcal{X}} x^n \mathrm{d}\mu(x).$$

The polynomial p_n, $n \geq 0$, then writes

$$p_n(x) = c_n \det \begin{pmatrix} m_0 & m_1 & \cdots & m_n \\ m_1 & m_2 & \cdots & m_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n-1} & m_n & \cdots & m_{2n-1} \\ 1 & x & \cdots & x^n \end{pmatrix},$$

with c_n a normalizing constant such that $\langle p_n, p_n \rangle = 1$.

Another way to compute p_n, $n \geq 0$, is to use a Gram-Schmidt procedure. Beginning with $\tilde{p}_0(x) = p_0(x) = 1$, the non-normalized polynomial $\tilde{p}_{n+1}$ writes

$$\tilde{p}_{n+1}(x) = q_{n+1}(x) - \sum_{k=0}^{n} \frac{\langle q_{n+1}, \tilde{p}_k \rangle}{\langle \tilde{p}_k, \tilde{p}_k \rangle} \tilde{p}_k(x)$$

with $q_{n+1}(x) = x^{n+1}$. The normalized polynomial p_{n+1} writes

$$p_{n+1}(x) = \tilde{p}_{n+1}(x) \langle \tilde{p}_{n+1}, \tilde{p}_{n+1} \rangle^{-1/2}.$$

However, in practice, due to rounding errors, this procedure might yield polynomials that are not exactly orthogonal. We then use another way of generating families of orthogonal polynomials yielding better results: the three-term recurrence relation.

B.2 Three-term recurrence relation for orthogonal polynomials

It can be shown that orthogonal polynomials of a same family follow the recurrence formula

$$\tilde{p}_{n+1}(x) = (x - \alpha_n)\tilde{p}_n(x) - \beta_n \tilde{p}_{n-1}(x), \quad n \geq 0,$$

with $\tilde{p}_{-1}(x) = 0$, $\tilde{p}_0(x) = 1$,

$$\alpha_n = \frac{\langle \tilde{p}_n, x\tilde{p}_n \rangle}{\langle \tilde{p}_n, \tilde{p}_n \rangle}, \quad \beta_n = \frac{\langle \tilde{p}_n, \tilde{p}_n \rangle}{\langle \tilde{p}_{n-1}, \tilde{p}_{n-1} \rangle}. \tag{B.1}$$

and $p_{n+1}(x) = \tilde{p}_{n+1}(x)/\gamma_{n+1}$, with $\gamma_{n+1} = \langle \tilde{p}_{n+1}, \tilde{p}_{n+1} \rangle^{1/2}$.

This recurrence formula is interesting because the coefficients α_n, β_n and γ_n can sometimes be known explicitly (some examples can be found in Section B.3). This way of generating

the polynomials can then yield better results than the Gram-Schmidt procedure presented above, by reducing the rounding errors in the computation of the inner products.

An interesting property is as follows: the roots of p_n are the eigenvalues of the Jacobi matrix

$$J = \begin{pmatrix} \alpha_0 & \sqrt{\beta_1} & & & & & \\ \sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & & & & \\ & \sqrt{\beta_2} & \alpha_2 & \sqrt{\beta_3} & & & \\ & & \sqrt{\beta_3} & \ddots & \ddots & & \\ & & & \ddots & \alpha_{n-3} & \sqrt{\beta_{n-2}} & \\ & & & & \sqrt{\beta_{n-2}} & \alpha_{n-2} & \sqrt{\beta_{n-1}} \\ & & & & & \sqrt{\beta_{n-1}} & \alpha_{n-1} \end{pmatrix}. \tag{B.2}$$

This is useful for instance to compute the Gauss quadrature points associated with a measure μ, as it is done in Section B.4.

B.3 Some classical orthonormal polynomials

We present in this section three examples of families of orthonormal polynomials, for which the recurrence coefficients are known explicitly.

Legendre polynomials. The Legendre polynomials are associated with the standard uniform random variable $X \sim \mathcal{U}(-1,1)$ with values in $\mathcal{X} = [-1,1]$. They are orthonormal with respect to the measure μ such that $\mathrm{d}\mu(x) = \frac{1}{2}\mathbf{1}_{x\in\mathcal{X}}\mathrm{d}x$. The coefficients of the three-term recurrence relation and the norms write, for $n \geq 0$,

$$\alpha_n = 0, \quad \beta_n = \frac{n^2}{4n^2-1}, \quad \gamma_n = \sqrt{\frac{1}{2n+1}}\frac{2^n n!^2}{(2n)!}.$$

Hermite polynomials. The Hermite polynomials are associated with the standard normal random variable $X \sim \mathcal{N}(0,1)$ with values in $\mathcal{X} = \mathbb{R}$. They are orthonormal with respect to the measure μ such that $\mathrm{d}\mu(x) = \frac{1}{2\pi}\exp(-x^2/2)\mathbf{1}_{x\in\mathcal{X}}\mathrm{d}x$. The coefficients of the three-term recurrence relation and the norms write, for $n \geq 0$,

$$\alpha_n = 0, \quad \beta_n = n, \quad \gamma_n = \sqrt{n!}.$$

Jacobi polynomials. The Jacobi polynomials are associated with the beta random variable $X \sim \mathrm{Beta}(\alpha+1, \beta+1)$ with values in $\mathcal{X} = [-1,1]$, with $\alpha, \beta > 0$. They are orthonormal with respect to the measure μ such that $\mathrm{d}\mu(x) = (1-x)^\alpha(1+x)^\beta\mathbf{1}_{x\in\mathcal{X}}\mathrm{d}x$. The coefficients

of the three-term recurrence relation and the norms (found in [57]) write, for $n \geq 0$,

$$\alpha_n = \frac{\beta^2 - \alpha^2}{(\alpha+\beta+2n)(\alpha+\beta+2n+2)},$$
$$\beta_n = \frac{4n(n+\alpha)(n+\beta)(n+\alpha+\beta)}{(2n+\alpha+\beta)^2(2n+\alpha+\beta+1)(2n+\alpha+\beta-1)},$$
$$\gamma_n = \sqrt{\frac{2^{\alpha+\beta+2n+1}\Gamma(\alpha+n+1)\Gamma(\beta+n+1)\Gamma(\alpha+\beta+n+1)n!}{\Gamma(\alpha+\beta+2n+1)\Gamma(\alpha+\beta+2n+2)}},$$

with $\Gamma(z) \coloneqq \int_0^\infty x^{z-1}\exp(-x)\mathrm{d}x$, $\Re(z) > 0$, the gamma function.

The Chebyshev polynomials of the first kind are a particular case of the Jacobi polynomials with $\alpha = \beta = -1/2$. The roots of the Chebyshev polynomials of the first kind, that can be found by computing the eigenvalues of the associated Jacobi matrix (B.2), are widely used as interpolation points, that give nice properties to the interpolation for polynomial approximation.

The Legendre polynomials are also (up to a coefficient) a particular case of the Jacobi polynomials, with $\alpha = \beta = 0$.

B.4 Empirical orthonormal polynomials

In some cases, for instance in Chapter 3, one wishes to create a family of polynomials orthonormal with respect to the probability density function of a random variable X, estimated from a sample of X. We propose in this section a methodology to build such a family, that we call empirical orthonormal polynomials. This methodology involves two major steps: first, estimate the density of the unknown measure from the sample, using a kernel density estimator, then, compute the coefficients of the three-term recurrence relation.

B.4.1 Estimation of the density of the random variable

We assume that we have at our disposal a sample $S = \{(x_i - m)/\sigma\}_{i=1}^n$, with x_i a realization of a random variable X with unknown density f_X, and where $m = \frac{1}{n}\sum_{i=1}^n x_i$ and $\sigma^2 = \frac{1}{n-1}\sum_{i=1}^n (x_i - m)^2$ are the empirical mean and standard deviation of S, respectively.

We estimate the density f_Z of the standard random variable $Z = (X - \mathbb{E}(X))/\sqrt{\mathbb{V}(Z)}$ with the kernel density estimator

$$\hat{f}_Z(z) = \frac{1}{nh}\sum_{k=1}^n K\left(\frac{z - z_k}{h}\right)$$

with $z_k = (x_k - m)/\sigma$, $k = 1, \ldots, n$, where K is a kernel (defining a probability density function), and where h is called the bandwidth, which is a real positive number that needs to be chosen.

Here, we use the Gaussian kernel

$$K(z) = \frac{1}{2\pi}\exp(-\frac{z^2}{2}),$$

and h is selected following Scott's rule [44]:

$$h = 3.5n^{-\frac{1}{3}}.$$

This rule is optimal for samples from normal random variables, however, it is an efficient way of computing a bandwidth that yields sufficiently good results for many distributions and applications.

B.4.2 Estimation of the coefficients of the three-term recurrence relation

Once the density f_Z of Z has been approximated by $\hat{f}_Z$, one can compute the coefficients of the three-term recurrence relation using Equation (B.1).

To estimate the different inner products appearing in (B.1), we use a Gaussian quadrature rule: consider the measure ν such that $\mathrm{d}\nu(z) = K(z)\mathrm{d}z$, then

$$\int_{\mathbb{R}} g(z)\mathrm{d}\nu(z) = \int_{\mathbb{R}} g(z)K(z)\mathrm{d}z \approx \sum_{l=1}^{N} w_l g(\xi_l) =: Q_N^{\nu}(g),$$

with $\{\xi_l\}_{l=1}^N$ and $\{w_l\}_{l=1}^N$ the points and weights of the Gauss quadrature rule, respectively. The quadrature points are the roots of the polynomials of degrees 0 to $N-1$, orthogonal with respect to ν, and are computed by finding the eigenvalues of the associated Jacobi matrix (B.2). The quadrature weights are related to the eigenvectors V^i of (B.2): $w_i = (V_1^i)^2/\|V^i\|_2$, $i = 1, \ldots, N$. This way of computing the quadrature points and weights is known as the Golub-Welsch algorithm [58]. This quadrature rule is exact for polynomials of degree $\delta \leq 2N-1$.

The inner products in (B.1) can then be computed as follows:

$$\begin{aligned}
\int_{\mathbb{R}} \tilde{p}_i(z)\tilde{p}_j(z)\hat{f}_Z(z)\mathrm{d}z &= \frac{1}{nh}\sum_{k=1}^{n}\int_{\mathbb{R}} \tilde{p}_i(z)\tilde{p}_j(z)K\left(\frac{z-z_k}{h}\right)\mathrm{d}z, \\
&= \frac{1}{n}\sum_{k=1}^{n}\int_{\mathbb{R}} \tilde{p}_i(h\xi+z_k)\tilde{p}_j(h\xi+z_k)K(\xi)\mathrm{d}\xi, \\
&= \frac{1}{n}\sum_{k=1}^{n} Q_N^{\nu}(\tilde{p}_i(h\cdot+z_k)\tilde{p}_j(h\cdot+z_k)),
\end{aligned}$$

with $2N-1 \geq i+j$ so that the quadrature rule is exact.

Here, we consider a Gaussian Kernel: the quadrature points are the roots of the Hermite polynomials; this special form of quadrature is called a Gauss-Hermite quadrature.

Algorithm 15 presents the proposed way of computing the coefficients of the three-term recurrence relation.

Algorithm 15 Computation of the recurrence coefficients for orthonormal polynomials.

Inputs: density $\hat{f}_Z$ with respect to which the polynomials are orthonormal, estimated with a kernel density estimator with kernel K, maximal degree δ of the polynomials

Outputs: coefficients α_i, β_i and γ_i, $i = 0, \ldots, \delta - 1$

1: compute the Gaussian quadrature points $\{\xi_i\}_{i=1}^{\delta}$ and weights $\{w_i\}_{i=1}^{\delta}$ associated with the measure ν such that $\mathrm{d}\nu(z) = K(z)\mathrm{d}(z)$, to exactly integrate polynomials of degree less or equal than $2\delta - 1$
2: set $\beta_0 = 0$ and $\gamma_0 = 1$
3: compute $\alpha_0 = \int_{\mathbb{R}} z\hat{f}_Z(z)\mathrm{d}z$ with the Gaussian quadrature rule
4: set $\tilde{p}_{-1}(z) = 0$
5: set $\tilde{p}_0(z) = 1$
6: **for** $i = 1, \ldots, \delta - 1$ **do**
7: compute $\tilde{p}_i(z) = (z - \alpha_{i-1})\tilde{p}_{i-1}(z) - \beta_{i-1}\tilde{p}_{i-2}(z)$
8: compute $\gamma_i = \sqrt{\langle \tilde{p}_i, \tilde{p}_i \rangle}$ with the Gaussian quadrature rule
9: compute $\alpha_i = \langle \tilde{p}_i, z\tilde{p}_i \rangle / \gamma_i^2$ with the Gaussian quadrature rule
10: compute $\beta_i = \gamma_i^2 / \gamma_{i-1}^2$
11: **end for**
12: compute $\tilde{p}_\delta(z) = (z - \alpha_{\delta-1})\tilde{p}_{\delta-1}(z) - \beta_{\delta-1}\tilde{p}_{\delta-2}(z)$

Remark B.4.1. *In practice, because of the finite precision arithmetic involved, the polynomials obtained with Algorithm 15 might not be exactly orthonormal, in the sense that $\langle p_i, p_j \rangle$ might not be equal to δ_{ij}, especially for large i, j. It is then necessary to ensure that $\langle p_i, p_j \rangle$ does not deviate "too much" from δ_{ij}. This deviation can be used as a stopping criterion in Algorithm 15.*

B.5 Shifted orthonormal polynomials

The coefficients of the three-term recurrence relation are exactly known for some usual standard random variables, a few examples being presented in Section B.3. Knowing the orthonormal polynomials p_i associated with a random variable Z, one can easily compute the orthonormal polynomials $\hat{p}_i$ associated with the random variable $X = sZ + b$:

$$\hat{p}_i(x) = p_i\left(\frac{x - b}{s}\right)$$

with b and s real numbers. We call these new polynomials shifted orthonormal polynomials. If X is with values in $\mathcal{X} = [x_l, x_r]$, then Z is with values in $\mathcal{Z} = [(x_l - b)/s, (x_r - b)/s]$.

Example B.5.1 (Uniform random variable on $[x_l, x_r]$)**.** *If $X \sim \mathcal{U}(x_l, x_r)$ with $\mathcal{X} = [x_l, x_r]$, then $b = (x_l + x_r)/2$, $s = (x_r - x_l)/2$ and $Z \sim \mathcal{U}(-1, 1)$ with $\mathcal{Z} = [-1, 1]$, whose associated orthonormal polynomials are the Legendre polynomials.*

Example B.5.2 (Normal random variable with mean m and standard deviation σ)**.** *If $X \sim \mathcal{N}(m, \sigma^2)$ with $\mathcal{X} = \mathbb{R}$, then $b = m$, $s = \sigma$ and $Z \sim \mathcal{N}(0,1)$ with $\mathcal{Z} = \mathbb{R}$, whose associated orthonormal polynomials are the Hermite polynomials.*

Denoting by f_X the density of X and by f_Z the density of Z, the moment $\mathbb{E}(\hat{p}_i(X)\hat{p}_j(X))$ writes, for $i, j \geq 0$,

$$\begin{aligned}
\mathbb{E}(\hat{p}_i(X)\hat{p}_j(X)) &= \int_{\mathcal{X}} \hat{p}_i(x)\hat{p}_j(x) f_X(x)\mathrm{d}x, \\
&= \int_{\mathcal{X}} p_i\left(\frac{x-b}{s}\right) p_j\left(\frac{x-b}{s}\right) f_X(x)\mathrm{d}x, \\
&= \int_{\mathcal{Z}} p_i(z)p_j(z) s f_X(sz+b)\mathrm{d}z, \\
&= \int_{\mathcal{Z}} p_i(z)p_j(z) f_Z(z)\mathrm{d}z, \\
&= \delta_{ij},
\end{aligned}$$

which shows the orthonormality of the family.

Finally, denoting by r_j^i the roots of p_i, and by $\hat{r}_j^i$ the roots of $\hat{p}_i$, we have

$$\hat{r}_j^i = s r_j^i + b, \quad 1 \leq j \leq i.$$

Bibliography

[1] E. Winsberg. "Computer Simulations in Science". In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Spring 2019. Metaphysics Research Lab, Stanford University, 2019. URL: https://plato.stanford.edu/archives/spr2019/entries/simulations-science/.

[2] R. Ghanem, D. Higdon, and H. Owhadi. *Handbook of uncertainty quantification*. Vol. 6. Springer, 2017.

[3] P. Temarel, W. Bai, A. Bruns, Q. Derbanne, D. Dessi, S. Dhavalikar, N. Fonseca, T. Fukasawa, X. Gu, A. Nestegård, et al. "Prediction of wave-induced loads on ships: Progress and challenges". In: *Ocean Engineering* 119 (2016), pp. 274–308.

[4] C. Leblond and J.-F. Sigrist. "A reduced basis approach for the parametric low frequency response of submerged viscoelastic structures". In: *Finite Elements in Analysis and Design* 119 (2016), pp. 15–29.

[5] M. Chevreuil, C. Leblond, A. Nouy, Y. Tampango, and J.-F. Sigrist. "Une méthode de réduction de modèle basée sur l'échantillonnage pour le calcul d'une réponse vibro-acoustique aléatoire en basse fréquence". In: *12e Colloque national en calcul des structures*. 2015.

[6] J.-F. Sigrist, C. Leblond, and S. Iakovlev. "Une modélisation semi-analytique du comportement dynamique de coques élastiques immergées". In: *9e Colloque national en calcul des structures,CSMA*. 2009.

[7] R. DeVore, G. Kerkyacharian, D. Picard, and V. Temlyakov. "Approximation methods for supervised learning". In: *Foundations of Computational Mathematics* 6.1 (2006), pp. 3–58.

[8] O. Bousquet, S. Boucheron, and G. Lugosi. "Introduction to statistical learning theory". In: *Summer School on Machine Learning*. Springer. 2003, pp. 169–207.

[9] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer New York, 2009.

[10] F. Bach, R. Jenatton, J. Mairal, G. Obozinski, et al. "Optimization with sparsity-inducing penalties". In: *Foundations and Trends® in Machine Learning* 4.1 (2012), pp. 1–106.

[11] G. Blatman and B. Sudret. "Adaptive sparse polynomial chaos expansion based on least angle regression". In: *Journal of Computational Physics* 230.6 (2011), pp. 2345–2367.

[12] D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard. "Wavelet shrinkage: asymptopia?" In: *Journal of the Royal Statistical Society: Series B (Methodological)* (1995), pp. 301–369.

[13] D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard. "Density estimation by wavelet thresholding". In: *The Annals of Statistics* (1996), pp. 508–539.

[14] M. I. Jordan et al. "Graphical models". In: *Statistical Science* 19.1 (2004), pp. 140–155.

[15] A. Pinkus. *Ridge Functions*. Cambridge Tracts in Mathematics. Cambridge University Press, 2015.

[16] J. H. Friedman and W. Stuetzle. "Projection pursuit regression". In: *Journal of the American statistical Association* 76.376 (1981), pp. 817–823.

[17] L. Grasedyck, D. Kressner, and C. Tobler. "A literature survey of low-rank tensor approximation techniques". In: *GAMM-Mitteilungen* 36.1 (2013), pp. 53–78.

[18] T. G. Kolda and B. W. Bader. "Tensor decompositions and applications". In: *SIAM review* 51.3 (2009), pp. 455–500.

[19] A. Falcó, W. Hackbusch, and A. Nouy. "Tree-based tensor formats". In: *SeMA Journal* (2018).

[20] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, and D. Mandic. "Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions". In: *Foundations and Trends® in Machine Learning* 9.4-5 (2016), pp. 249–429.

[21] W. Hackbusch and S. Kuhn. "A New Scheme for the Tensor Representation". English. In: *Journal of Fourier analysis and applications* 15.5 (2009), pp. 706–722.

[22] I. Oseledets and E. Tyrtyshnikov. "Breaking the curse of dimensionality, or how to use SVD in many dimensions". In: *SIAM Journal on Scientific Computing* 31.5 (2009), pp. 3744–3759.

[23] A. Falcó and W. Hackbusch. "On Minimal Subspaces in Tensor Representations". English. In: *Foundations of Computational Mathematics* 12 (2012), pp. 765–803.

[24] M. Ali and A. Nouy. "Singular Value Decomposition in Sobolev Spaces". In: *ArXiv e-prints* (2018).

[25] A. Uschmajew and B. Vandereycken. "The geometry of algorithms using hierarchical tensors". In: *Linear Algebra and its Applications* 439.1 (2013), pp. 133–166.

[26] A. Falco, W. Hackbusch, and A. Nouy. "Geometric Structures in Tensor Representations (Final Release)". In: *ArXiv e-prints* (2015).

[27] A. Falcó, W. Hackbusch, and A. Nouy. "On the Dirac-Frenkel Variational Principle on Tensor Banach Spaces". In: *Foundations of Computational Mathematics* (2018).

[28] S. Holtz, T. Rohwedder, and R. Schneider. "On manifolds of tensors of fixed TT-rank". English. In: *Numerische Mathematik* 120.4 (2012), pp. 701–731.

[29] V. Khrulkov, A. Novikov, and I. Oseledets. "Expressive power of recurrent neural networks". In: *International Conference on Learning Representations*. 2018.

[30] N. Cohen, O. Sharir, and A. Shashua. "On the expressive power of deep learning: A tensor analysis". In: *Conference on Learning Theory*. 2016, pp. 698–728.

[31] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[32] W. Hackbusch. *Tensor spaces and numerical tensor calculus*. Vol. 42. Springer series in computational mathematics. Heidelberg: Springer, 2012, pp. xxiv, 500.

[33] E. Grelier, A. Nouy, and M. Chevreuil. "Learning with tree-based tensor formats". Preprint available on arXiv. 2018. URL: https://arxiv.org/abs/1811.04455.

[34] A. Nouy and E. Grelier. "Learning high-dimensional probability distributions using tree tensor networks". Unpublished. 2019.

[35] B. N. Khoromskij. "O(dlogN)-Quantics Approximation of N-d Tensors in High-Dimensional Numerical Modeling". In: *Constructive Approximation* 34.2 (2011), pp. 257–280.

[36] E. Nikolaidis and P. Kaplan. *Uncertainties in stress analysis on marine structures*. Tech. rep. Virginia Polytechnic Inst And State Univ Blacksburg Dept Of Aerospace and Ocean Engineering, 1991.

[37] A. Nouy. "Higher-order principal component analysis for the approximation of tensors in tree-based low-rank formats". In: *Numerische Mathematik* (2019).

[38] S. Arlot and A. Celisse. "A survey of cross-validation procedures for model selection". In: *Statistics surveys* 4 (2010), pp. 40–79.

[39] O. Chapelle, V. Vapnik, and Y. Bengio. "Model selection for small sample regression". In: *Machine Learning* 48.1-3 (2002), pp. 9–23.

[40] A. Celisse. "Optimal cross-validation in density estimation with the L^2-loss". In: *Annals of Statistics 2014, Vol. 42, No. 5, 1879-1910* (2008). arXiv: 0811.0802v4 [math.ST].

[41] S. Mallat and Z. Zhang. "Matching pursuits with time-frequency dictionaries". In: *Signal Processing, IEEE Transactions on* 41.12 (1993), pp. 3397–3415.

[42] A. Belloni and V. Chernozhukov. "Least squares after model selection in high-dimensional sparse models". In: *Bernoulli* 19.2 (2013), pp. 521–547.

[43] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. "Least angle regression". In: *The Annals of statistics* 32.2 (2004), pp. 407–499.

[44] D. W. Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[45] J. H. Friedman et al. "Multivariate adaptive regression splines". In: *The annals of statistics* 19.1 (1991), pp. 1–67.

[46] A. Bovis. *Hydrodynamique navale : le sous-marin*. Les cours. Les Presses de l'ENSTA, 2011.

[47] R. E. Francois and G. R. Garrison. "Sound absorption based on ocean measurements. Part II: Boric acid contribution and equation for total absorption". In: *The Journal of the Acoustical Society of America* 72.6 (1982), pp. 1879–1890.

[48] C. Leblond, M. Abbas, J. Vernet-Castex, S. Prigent, and J.-F. Sigrist. "Application de la méthode de Base Réduite pour la réponse vibroacoustique de structures immergées avec paramètres incertains". In: *13e colloque national en calcul des structure*. 2017.

[49] J.-F. Sigrist. *Fluid-structure interaction: an introduction to finite element coupling*. John Wiley & Sons, 2015.

[50] V. Yaghoubi, S. Marelli, B. Sudret, and T. Abrahamsson. "Sparse polynomial chaos expansions of frequency response functions using stochastic frequency transformation". In: *Probabilistic engineering mechanics* 48 (2017), pp. 39–58.

[51] Y. Maday, N. C. Nguyen, A. T. Patera, and G. S. Pau. "A general, multipurpose interpolation procedure: the magic points". In: *Communications on Pure and Applied Analysis* (2009).

[52] F. Bonizzoni, F. Nobile, and I. Perugia. *Convergence analysis of Padé approximations for Helmholtz frequency response problems*. Tech. rep. 2017.

[53] B. Gustavsen and A. Semlyen. "Rational approximation of frequency domain responses by vector fitting". In: *IEEE Transactions on power delivery* 14.3 (1999), pp. 1052–1061.

[54] I. V. Oseledets. "Approximation of 2ˆd\times2ˆd matrices using tensor decomposition". In: *SIAM Journal on Matrix Analysis and Applications* 31.4 (2010), pp. 2130–2145.

[55] A. V. Chertkov, I. V. Oseledets, and M. V. Rakhuba. "Robust discretization in quantized tensor train format for elliptic problems in two dimensions". In: *arXiv preprint arXiv:1612.01166* (2016).

[56] V. Kazeev and C. Schwab. "Quantized tensor-structured finite elements for second-order elliptic PDEs in two dimensions". In: *Numerische Mathematik* 138.1 (2018), pp. 133–190.

[57] Y. Chen and M. Ismail. "Jacobi polynomials from compatibility conditions". In: *Proceedings of the American Mathematical Society* 133.2 (2005), pp. 465–472.

[58] G. H. Golub and J. H. Welsch. "Calculation of Gauss quadrature rules". In: *Mathematics of computation* 23.106 (1969), pp. 221–230.

Titre : Apprentissage statistique avec des formats de tenseurs basés sur des arbres– Application à la quantification d'incertitudes en vibroacoustique

Mots clés : Apprentissage statistique, approximation en haute dimension, réseaux de tenseurs basés sur des arbres, formats de tenseurs hiérarchiques, algorithmes adaptatifs, quantification d'incertitudes

Résumé : De nombreux problèmes nécessitent l'évaluation de modèles paramétrés complexes pour de nombreuses valeurs des paramètres, en particulier pour la quantification d'incertitudes. Quand le modèle est coûteux à évaluer, il est souvent approximé par un autre modèle, moins coûteux à évaluer.

L'objectif de cette thèse est de développer des méthodes d'apprentissage statistique utilisant des classes de fonctions au format de tenseurs basés sur des arbres pour l'approximation de fonctions en haute dimension, pour l'apprentissage supervisé et non supervisé. Ces classes de fonctions, qui sont structurés par rangs et paramétrées par un réseau de tenseurs de faible ordre à structure d'arbre, peuvent être interprétées comme des réseaux de neurones profonds avec une architecture et des fonctions d'activation particulières. L'approximation est obtenue par minimisation du risque empirique sur l'ensemble des fonctions au format de tenseurs basés sur des arbres.

Pour l'approximation de fonctions en haute dimension, ou quand peu d'information sur la fonction est disponible, la classe de fonctions doit être soigneusement choisie. Nous proposons des algorithmes d'apprentissage stables qui adaptent l'arbre et les rangs et sélectionnent le modèle en s'appuyant sur des estimateurs de validation croisée. De plus, certaines fonctions peuvent n'exhiber une structure de faible rang qu'après un changement de variables adapté. Dans de tels cas, nous proposons des algorithmes d'apprentissage adaptatifs avec des classes de fonctions combinant formats de tenseurs basés sur des arbres et changements de variables.

Les algorithmes proposés sont appliqués à la quantification d'incertitudes en vibroacoustique.

Cette thèse est incluse dans le Joint Laboratory of Marine Technology entre Naval Group, Centrale Nantes et l'Université de Nantes, et dans le projet Eval-PI.

Title: Learning with tree-based tensor formats–Application to uncertainty quantification in vibroacoustics

Keywords: Statistical learning, high-dimensional approximation, tree tensor networks, hierarchical tensor format, adaptive algorithms, uncertainty quantification

Abstract: Many problems require the evaluation of complex parametrized models for many instances of the parameters, particularly for uncertainty quantification. When the model is costly to evaluate, it is usually approximated by another model cheaper to evaluate.

The aim of this thesis is to develop statistical learning methods using model classes of functions in tree-based tensor formats for the approximation of high-dimensional functions, both for supervised and unsupervised learning tasks. These model classes, which are rank-structured functions parametrized by a tree-structured network of low-order tensors, can be interpreted as deep neural networks with particular architecture and activation functions. The approximation is obtained by empirical risk minimization over the set of functions in tree-based tensor format.

For a high-dimensional function, or when little information on the function is available, the model class has to be carefully selected. We propose stable learning algorithms that adapt the tree and ranks and select the model based on cross-validation estimates. Furthermore, some functions might only exhibit a low-rank structure after a suitable change of variables. For such cases, we propose adaptive learning algorithms with model classes combining tree-based tensor formats and changes of variables.

The proposed algorithms are applied to uncertainty quantification in vibroacoustics.

This thesis is included in the Joint Laboratory of Marine Technology between Naval Group, Centrale Nantes and Université de Nantes, and in the Eval-PI project.