Technische Universität München
Lehrstuhl für Informatik mit Schwerpunkt Wissenschaftliches Rechnen

Sparse Grids for Big Data: Exploiting Parsimony for Large-Scale Learning

Valeriy Khakhutskyy

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des Akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzende(r):	Prof. Dr. Daniel Cremers
Prüfer der Dissertation:	1. Prof. Dr. Hans-Joachim Bungartz
	2. Prof. Dr. Markus Hegland

Die Dissertation wurde am 12.07.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 07.11.2016 angenommen.

Abstract

High-dimensional data analysis becomes ubiquitous in both science and industry. An important tool for data analysis is supervised learning with nonparametric models, which estimates the dependency between target and input variables without imposing explicit assumptions on the data. This generality, however, comes at a price of computational costs that grow exponentially with the dimensionality of the input. In general, nonparametric models cannot evade this curse of dimensionality unless the problem exhibits certain properties.

Hence, to facilitate large-scale supervised learning, this thesis focuses on two such properties: the existence of a low-dimensional manifold in the data and the discounting importance of high-order interactions between input variables. Often a problem would exhibit both these properties to a certain degree. To identify and exploit these properties, this work extends the notion of parsimony for hierarchical sparse grid models. It develops learning algorithms that simultaneously optimise the model parameters and the model structure to befit the problem at hand.

The new algorithms for adaptive sparse grids increase the range of computationally feasible supervised learning problems. They decrease the computation costs for training sparse grid models and the memory footprint of the resulting models. Hence, the algorithms can be used for classification and regression on high-dimensional data. Furthermore, they improve the interpretability of the sparse grid model and are suitable for learning the structure of the underlying data distribution.

Contents

1. Introduction

> *The idea of a learning machine may appear paradoxical to some readers. How can the rules of operation of the machine change? They should describe completely how the machine will react whatever its history might be, whatever changes it might undergo. The rules are thus quite time-invariant. This is quite true. The explanation of the paradox is that the rules which get changed in the learning process are of a rather less pretentious kind, claiming only an ephemeral validity. [...]*
> *An important feature of a learning machine is that its teacher will often be very largely ignorant of quite what is going on inside, although he may still be able to some extent to predict his pupil's behavior.*
>
> — Alan Turing, *Computing Machinery and Intelligence*

In this day and age the importance of data analysis can hardly be overrated. Between Harvard Business Review calling data scientist "the sexiest job of the 21st century" (Davenport & Patil, 2012) and the German Chancellor Angela Merkel calling data "the commodity of the 21st century" (Merkel, 2015), for somebody in the business of gathering, storing, and processing large amounts of information this century could not have started better. This dissertation presents novel methods for data mining with sparse grids to accommodate for the growing demand in large-scale data analysis.

At the dawn of computer science in the middle of the 20th century, statistics and computer science walked hand in hand. The founding father of computer science, Alan Turing, developed new statistical techniques to revolutionise cryptoanalysis in the 40s, while Sir Ronald A. Fisher wrote the first computer application for biology in the 50s. Unfortunately, soon afterwards the communities of statisticians and computer scientists diverged developing new data analysis techniques independently from each other.

In the beginning of the 21st century the rapid growth of computational power and the diminishing costs for collecting and storing information resulted in the increasing interest in large-scale data analysis by science and industry (and funding agencies). The emerging need for data analysis algorithms that can efficiently run on computational clusters on the one side, while giving theoretical performance guarantees on the other, compelled computer scientists and statisticians to unite their forces.

This rapprochement, while still ongoing, already produced some fruitful results. Sparse gird techniques is a paragon of this exchange.

Sparse grid techniques have their beginning in the finite element methods—a traditional technique for function discretisation in numerics. A function is represented as a linear combination of basis functions with compact supports located on a regular grid. The theory of the finite element methods is thoroughly studied in numerics and includes error bound estimates for different classes of functions. Unfortunately, to maintain the discretisation accuracy as the dimensionality of the input space grows, the number of basis functions has to grow exponentially with it. This curse of dimensionality drives the finite element methods to the limits of computational feasibility even for four- or five-dimensional problems.

An important step to solve this problem was the move from nodal basis functions to hierarchical ones. This gave a granular control of the frequencies of the approximant in different dimensions. It turns out that, if the function is sufficiently smooth, many high-frequency basis functions of the full grid discretisation can be removed with a negligible loss of accuracy. This observation gave rise to the *sparse grid representation* that was successfully used to delay the onset of the curse of dimensionality for numerical problems such as solving PDEs and estimating the quadrature of functions.[1]

The second important step was the transition of the sparse grid techniques from a class of finite element models in the realm of numerics into the class of nonparametric models in the realm of machine learning and data mining. This allowed one to solve high-dimensional statistical problems using rigorously studied numerical methods.

For supervised learning problems the goal is to approximate a function that generated the observed data. Provided that this function is sufficiently smooth, the use of sparse grid models delays the onset of the curse of dimensionality. If the function possesses some irregularities in a certain area or dimension of the input space then spatially-adaptive or dimension-adaptive sparse grid techniques can provide additional resolution where it is needed.

Compared to established machine learning techniques like artificial neural networks or support vector machines, sparse grids provide an approximant that is easier to analyse and to interpret. More importantly, they are based on a high-dimensional discretisation of the feature space, and thus less data-dependent than conventional approaches, scale only linearly in the number of data points and are well-suited to handle large-scale problems with vast amounts of data.

However, there is no free lunch in machine learning. As Wolpert (1996) showed in his famous "No Free Lunch" Theorem, for a general supervised learning problem the different machine learning algorithms are a-priori indistinguishable. This means that in order to successfully apply an algorithm to a high-dimensional data mining problem we are bound to exploit special properties

[1] The concept of sparse grids should not be confused with the concept of adaptive grids. These concepts are orthogonal. Similar to the full grid representation, the location of the sparse grid basis functions is problem independent. Additionally, adaptivity to the structure in a particular problem can be integrated into the sparse grid methods similar to the adaptivity in full grids.

of this problem. We consider three such properties: 1) the approximant has a high degree of smoothness, 2) the data lie on a low-dimensional manifold embedded in a high-dimensional ambient space, and 3) the approximant can be represented as a sum of lower-dimensional functions. Besides the exploitation of these properties, an algorithm for large-scale data analysis should benefit from the current trend of growing computational performance in emerging multi-core and multi-CPU computer architectures. Hence, it should be efficiently parallelisable.

Fortunately, sparse grids with regular structure have already been shown as an appropriate method for fitting functions where the *first property* is satisfied (Garcke, 2004). This dissertation begins with the introduction of sparse grid techniques in **Chapter 2** providing the notation, definitions, and theoretical foundations for later chapters. Moreover, this chapter includes a discussion of the relationship between the sparse grids and other machine learning techniques. It establishes the common vocabulary for readers coming from machine learning and numerical analysis backgrounds. We assume the Bayesian perspective on the sparse grid models and reinterpret the regularisation as a prior assumption of the parameter distribution. Besides Bayesian linear regression, we consider the use of sparse grid models as kernel machines, deriving a new efficient kernel trick algorithm based on the divide-and-conquer paradigm. Finally, we consider the sparse grid regression as a linear smoother gaining new insights into the sparse grid approximation space.

The *second property* is discussed in **Chapter 3**. Generally, the low-dimensional manifold is unknown and estimated from data. Some of the estimation methods were relying on multi-resolution analysis (Szlam et al., 2005). We follow the similar intuition and develop refinement heuristics for spatially-dimension-adaptive sparse grid methods that discover the manifold during the learning phase. We review the methods for optimal subset selection and the sparsity-inducing norms focusing on their applicability to the sparse grid models. In a series of numerical experiments we compare the different methods and show how new spatially-dimension-adaptive sparse grid methods can be used to further delay the onset of the curse of dimensionality.

The *third property* is satisfied with a rapid decrease of importance of high-order interaction terms of a functional ANOVA decomposition of the underlying function. In this case we can relate the ANOVA structure with the structure of the sparse grid model, using an established mathematical theory to estimate the importance of different sparse grid components. In **Chapter 4** we describe a number of methods that can exploit the additive structure. These methods range from an ensemble of low-dimensional sparse grid models over generalised sparse grids with arbitrary "thinness" to the ANOVA extension of the sparsity-inducing regularisation norms introduced in the previous chapter. A series of numerical experiments evaluates the efficacy of the presented methods.

Finally, the methods derived in Chapters 3 and 4 require new optimisation algorithms to train the sparse grid models. *Parallelisation* of these optimisation

algorithms requires new paradigms that exceed a simple parallel sparse grid evaluation. Therefore, **Chapter 5** focuses on the description and analysis of parallel optimisation algorithms. We derive two new methods based on the BiCGStab procedure and the Alternating Directions Method of Multipliers. We analyse the numerical properties and discuss the parallelisation of these methods using synchronous and asynchronous communication patterns.

In corporate jargon, the term *Big Data* is often used to describe datasets that exhibit one or several of the five *V*'s: variability, variety, velocity, veracity, and volume. In this dissertation we focus on the last *V*. For data analysis, the volume is equal to the number of measurements times the number of attributes or dimensions. Sparse grid methods exhibit linear complexity with respect to the number of measurements, which is already optimal. This dissertation, therefore, aims to improve the effectiveness of handling dimensionality, posing the question: How can we keep building tractable sparse grid models even as the dimensionality of the problems we consider keeps growing? And the answer I give in this dissertation is: By being economical and identifying only those parts of the model that are crucial for solving the problem.

So without further ado, let us embark into the dense forest of sparse grid methods!

2. Sparse Grids for Statistical Learning

> *PSYCHOHISTORY–...Gaal Dornick, using nonmathematical concepts, has defined psychohistory to be that branch of mathematics which deals with the reactions of human conglomerates to fixed social and economic stimuli....*
> *... Implicit in all these definitions is the assumption that the human conglomerate being dealt with is sufficiently large for valid statistical treatment. The necessary size of such a conglomerate may be determined by Seldon's First Theorem which ... A further necessary assumption is that the human conglomerate be itself unaware of psychohistoric analysis in order that its reactions be truly random ...*
> *The basis of all valid psychohistory lies in the development of the Seldon Plan. Functions which exhibit properties congruent to those of such social and economic forces as ...*
> *ENCYCLOPEDIA GALACTICA*
>
> — Isaac Asimov, *Foundation*

This chapter lays the foundations for the advanced learning algorithms discussed in the next chapters. It starts with a discussion of supervised learning from a statistical perspective. For sparse grid novices, Section 2.2 offers an introduction focused on the properties and methods we will need later. A sparse grid veteran should still skim through this section to get familiar with the notation. Section 2.3 describes the algorithm for supervised learning with adaptive sparse grids, which we will build upon in the later chapters. We conclude this chapter with a discussion of the relationship between sparse grids and other machine learning methods. Hopefully, this discussion can assist in understanding of sparse grids for people with an intimate knowledge of particular machine learning techniques and inspires new machine learning applications in the sparse grids community.

2.1. Supervised Learning

Supervised learning aims to predict new data from past observations. Let the finite dataset $S = \{(x_{i1}, \ldots, x_{iD}, y_i)\}_{i=1}^{N} \subset [0,1]^D \times \mathcal{Y}\}$ be drawn from a probability distribution with some "nature's" probability density function p. We call $\mathbf{x}_i = (x_{i1}, \ldots, x_{iD})^T$ the input variables and y_i the dependent or target variable.

In general, $\mathbf{x}_i$ may take any value in $\mathbb{R}^D$. However, for any finite dataset we can normalise the input variables to $[0,1]^D$.

We are concerned with the prediction of observable targets y using input variables $\mathbf{x}$. In the terms of frequentist statistics, we are looking for a hypothesis function f of the form $[0,1]^D \to \mathcal{Y}$ that minimises the risk

$$R(f;p) := \int L(f(\mathbf{x}),y)p(\mathbf{x},y)\,\mathrm{d}\mathbf{x}\,\mathrm{d}y \tag{2.1}$$

for a certain loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.

Certainly, the density function p is usually unknown and the common approach approximates p using the empirical distribution

$$p_{\mathrm{emp}}(\mathbf{x},y|S) := \tfrac{1}{N}\sum_{i=1}^{N} \delta_{\mathbf{x}_i}(\mathbf{x})\delta_{y_i}(y), \tag{2.2}$$

with the Kronecker delta function

$$\delta_a(b) := \begin{cases} 1 & \text{if } a = b, \\ 0 & \text{else.} \end{cases} \tag{2.3}$$

This approximation of the density function yields the approximation of the risk function (2.1) called empirical risk:

$$R_{\mathrm{emp}}(f;S) := R(f;p_{\mathrm{emp}}(\cdot|S)) = \tfrac{1}{N}\sum_{i=1}^{N} L(f(\mathbf{x}_i),y_i). \tag{2.4}$$

Now, in order to make predictions for the new input vectors, we first need to find the function from a hypothesis space V that minimises the empirical risk:

$$f_{\mathrm{ERM}} = \arg\min_{f\in V} R_{\mathrm{emp}}(f;S). \tag{2.5}$$

What happens if the empirical distribution does not approximate "nature's" distribution sufficiently well? In this case the minimisation of the empirical risk can lead to overfitting and the prediction accuracy of f_{ERM} deteriorates. To prevent overfitting we equip the objective function with a regularisation term obtaining a *regularised empirical risk functional*

$$R_\Omega(f;S) = R_{\mathrm{emp}}(f;S) + \lambda\Omega(f). \tag{2.6}$$

The functional $\Omega(f)$ measures the model complexity, while the regularisation parameter λ controls the trade-off between the approximation error and the function smoothness.

Regression

If $\mathcal{Y}$ is a set of real numbers, like in prediction of the temperature or housing prices, we speak of *regression*. In this case, we use the squared loss function $L(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$, and the empirical risk becomes the mean squared error.

Classification

If $\mathcal{Y}$ is a discrete and non-metric space, we speak of classification. For example, for $\mathcal{Y} = \{\text{“sweet”}, \text{“bitter”}, \text{“sour”}\}$ we cannot measure the distance between the individual elements. If there are only two classes, we speak of binary classification, map these classes to $\{0, 1\}$, and use the 0-1 loss $L(f(\mathbf{x}), y) = \mathbb{I}(f(\mathbf{x}) \neq y)$, with the indicator function $\mathbb{I}$. Alternatively, we map the classes to $\{-1, 1\}$ and use the hinge loss function $L(f(\mathbf{x}), y) = |1 - f(\mathbf{x}) \cdot y|_+$. Here the expression $|\cdot|_+$ denotes the positive part of its subject:

$$|x|_+ = \max\{0, x\}.$$

Yet another way is to relax the problem, assuming $\mathcal{Y} = [-1, 1]$ instead of $\{-1, 1\}$. Now we can solve a regression problem with squared loss and classify new targets using $\operatorname{sgn}(f(\mathbf{x}))$.

Minimisation of the empirical risk functional and minimisation the regularised empirical risk functional are the key ideas common for most supervised learning approaches. On several occasions in this dissertation we will come back to these ideas deriving new learning algorithms and models.

2.2. Grid-Based Prediction Models

To find a model that minimises the regularised empirical risk (2.6), we need a way to represent the hypothesis space and its elements. We use a nonparametric representation, which avoids any strong assumptions about the form of the model, representing it as a weighted sum of simple basis functions instead. Initially, we disconnect from a particular data distribution, ensuring a uniform resolution in the complete input domain. Hence the name of grid-based prediction models. Later, we extend the model construction methods to account for non-uniform distributions and irregularities.

Following a long tradition, we consider a space of piecewise d-linear functions as our hypothesis space V (Zenger, 1990). The basis functions of this space are constructed using the following principles:

1 We start by defining the mother hat function ϕ:

$$\phi : [-1, 1] \to [0, 1] \tag{2.7}$$

$$x \mapsto |1 - |x||_+ .$$

2a The one-dimensional *linear* basis functions are obtained from ϕ by scaling and shifting. The scaling coefficient 2^l is determined by a *level* $l \in \mathbb{N}$ and the corresponding shifting coefficient $i \cdot 2^{-l}$ is additionally determined by an odd *index* $i \in \{1, 3, \cdots, 2^l - 1\}$:

$$\phi_{l,i}(x) := \phi\left(\frac{x - 2^{-l}i}{2^{-l}}\right) = \phi(2^l x - i). \tag{2.8}$$

The linear basis functions for levels 1, 2, and 3 are depicted in Fig. 2.1a. We use this construction if it can be safely assumed that the model has the values 0 on the boundaries.

2b If the boundary values of the model are different from 0, we use the following construction:

$$\phi_{l,i}(x) := \begin{cases} 1 & \text{if } l = 1 \wedge i = 1, \\ \begin{cases} 2 - 2^l \cdot x & \text{if } x \in [0, 2^{1-l}] \\ 0 & \text{else} \end{cases} & \text{if } l > 1 \wedge i = 1, \\ \begin{cases} 2^l \cdot x + 1 - i & \text{if } x \in [1 - 2^{1-l}, 1] \\ 0 & \text{else} \end{cases} & \text{if } l > 1 \wedge i = 2^l - 1, \\ \phi(x \cdot 2^l - i) & \text{else.} \end{cases} \tag{2.9}$$

Pflüger (2010) called it the *modified linear* basis and so will we. The modified linear basis functions up to the level 3 are depicted in Fig. 2.1b.

3 Finally, we use the tensor-product scheme to combine the one-dimensional basis functions into D-dimensional. Let us assemble the level parameters l_t in the dimension t into the vector $\mathbf{l} = (l_1, \ldots, l_D)$ and the index parameters i_t into the vector $\mathbf{i} = (i_1, \ldots, i_D)$. Then we can write the D-dimensional basis function as

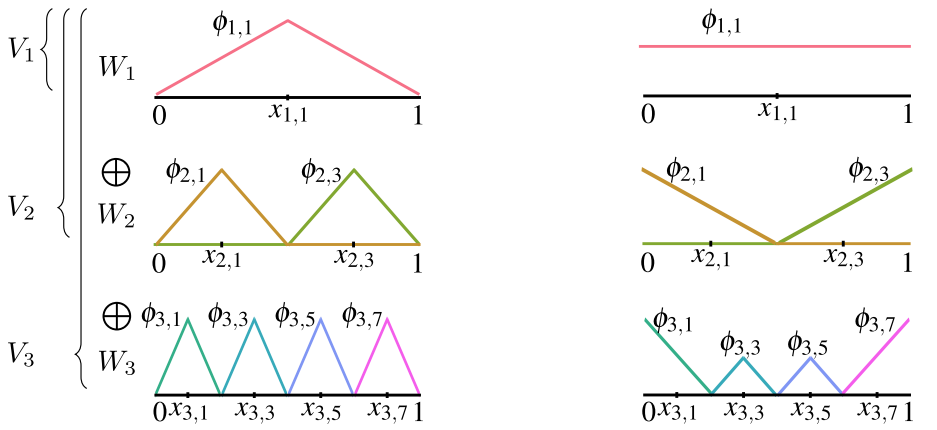
$$\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) := \prod_{t=1}^{D} \phi_{l_t,i_t}(x_t). \tag{2.10}$$

See Fig. 2.2 for illustration of two-dimensional multi-linear basis functions.

Full Grid

We combine all multi-indices associated with the level vector $\mathbf{l}$ into an index-set

$$G_{\mathbf{l}} := \{(\mathbf{l}, \mathbf{i}) \in \mathbb{N}^D \times \mathbb{N}^D \,|\, 1 \leq i_t \leq 2^{l_t} - 1,\ i_t \text{ odd},\ 1 \leq t \leq D\}. \tag{2.11}$$



(a) Linear basis (b) Modified linear basis

Figure 2.1.: Illustration of one-dimensional hierarchical basis functions up to level 3. All basis functions at the level l span a hierarchical subspace W_l. The hierarchical subspaces combined together construct the function space V_l. One-dimensional function spaces for full and sparse grids are indistinguishable.

The basis functions generated using the level and index vectors from $G_{\mathbf{l}}$ span a function space

$$W_{\mathbf{l}} := \operatorname{span}\{\phi_{\mathbf{l},\mathbf{i}} \mid (\mathbf{l},\mathbf{i}) \in G_{\mathbf{l}}\}. \tag{2.12}$$

Figure 2.2 illustrates the two-dimensional basis functions of different spaces $W_{\mathbf{l}}$. Notice how the tips of the basis functions build a regular grid, with grid points corresponding to individual multi-indices. If a basis function has an index i_t and level l_t in a particular dimension t, then the coordinate of the corresponding tip is $2^{-l_t} \cdot i_t$. Hence, we use the tip "points" to visually identify the basis functions.

The combination of the spaces $W_{\mathbf{l}}$ with levels up to ℓ forms the function space V_ℓ:

$$\begin{aligned} G_\ell &:= \bigcup_{|\mathbf{l}|_\infty \leq \ell} G_{\mathbf{l}}, \\ V_\ell &:= \bigoplus_{|\mathbf{l}|_\infty \leq \ell} W_{\mathbf{l}}. \end{aligned} \tag{2.13}$$

The function space $W_{\mathbf{l}}$ is called a *hierarchical subspace*. The space $V_{\mathbf{l}}$ is the space of D-dimensional piecewise linear functions with the mesh width $h_\ell = 2^{-\ell}$ in each dimension (see Fig. 2.3). Since the basis function "points" build a regular grid, we call this scheme (2.13) a *full grid*.

Let $\mathbb{R}^G$ be a set of real numbers, indexed by the elements of the index-set G:

$$\mathbb{R}^G := \{(w_g)_{g \in G} \mid w_g \in \mathbb{R}\}. \tag{2.14}$$

Clearly, the cardinality $|\mathbb{R}^G|$ is the same as $|\mathbb{R}^{|G|}|$ and an element of this set, $\mathbf{w} \in \mathbb{R}^G$ can be viewed as a vector from $\mathbb{R}^{|G|}$ where the components are accessed not by their sequential number, but by the corresponding multi-index.

Every $\mathbf{w}$ represents a function $u \in V_\ell$ defining a linear combinations of the basis vectors

$$u(\mathbf{x}) = \sum_{g \in G_\ell} w_g \phi_g(\mathbf{x}). \tag{2.15}$$

How useful is V_ℓ for supervised learning? Assume, that our "nature" is sufficiently smooth. Mathematically speaking, we take the differential operator

$$D^{\mathbf{l}} f := \frac{\partial^{|\mathbf{l}|_1}}{\partial x_1^{l_1} \partial x_2^{l_2} \cdots \partial x_D^{l_D}} f$$

and consider the space of functions where the weak mixed derivatives up to the order two are bounded:

$$\mathcal{H}^2_{\text{mix}} := \left\{ f : [0,1]^D \to \mathbb{R} \mid D^{\mathbf{l}} f < \infty, |\mathbf{l}|_\infty \leq 2, f|_{\delta[0,1]^D} = 0 \right\}.$$

For $f \in \mathcal{H}^2_{\text{mix}}$ the "benefit"—the interpolation error of $f_\ell \in V_\ell$—is bounded as

$$\|f - f_\ell\|_2 \leq \tfrac{D}{9^D} \cdot 2^{-2\ell} \cdot \|f\|_{\mathcal{H}^2_{\text{mix}}} \in \mathcal{O}(2^{-2\ell})$$

(see Bungartz, 1998, Lemma 3.1).

The "cost"—the number of basis functions required to represent V_ℓ—is $(2^\ell - 1)^D \in \mathcal{O}(2^{\ell D})$. Hence, to achieve the approximation accuracy of ε, we need to pay $\mathcal{O}(\varepsilon^{-D/r})$ in terms of memory and time, with $r > 0$. This holds even when the functions are sufficiently smooth. Bellman (1961) coined the term *curse of dimensionality* to describe this exponential dependency between accuracy and costs.

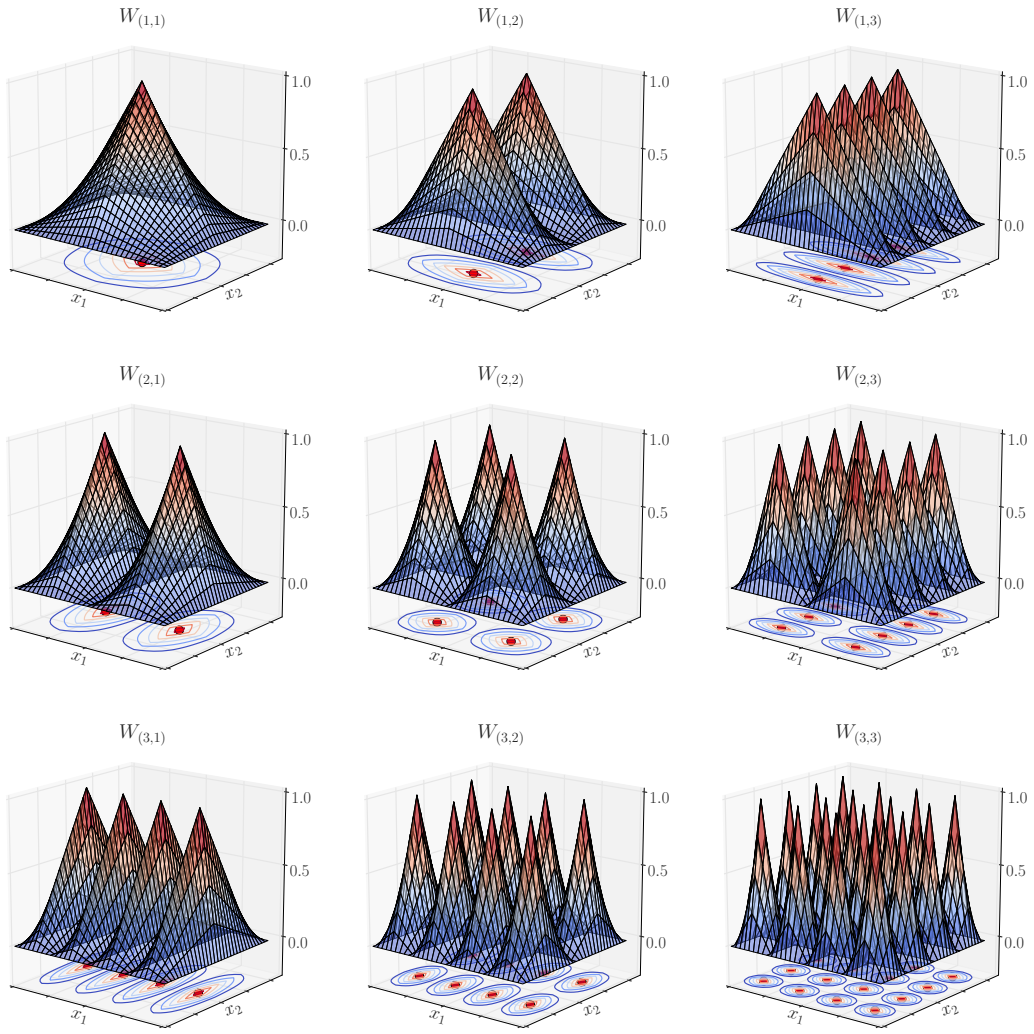


Figure 2.2.: Tableau of two-dimensional piecewise linear basis functions that build different hierarchical subspaces $W_\mathbf{l}$. Each two-dimensional "tent" function is a result of tensor product of two one-dimensional "hat" functions. The placement of the functions resembles a regular grid.

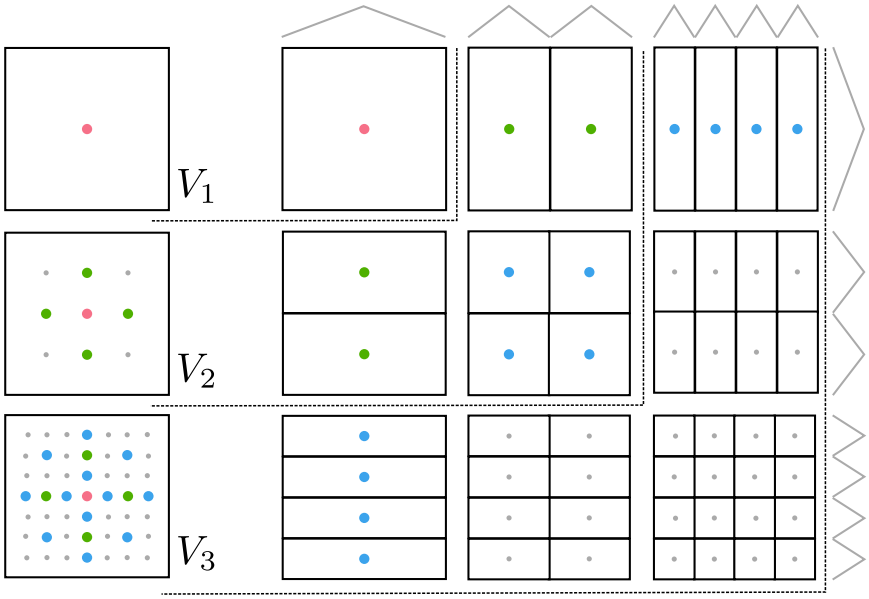


Figure 2.3.: Construction of full grid spaces from hierarchical subspaces. The scheme illustrates the construction of full grid function spaces for maximum levels 1, 2 and 3 (all points) as well as sparse grid function spaces (bold coloured points only).

Sparse Grids

Due to their high storage and computational costs, full grid models in V_ℓ can be prohibitively expensive even for problems with moderate dimensionality of, say, four or five. However, we can significantly reduce these costs to the price of a slightly lower accuracy. Given a fixed budget of M basis functions, we are looking for the index-set G_ℓ^1 such that

$$G_\ell^1 := \operatorname*{arg\,min}_{\substack{G \subset G_\ell \\ |G|=M}} \max_{f \in \mathcal{H}^2_{\text{mix}}} \min_{f_G \in V_G} \|f - f_G\|_2. \tag{2.16}$$

We call (2.16) an *a priori* optimisation problem, as it identifies the optimal index-set for an approximation of a whole problem class and not for a particular learning problem and a given dataset. This a priori optimisation problem, although seemingly complicated, can be solved analytically. If the cardinality constraint M is selected accordingly, the solution is

$$G_\ell^1 = \bigcup_{|\mathbf{l}|_1 \leq \ell + D - 1} G_{\mathbf{l}}, \quad V_\ell^1 = \operatorname{span}\{\phi_g | g \in G_\ell^1\} \tag{2.17}$$

and

$$\left|G_\ell^1\right| = M = \sum_{i=0}^{\ell-1} 2^i \cdot \binom{D-1+i}{D-1} \in \mathcal{O}(2^\ell \ell^{D-1}) \tag{2.18}$$

(see Bungartz, 1998, Lemma 3.2).

Notice, this is a dramatic reduction from $\mathcal{O}(2^{\ell D})$ for the full grids! The interpolation error now is bounded from above by an element from $\mathcal{O}(2^{-2\ell} \cdot \ell^{D-1})$—somewhat worse than $\mathcal{O}(2^{-2\ell})$ previously. We refer the reader to the paper by Bungartz and Griebel (2004) for the derivation of these results. In contrast to the full grid scheme (2.13), we call (2.17) a *sparse grid* scheme. In Fig. 2.3 the sparse grid "points" are larger and depicted in colour.

The idea of sparse grids was originally discussed by Smolyak (1963) and then rediscovered by Zenger (1990) years later. Sparse grids have been successfully used for different data mining and machine learning applications including regression, classification, clustering, and density estimation (e.g., O. M. Nielsen, 2000; Garcke et al., 2001, 2003; O. Nielsen et al., 2004; Garcke, 2004; Bungartz et al., 2008; Pflüger, 2010; Peherstorfer et al., 2011; Peherstorfer, 2013).

In practice, the use of a sparse grid approximation instead of a full grid allows a substantial reduction in time complexity with just slightly lower accuracy, consequently delaying the onset of the curse of dimensionality. As the result, Garcke (2004) and Pflüger (2010) could approach high-dimensional data mining problems including the 18-dimensional Data-Mining-Cup, the 22-dimensional mushroom classification, the 64-dimensional optical digits recognition, and the 166-dimensional Musk-1 benchmark.

Adaptivity

The sparse grid construction scheme is optimal for a general class of problems. However, it can be further improved for many given concrete problems, for example, if the function we want to approximate is not sufficiently smooth in some areas of the domain. In such cases we can further optimise the sparse grid scheme taking into account the specifics of the prediction problem at hand. The a priori optimisation problem (2.16) becomes an *a posteriori* optimisation problem

$$\widetilde{G}_\ell = \operatorname*{arg\,min}_{\substack{G \subset G_\ell \\ |G|=M}} \min_{f \in V_G} R_\Omega(f; S). \tag{2.19}$$

In practice, finding $\widetilde{G}_\ell$ would require solving a combinatorial optimisation problem, which is computationally infeasible. Hence, a forward greedy algorithm—a sparse grid refinement procedure—is employed instead. This procedure identifies a sequence of incrementally growing sparse grid index-sets $G^{(0)} \subset G^{(1)} \subset \cdots$ where each increment maximises the *marginal gain*

$$G^{(k+1)} := \operatorname*{arg\,max}_{\substack{G \supset G^{(k)} \\ |G|-|G^{(k)}|=m}} \left(\min_{f \in V_{G^{(k)}}} R_\Omega(f; S) - \min_{g \in V_G} R_\Omega(g; S) \right) \tag{2.20}$$

from adding m new indices to the set.

We keep the rigorous discussion of Problem (2.19) for the next chapter and focus rather on an intuitive explanation of sparse grids *adaptivity* here. Since basis functions have local support, by adding new basis functions we are able to refine the model in the areas with high data density or some function irregularities while keeping the rest of the model unchanged. This leads to higher resolution in the areas where it is really needed.

A formal definition of refinement requires the notion of hierarchical ancestry and descendancy between indices. Let Ω be a set of all possible indices

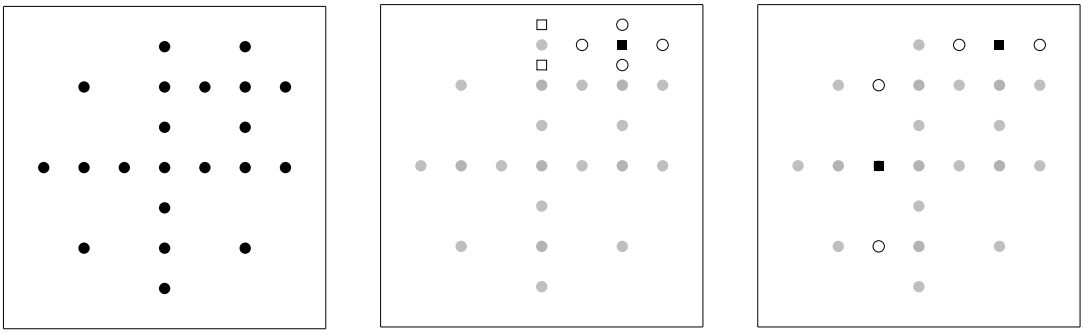
$$\Omega := \{(\mathbf{l}, \mathbf{i}) \mid (\mathbf{l}, \mathbf{i}) \in \mathbb{N}^D \times \mathbb{N}^D,\ i_t \text{ odd for each } 1 \leq t \leq D\}$$

and let `desc` be the function that computes the hierarchical descendants of a level-index pair $(\mathbf{l}, \mathbf{i})$ in the dimension t:

$$\begin{aligned} \texttt{desc} : \mathbb{N}^D \times \mathbb{N}^D \times \{1, \ldots, D\} &\to \mathcal{P}(\Omega) \\ \mathbf{l}, \mathbf{i}, t &\mapsto \{(\mathbf{l} + \mathbf{e}_t, \mathbf{i} + (i_t + r)\mathbf{e}_t) \mid r \in \{-1, +1\}\}. \end{aligned} \tag{2.21}$$

Here, $\mathbf{e}_t$ is a vector containing 1 in the t-th component and 0 everywhere else and r signifies if the left or the right hierarchical descendant is created. The reverse function `anc` computes all hierarchical ancestors of a level-index pair $(\mathbf{l}, \mathbf{i})$:

$$\texttt{anc} : \mathbb{N}^D \times \mathbb{N}^D \to \mathcal{P}(\Omega) \tag{2.22}$$



(a) Initial sparse grid

(b) One grid point refinement in all dimensions

(c) Two grid points refined in one dimension each

Figure 2.4.: Illustration of "spatially-adaptive" and "spatially-dimension-adaptive" refinement schemes. The sparse grid points with indices to be refined are depicted as black squares, the new children points are white circles, the new ancestor grid points are white squares.

$$\mathbf{l}, \mathbf{i} \mapsto \{(\mathbf{l} - \mathbf{e}_t, \lfloor \mathbf{i} - \tfrac{i_t - ((i_t+1) \bmod 4)}{2} \mathbf{e}_t \rfloor) \mid 1 \leq t \leq D\}.$$

Let $G^{(k)}$ denote the sparse grid index-set after the k'th refinement. The set $G^{(k)}$ is extended from the *admissible set* of candidates

$$A_{G^{(k)}} := \left\{(\mathbf{l}, \mathbf{i}) \mid \mathtt{anc}(\mathbf{l}, \mathbf{i}) \cap G^{(k)} \neq \emptyset\right\}. \tag{2.23}$$

We can either refine a point in a particular dimension:

$$G^{(k+1)} = G^{(k)} \cup \left(\mathtt{desc}(\mathbf{l}, \mathbf{i}, t) \cap A_{G^{(k)}}\right), \tag{2.24}$$

or in all dimensions:

$$G^{(k+1)} = G^{(k)} \cup \left(\mathtt{alldesc}(\mathbf{l}, \mathbf{i}) \cap A_{G^{(k)}}\right) \tag{2.25}$$

with

$$\mathtt{alldesc}(\mathbf{l}, \mathbf{i}) := \bigcup_{t=1}^{D} \mathtt{desc}(\mathbf{l}, \mathbf{i}, t). \tag{2.26}$$

In the second case we speak of *spatially-adaptive* (Fig. 2.4b) and in the first case of *spatially-dimension-adaptive* refinement (Fig. 2.4c).

Alternatively, the *dimension-adaptive* refinement strategy takes all refinable indices and creates all children indices along one direction t (Hegland, 2003; Gerstner & Griebel, 2003):

$$G^{(k+1)} = G^{(k)} \cup \bigcup_{(\mathbf{l},\mathbf{i}) \in G_{\mathbf{l}}} \mathtt{desc}(\mathbf{l}, \mathbf{i}, t). \tag{2.27}$$

The hierarchical relationship between the grid indices in a sparse grid can be specified in a directed acyclic graph (DAG). We illustrate such DAG in Fig. 2.5.

Definition 2.1 (Sparse grid graph)**.** *A sparse grid index set G induces a directed acyclic (weakly) connected graph, an ordered pair $\mathcal{G} = (G, E)$ with the set of edges*

$$E = \{(u, v) \mid u, v \in G, u \in \texttt{anc}(v)\}.$$

This graph contains only one node without any incoming edges, which is called the ***root*** *node. Every connected subgraph of $\mathcal{G}$ that contains the root node also corresponds to a sparse grid. The graph $\mathcal{G}$ is said to obey* ***strong hierarchy*** *if for every node g in G all ancestors of g are also in G. Otherwise, $\mathcal{G}$ is said to obey* ***weak hierarchy****.*

This definition allows for configurations of sparse grids invalid in other literature. The so-called *up-down* algorithms for efficient operations on a sparse grid (Balder, 1994; Balder & Zenger, 1996) require every non-root node to be accessible from all directions. However, since supervised learning does not rely on the up-down algorithms, we do not make strong hierarchy a necessary condition.

Adaptive sparse grids offer flexible nonparametric models for supervised learning problems. Because of adaptivity, these models do not require any specific assumptions about the problem at hand besides the choice of the suitable basis functions family. The hierarchical relationship between the basis functions can be a blessing or a curse depending on the situation. When designing a new algorithm for sparse grids, these dependencies should be taken into careful consideration.

2.3. Training Sparse Grid Models

Speaking of "training a sparse grid model", we mean the identification of the combination coefficients $\mathbf{w} \in \mathbb{R}^G$ and the evaluation of the model as

$$f(\mathbf{x}) = \sum_{g \in G} w_g \phi_g(\mathbf{x}).$$

Figure 2.6 illustrates how a set of hierarchical piecewise linear functions from Fig. 2.1a is used to fit the data points. Figuratively speaking, the basis functions on higher levels compensate for the prediction errors on lower levels. That is why some of the function on level 3 are negative, even though the optimal prediction is positive everywhere (Fig. 2.6a). Added up, these functions recreate the correct result (Fig. 2.6b).

Let us discuss how the combination of sparse grid models and the regularised empirical risk minimisation leads to a learning algorithm. Altogether, regression with adaptive sparse grids is performed in a succession of fitting and refinement

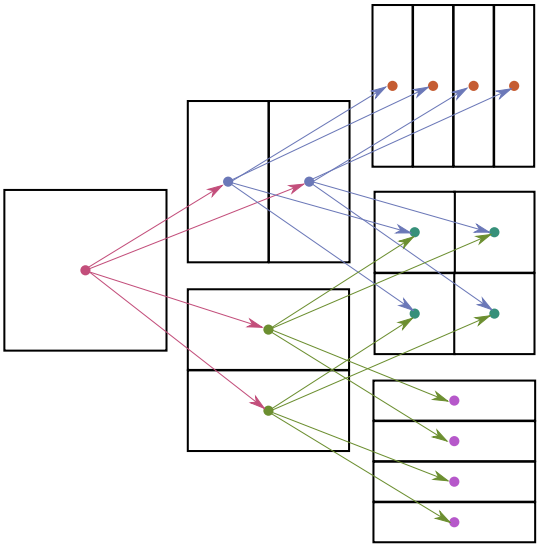
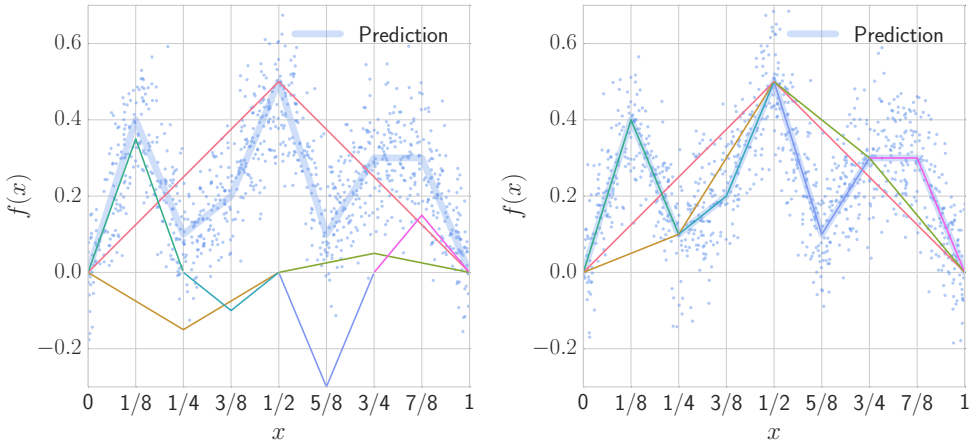


Figure 2.5.: Hierarchical dependencies between sparse grid indices depicted as a directed acyclic graph. Graph nodes signify the sparse grid indices, an edge denotes the existence of an ancestor-descendant relationship between the indices.



(a) Basis functions plotted next to each other

(b) Basis functions stacked on top of each other

Figure 2.6.: Example of regression with a hierarchical basis. The basis functions on higher levels compensate for the prediction errors on lower levels. That is why some of the function on level 3 are negative, even though the optimal prediction is positive everywhere.

steps as illustrated in Alg. 1. These steps alternate until some global convergence criterion is satisfied, for example, until the generalisation error on a validation dataset is sufficiently small or until the computational limit is reached. A practical implementation of these steps is often a compromise between mathematical correctness and pragmatism.

Algorithm 1: Regression with Adaptive Sparse Grids.

start with some initial $G^{(0)}$, $k \leftarrow 0$
for $k = 0, 1, \ldots$ **do**
  Fit-Step: Compute $f \in V_{G^{(k)}}$ that minimises the regularised empirical risk function (2.6)
  if *global convergence criterion satisfied* **then**
    break
  Refine-Step: Refine the sparse grid structure to $G^{(k+1)}$ that maximises the marginal gain (2.20)

Fit-Step

To use the functional (2.6), we need to specify two components: the loss function and the regularisation term. Hereafter, we need to figure out, how to find a sparse grid model that minimises the risk function. Regression requires the

squared loss for the cost function. As the result of the sparse grid discretisation, the regularisation operator $\Omega(f)$ becomes $\|\mathbf{D}\mathbf{w}\|_2^2$ for a suitable choice of $\mathbf{D}$.

Let $\mathcal{I} : \{1, \ldots, |G|\} \to G$ be a bijective mapping that enumerates the elements of G. Garcke et al. (2001) suggested a differential operator that takes the form of the discrete Laplacian

$$(\mathbf{D})_{jk} := (\nabla\phi_{\mathcal{I}(j)}, \nabla\phi_{\mathcal{I}(k)})_{L_2}, \tag{2.28}$$

while Pflüger (2012) argued that a simple identity matrix

$$(\mathbf{D})_{jk} := \delta_j(k) \tag{2.29}$$

leads to similar results. The identity matrix is inexpensive, works well in practice and has a probabilistic interpretation as we will see in Sec. 2.4.2. Therefore, we will primarily use this regularisation operator throughout this thesis.

Since every function $f \in V_G$ is uniquely identified via its coefficients $\mathbf{w} \in \mathbb{R}^G$, the regularised expected risk functional (2.6) for finite-dimensional function spaces can be rewritten in terms of $\mathbf{w}$ and G:

$$J(\mathbf{w}; G) := \sum_{i=1}^{N} \left(\sum_{g \in G} w_g \phi_g(\mathbf{x}) - y_i \right)^2 + \lambda N \sum_{g \in G} w_g^2. \tag{2.30}$$

Here we also multiplied the expression by N, which simplifies the notation and does not change the minimiser of J.

Hence, in the fit-Step of Alg. 1 we are looking for the minimiser

$$\mathbf{w}^\star := \operatorname*{arg\,min}_{\mathbf{w} \in \mathbb{R}^G} J(\mathbf{w}; G). \tag{2.31}$$

Note that if we denote by J_{point} the cost at an individual data point

$$J_{\text{point}}(\mathbf{w}, \mathbf{x}, y; G) := \left(\sum_{g \in G} w_g \varphi_g(\mathbf{x}) - y \right)^2 + \lambda \sum_{g \in G} w_g^2, \tag{2.32}$$

we can further split $J(\mathbf{w}; G)$ into the sum of $J_{\text{point}}(\mathbf{w}, \mathbf{x}, y; G)$:

$$\begin{aligned} J(\mathbf{w}; G) &= \sum_{i=1}^{N} \left(\sum_{g \in G} w_g \varphi_g(\mathbf{x}_i) - y_i \right)^2 + \lambda N \sum_{g \in G} w_g^2 \\ &= \sum_{i=1}^{N} \left(\left(\sum_{g \in G} w_g \varphi_g(\mathbf{x}_i) - y_i \right)^2 + \lambda \sum_{g \in G} w_g^2 \right) \\ &= \sum_{i=1}^{N} J_{\text{point}}(\mathbf{w}, \mathbf{x}_i, y; G). \end{aligned}$$

One way to minimise the cost function (2.30) with respect to $\mathbf{w}$ is by solving the corresponding normal equation

$$\left(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda N \cdot \mathbf{I}\right)\mathbf{w} = \mathbf{\Phi}^T\mathbf{y}, \tag{2.33}$$

where $\mathbf{\Phi}$ is a rectangular $N \times M$ matrix with entries $(\mathbf{\Phi})_{i,j} = \phi_{\mathcal{I}(j)}(\mathbf{x}_i)$ and $\mathbf{I}$ is an identity matrix. Conjugate gradients (CG) is often used to deal with this system of equations (Pflüger, 2012).

The more general approach follows the gradient descent:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \gamma^t \nabla_{\mathbf{w}} J(\mathbf{w}^t; G). \tag{2.34}$$

Depending on the particular choice of γ^t, the update step (2.34) leads to a number of different methods, such as Newton-Raphson, BFGS, etc. Borrowing a term from neural network literature, we call this class of algorithms the optimisation in *batch mode*, since the information from the whole "batch" S is taken into account at every update step (Bottou, 1998).

The complexity of these descent methods is dominated by function evaluations. And since the intermediate results are rarely stored explicitly, every gradient descent iteration would have the time complexity of $\mathcal{O}(NM)$. For k update steps the algorithms also perform k passes through the data points. This results in the time complexity of $\mathcal{O}(k \cdot NM)$.

For some problems the batch gradient descent method is infeasible, for example, when the dataset is to large to fit into the main memory at once. For some problems it is impractical, for example, when the training patterns arrive continually in a data stream. To overcome these limitations, one can consider a stochastic approximation of the gradient term in (2.34) by the gradient at one (random) point (Bottou, 1998), which yields an update of the form

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \gamma^t \nabla_{\mathbf{w}} J_{\text{point}}(\mathbf{w}^t, \mathbf{x}_t, y_t; G). \tag{2.35}$$

In practice, additional measures are taken to obtain robust results and faster convergence (Bach & Moulines, 2013; Polyak & Juditsky, 1992; Schaul, Zhang, & LeCun, 2013).

Refine-Step

Let $\widetilde{J}(G)$ be a set function over the sparse grid index-set G that estimates optimal parameters $\mathbf{w}$:

$$\widetilde{J}(G) := \min_{\mathbf{w} \in \mathbb{R}^G} J(\mathbf{w}; G). \tag{2.36}$$

The refinement procedure aims to find the best multi-indices to maximise the marginal gain among all candidates. In general, marginal gain is defined as follows.

Definition 2.2. *Let Γ be a set, $\mathcal{P}(\Gamma)$ its powerset, and $h : \mathcal{P}(\Gamma) \to \mathbb{R}$ a set function. The* ***marginal gain*** *with respect to the function h is defined as*

$$h(x|X) := h(X \cup \{x\}) - h(X), \tag{2.37}$$

where $x \in \Gamma$ is an element of the set and $X \subset \Gamma$ its subset.

Hence, putting it formally, the refinement identifies the best m multi-indices $I^\star$ among all candidates A_G that maximises the marginal gain associated with $\widetilde{J}$:

$$I^\star = \underset{\substack{I \subset A_G \\ |I|=m}}{\arg\max} -\widetilde{J}(I|G). \tag{2.38}$$

This view was first suggested by Hegland (2003) and Garcke (2004) in the context of *generalised* sparse grids and the dimension-adaptive refinement. They considered subsets of multi-indices that belong to the same hierarchical subspaces. As the result, the authors derived a greedy method, which yields optimality bounds under certain conditions as discussed in Chapter 3. However, their method bears large computational costs at every step and becomes infeasible for large problems. Furthermore, the need to include complete hierarchical subspaces limits the desired ability to refine only some parts of the domain important for a regression problem at hand.

Pflüger (2010) suggested a heuristic indicator for marginal gain and local adaptivity in order to overcome these limitations. Assuming that the errors are normally distributed around 0, one can improve marginal return by adding new multi-indices that correspond to the area with high local error variation. This variation is weighted by potential influence of the basis functions. To capture this influence, Pflüger used the absolute value of the basis functions evaluated at the data points. Altogether, he suggested to refine the indices that have the highest reduction indicator:

$$\begin{aligned} s^* &:= \arg\max\{\xi(s) \mid s \in G^{(k)}, \texttt{alldesc}(s) \cap A_{G^{(k)}} \neq \emptyset\}, \\ G^{(k+1)} &:= G^{(k)} \ \cup \ \texttt{alldesc}(s^*) \ \cup \ \texttt{anc}(s^*) \end{aligned} \tag{2.39}$$

with

$$\xi(s) := \sum_{i=1}^{N} \left(y_i - f(\mathbf{x}_i; \mathbf{w})\right)^2 \cdot |w_s| \, \phi_s(\mathbf{x}_i). \tag{2.40}$$

As there is no way of telling how the new indices would contribute to the results, the refinement occurs in all directions. Adding all ancestor $\texttt{anc}(s^*)$ ensures the preservation of strong hierarchy after the refinement. The computation of this indicator is inexpensive even for large data sets. However, even though the intuition behind the indicator is clear, its optimality is difficult to analyse. It

also fails to indicate the importance of the individual dimensions that will be refined.

A training procedure for sparse grid models always alternates between two step: fitting and refining. Particular choice for these two steps lead to different algorithms and properties of the training procedure. A good fitting procedure can accelerate the training, but, more importantly, a good refining criterion is crucial for a sparse grid model to achieve a desired result. In Chapter 3 we come back to this idea, deriving a refinement method that combines efficiency, flexibility, and optimality guarantees.

2.4. Sparse Grid and other Machine Learning Techniques

We just learned one way to find sparse grid models that solve supervised learning problems. However, the supervised learning with sparse grids relates to a web of other models and representations. Understanding those, yields to a discovery of new properties and algorithms that can extend the sparse grid methods.

The discussion of the relationship to other machine learning methods helps the researchers and practitioners well acquainted with other, more customary methods, to grasp the inherited characteristics and behaviour of sparse grid models.

2.4.1. Feature Transformation

A simple way to view sparse grid discretisation is through the notion of feature maps. A feature map $\boldsymbol{\phi}$ transforms the elements of the original input space $[0,1]^D$ into the feature space $[0,1]^M$

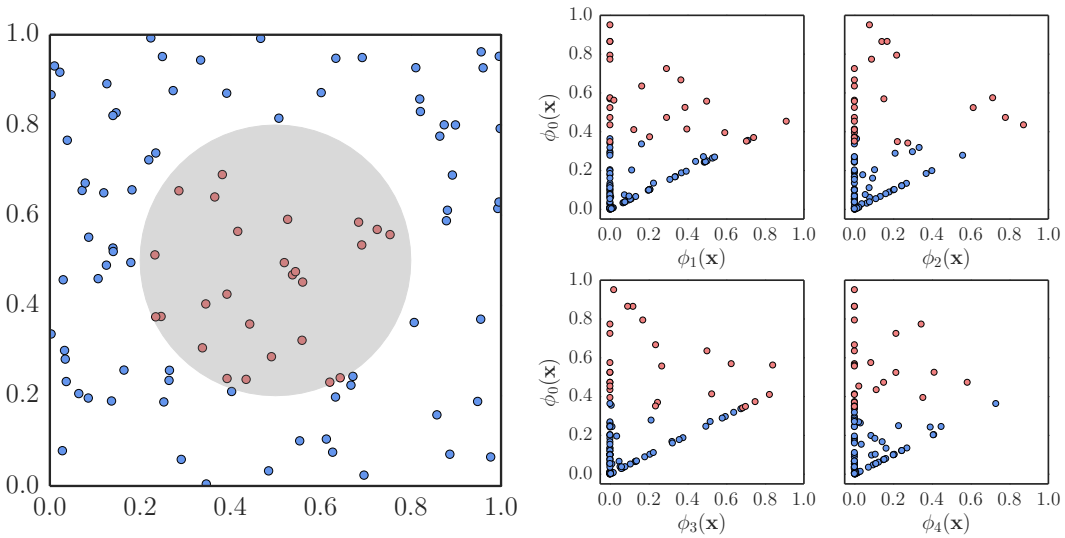
$$\begin{aligned} \boldsymbol{\phi} : [0,1]^D &\rightarrow [0,1]^M, \\ \mathbf{x} &\mapsto (\phi_1(\mathbf{x}), \ldots, \phi_M(\mathbf{x}))^T \end{aligned} \tag{2.41}$$

where usually $M \gg D$.

Feature maps or feature transformations are a popular machine learning technique. Some problems, i.e. clustering, benefit from a new similarity measure in the measurable feature space. Other problems, i.e. classification, can be separable in higher-dimensional feature spaces and non-separable in the original low-dimensional input space.

Consider the following classification problem: We have data points sampled uniformly on $[0,1]^2$ as show in Fig. 2.7a. The points inside the circle are of one class (red), the points outside are of other (blue). There is no hyperplane that would be able to separate the points of one class from another. If, however, we transform the points into a 5-dimensional feature space of a level 2 sparse grid, the problem becomes easily separable. Figure 2.7b shows two-dimensional projections of the first basis function (level 1) with other four (level 2). We can easily imagine a separating hyperplane in each of the projections.

In fact, if we forget for a moment the complexity of sparse grid space discretisation, we can consider many sparse grid learning algorithms as linear models in the corresponding feature space! With refinement we can expand the feature space on the fly to fit our needs, which gives us the property only a few other adaptive basis function methods have. So let us now discuss the implications of looking on the machine learning methods through the sparse grid feature map spectacles.



(a) Original dataset

(b) Sparse grid features representation

Figure 2.7.: Sparse grid basis function evaluation as feature transformation. Left: red and blue points belong to different classes and cannot be separated by a hyperplane. Right: the same points transformed into a 5-dimensional feature space of a level 2 sparse grid, the problem is separable.

2.4.2. Bayesian Linear Regression

In this section we are going to show that the sparse grid approximation via penalised least squares leads to the MAP estimator. This embeds the sparse grids approximation into the more general framework of Bayesian linear regressors, which uses a probabilistic interpretation to describe the underlying model.

We start, as before, with a dataset composed of inputs $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and corresponding target values $\mathbf{y} = (y_1, \ldots, y_N)^T$. We assume that targets were generated from inputs using a function $f(\mathbf{x}; \mathbf{w})$ and distorted by some additive Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. Such noise can come, for example, from calibrated measurement instruments, which do no show absolutely exact results, but are not biased in a particular direction and the amplitude of the measurement does not depend on the target value. A counter-example would be the information about the persons height, age, or income on a dating website, which tend to be systematically over- or underestimated.

Stochasticity of the noise makes the measurement to a random variable:

$$y = f(\mathbf{x}; \mathbf{w}) + \varepsilon, \tag{2.42}$$

$$p(y|\mathbf{x}; \mathbf{w}, \sigma_\varepsilon^2) = \mathcal{N}(y|f(\mathbf{x}; \mathbf{w}), \sigma_\varepsilon^2). \tag{2.43}$$

It is normally distributed with the same standard deviation as the noise and the expected value of $f(\mathbf{x};\mathbf{w})$, since the noise is expectedly 0.

As we assume that the dataset samples are drawn iid, the likelihood of the observations $\mathbf{y}$ for the inputs $\mathbf{X}$ becomes

$$p(\mathbf{y}|\mathbf{X},\mathbf{w},\sigma_\varepsilon^2) = \prod_{i=1}^{N} \mathcal{N}(y_n|\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i),\sigma_\varepsilon^2) = \mathcal{N}(\boldsymbol{\Phi}\mathbf{w},\sigma_\varepsilon^2\mathbf{I}), \tag{2.44}$$

where $\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)$ stands for the evaluation of the sparse grid approximant $f(\mathbf{x}_i)$.

We follow (Bishop, 2006) and choose a conjugate prior, also a Gaussian distribution, of the form

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_0,\mathbf{S}_0) \tag{2.45}$$

with the mean vector $\mathbf{w}_0$ and covariance matrix $\mathbf{S}_0$. And so the corresponding posterior distribution assumes the form

$$p(\mathbf{w}|\mathbf{y}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_N,\mathbf{S}_N) \tag{2.46}$$

where

$$\mathbf{S}_N = \left(\mathbf{S}_0^{-1} + \sigma_\varepsilon^{-2}\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1} \tag{2.47}$$

$$\mathbf{w}_N = \mathbf{S}_N(\mathbf{S}_0^{-1}\mathbf{w}_0 + \sigma_\varepsilon^{-2}\boldsymbol{\Phi}^T\mathbf{y}). \tag{2.48}$$

The mode and the mean of the posterior distribution coincide, and hence the maximum posterior weight vector $\mathbf{w}_{\mathrm{MAP}}$ is the same as $\mathbf{w}_N$.

In the context of online learning, where data arrive sequentially, the posterior at any time acts as the prior for the subsequently arriving data point and hence the posterior distribution has again the form (2.46).

The marginal distribution of the dataset is given by

$$p(\mathbf{y}|\mathbf{X},\sigma_\varepsilon^2,\mathbf{S}_0) = \int p(\mathbf{y}|\mathbf{X},\mathbf{w},\sigma_\varepsilon^2)\cdot p(\mathbf{w}|\mathbf{S}_0)\mathrm{d}\mathbf{w}, \tag{2.49}$$

where the first term in the integral is defined in (2.44) and the second in (2.45). Using the result (A.7) from the appendix, we can analytically compute the marginal distribution as

$$p(\mathbf{y}|\mathbf{X},\sigma_\varepsilon^2,\mathbf{S}_0) = \mathcal{N}(\mathbf{y}|\boldsymbol{\Phi}\cdot\mathbf{0},\sigma_\varepsilon^2\mathbf{I} + \boldsymbol{\Phi}\mathbf{S}_0\boldsymbol{\Phi}^T) \tag{2.50}$$

$$= \mathcal{N}(\mathbf{y}|\mathbf{0},\boldsymbol{\Sigma}) \tag{2.51}$$

$$= (2\pi)^{-\frac{N}{2}}\cdot|\boldsymbol{\Sigma}|^{-\frac{1}{2}}\exp\left(-\tfrac{1}{2}\mathbf{y}^T\boldsymbol{\Sigma}^{-1}\mathbf{y}\right) \tag{2.52}$$

with $\boldsymbol{\Sigma}$ defined as $\sigma_\varepsilon^2\mathbf{I} + \boldsymbol{\Phi}\mathbf{S}_0\boldsymbol{\Phi}^T$.

The regularisation operator (2.29) corresponds to a particular choice of Gaussian prior—a zero-mean isotropic Gaussian governed by a single parameter $\sigma_{\mathbf{w}}^2$:

$$p(\mathbf{w}|\sigma_{\mathbf{w}}^2) = \mathcal{N}(\mathbf{w}|\mathbf{0},\sigma_{\mathbf{w}}^2\mathbf{I}). \tag{2.53}$$

The corresponding posterior distribution over $\mathbf{w}$ is then given by (2.46) with

$$\mathbf{S}_N = \left(\sigma_{\mathbf{w}}^{-2}\mathbf{I} + \sigma_{\varepsilon}^{-2}\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1} \tag{2.54}$$

$$\mathbf{w}_N = \sigma_{\varepsilon}^{-2}\mathbf{S}_N\mathbf{\Phi}^T\mathbf{y}. \tag{2.55}$$

The logarithm of the posterior distribution is the sum of the log-likelihood and the logarithm of the prior and, as the function of $\mathbf{w}$, assumes the form

$$\begin{aligned}\ln p(\mathbf{w}|\mathbf{y}) &= -\tfrac{1}{2}\sigma_{\varepsilon}^{-2}\sum_{i=1}^{N}\left(y_i - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_N)\right)^2 - \tfrac{1}{2}\sigma_{\mathbf{w}}^{-2}\mathbf{w}^T\mathbf{w} + \text{const} \\ &\sim -\sum_{i=1}^{N}\left(y_i - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_N)\right)^2 - \frac{\sigma_{\varepsilon}^2}{\sigma_{\mathbf{w}}^2}\mathbf{w}^T\mathbf{w}. \end{aligned} \tag{2.56}$$

We have already seen this form in (2.30). Maximisation of this posterior distribution with respect to $\mathbf{w}$ is, hence, equivalent to the minimisation of the sum-of-squares loss function with a quadratic regularisation term and $\lambda = \sigma_{\varepsilon}^2/\sigma_{\mathbf{w}}^2$.

If we chose an infinitely broad prior, i.e. $\sigma_{\mathbf{w}} \to \infty$, the posterior distribution becomes the likelihood distribution and as $\lambda \to 0$ the impact of the regularisation diminishes.

Estimation of the optimal (continuous) hyperparameters σ_{ε}^2 and $\sigma_{\mathbf{w}}^2$ as well as of the optimal (discrete) sparse grid size and structure is a subject of the model selection problem. We discuss the choice of the hyperparameters now and return to the search of the optimal grid structure in the following chapters.

A tractable form of the hyperparameter optimisation maximises the type-2 likelihood function (2.52). We assemble the hyperparameters in the vector $\boldsymbol{\theta}$ and so the marginal log-likelihood has the form

$$\ln p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\tfrac{1}{2}\mathbf{y}^T\mathbf{\Sigma}^{-1}\mathbf{y} - \tfrac{1}{2}\ln|\mathbf{\Sigma}| - \tfrac{N}{2}\log 2\pi. \tag{2.57}$$

The gradient of the log-likelihood is required for many minimisation algorithms can be computed as (Bishop, 2006)

$$\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \tfrac{1}{2}\left(\mathbf{y}^T\mathbf{\Sigma}^{-1}\frac{\partial\mathbf{\Sigma}}{\partial\theta_j}\mathbf{\Sigma}^{-1}\mathbf{y} - \operatorname{tr}\left[\mathbf{\Sigma}^{-1}\frac{\partial\mathbf{\Sigma}}{\partial\theta_j}\right]\right) \tag{2.58}$$

$$= \tfrac{1}{2}\operatorname{tr}\left[(\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \mathbf{\Sigma}^{-1})\frac{\partial\mathbf{\Sigma}}{\partial\theta_j}\right] \quad \text{with} \quad \boldsymbol{\alpha} = \mathbf{\Sigma}^{-1}\mathbf{y}, \tag{2.59}$$

$$\frac{\partial\mathbf{\Sigma}}{\partial\sigma_{\varepsilon}^2} = \mathbf{I}, \tag{2.60}$$

$$\frac{\partial\mathbf{\Sigma}}{\partial\sigma_{\mathbf{w}}^2} = \mathbf{\Phi}\mathbf{\Phi}^T. \tag{2.61}$$

The prediction of new targets may be done using $\mathbf{w}_N$. However, we are often interested not only in the point estimation but in the complete distribution. In

this case, again using (A.7) from Appendix A, the prediction of a new target value y for new input values $\mathbf{x}$ is obtained by evaluating the predictive distribution (Bishop, 2006)

$$p(y|\mathbf{x}, \mathbf{y}, \mathbf{X}, \sigma_{\mathbf{w}}^2, \sigma_{\varepsilon}^2) = \int p(y|\mathbf{w}, \mathbf{x}, \sigma_{\varepsilon}^2) p(\mathbf{w}|\mathbf{y}, \mathbf{X}, \sigma_{\mathbf{w}}^2, \sigma_{\varepsilon}^2) \mathrm{d}\mathbf{w} \tag{2.62}$$

$$= \mathcal{N}(y|\mathbf{w}_N^T \boldsymbol{\phi}(\mathbf{x}), \sigma_N^2(\mathbf{x})), \tag{2.63}$$

where the variance is given by

$$\sigma_N^2(\mathbf{x}) = \sigma_{\varepsilon}^2 + \underbrace{\boldsymbol{\phi}(\mathbf{x})^T \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x})}_{=:\sigma_{\mathrm{mod}}^2} = \sigma_{\varepsilon}^2 + \sigma_{\mathrm{mod}}^2. \tag{2.64}$$

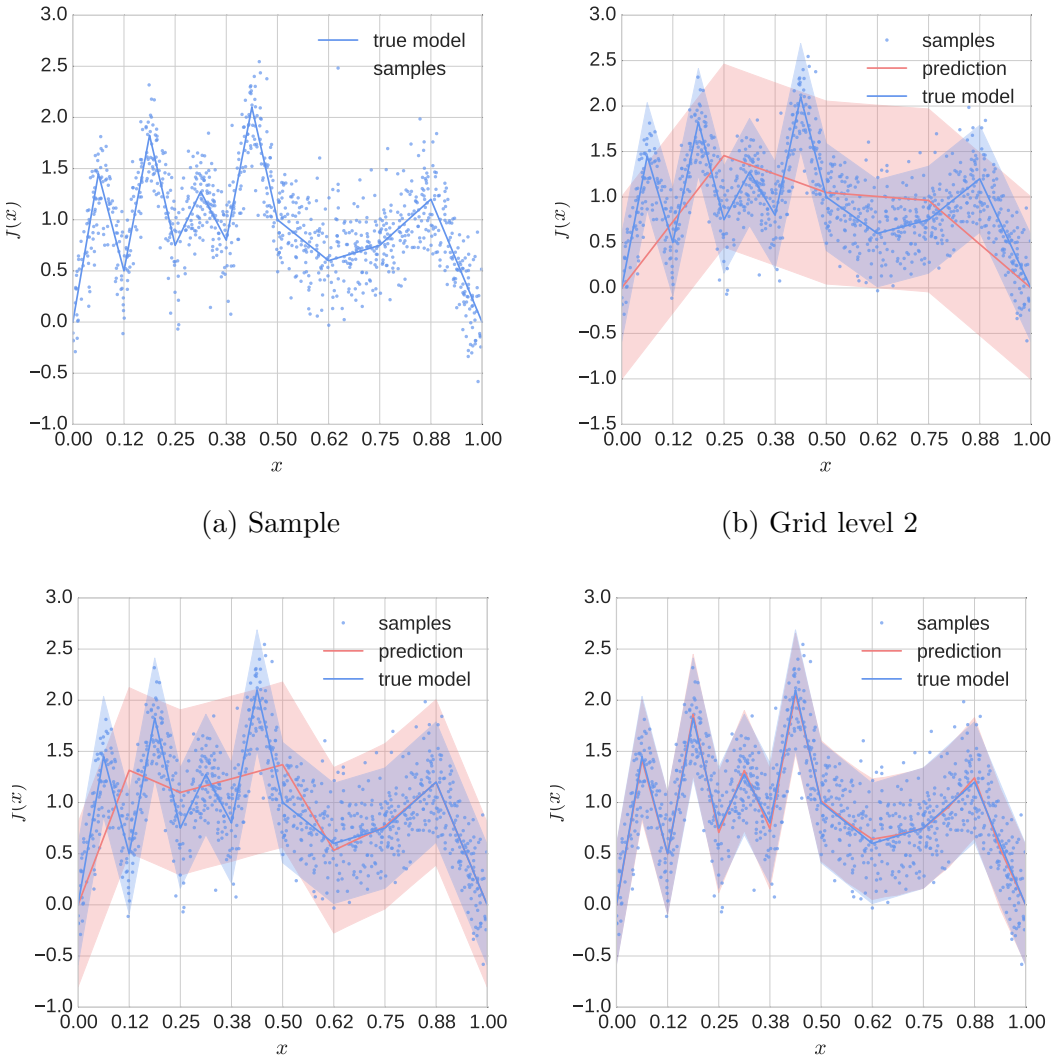
The two summands in (2.64) represent the noise in the data and the uncertainty of the parameters $\mathbf{w}$. We illustrate the form and the behaviour of this uncertainties in Figures 2.8 and 2.9. A sparse grid interpolant generates data distorted by the additive white noise with the standard deviation 0.3.

Figures 2.8b–2.8d illustrate the true and predicted functions as well as the 95% confidence intervals (roughly twice the standard deviation) of the posterior and data distributions. Three regular sparse grids with different maximum levels were used to construct the models, while the hyperparameters were estimated using the equations described in this section.

Models with lower level grids show higher overall variance σ_N^2, although one can see that even in this case most (roughly 95%) of observations fall within the confidence interval. As the complexity of the predictive model starts to resemble the complexity of the generating model (Fig. 2.8d) the agreement between the distributions becomes almost perfect.

We also illustrate the behaviour of the hyperparameters and variance components in Fig. 2.9. If the grid level is fixed and N grows, σ_{mod}^2 vanishes and the additive data noise σ_{ε}^2 dominates the variance of the predictive distribution. If we fix the dataset size and increase the gird level (Fig. 2.9b), we observe the decrease of the optimal hyperparameters—and effect that can result in overfitting eventually. However, we also see the strong rise of variance component associated with model complexity. For large high-dimensional models with high complexity and insufficient data this term may exceed the noise in data and dominate the uncertainty.

Regarding sparse grid models as Bayesian linear regressors immediately rewards us with a well-developed and applicable theory. We can now not only predict a new value using a sparse grid model, but also to assign a confidence to this prediction. Moreover, it becomes possible to interpret the regularisation term as a belief of the prior probability distribution of $\mathbf{w}$. This interpretation, in turn, explicates the need to adapt the regularisation function if the problem assumptions change.



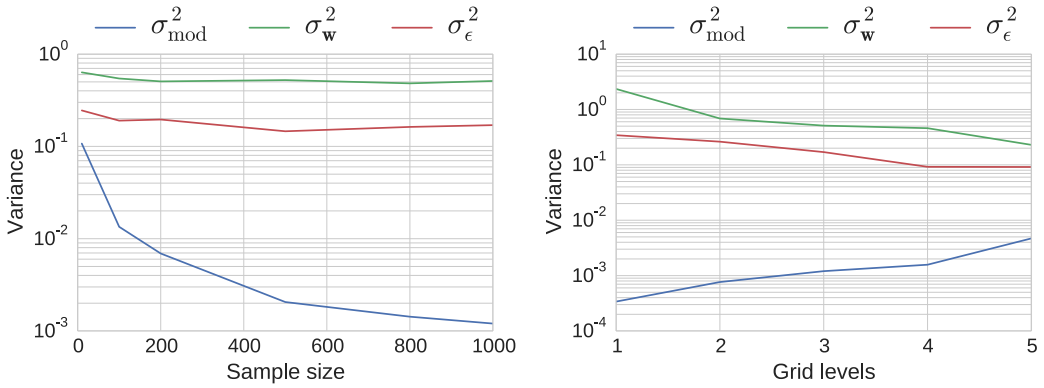
(a) Sample

(b) Grid level 2

(c) Grid level 3

(d) Grid level 4

Figure 2.8.: Sparse grid as a Bayesian linear regressor with depicted uncertainty. Figures show the true and predicted functions as well as the 95% confidence intervals of the posterior and data distributions. Lower grid level corresponds to higher overall variance. Level 4 of the esimtated model matches the level of the true model, in this case the agreement between the distributions becomes almost perfect.



(a) Sample size (b) Levels

Figure 2.9.: Dependence of variance components from level and sample size. Left: for the fixed grid level as the sample size grows the model variance σ^2_{mod} vanishes and the additive data noise σ^2_{ε} dominates the variance of the predictive distribution. Right: for the fixed dataset size as the grid level increases the optimal hyperparameters decrease while the model variance increases.

2.4.3. Automatic Relevance Determination

Bayesian interpretation of sparse grids in form (2.46)–(2.48) gives us a tool for understanding and identification of the optimal regularisation parameter. It also explicates the choice of the prior covariance matrix $\mathbf{S}_0$. The interpretation of the Tykhonov regularisation term as the prior covariance matrix $\sigma^2_{\mathbf{w}}\mathbf{I}$ can make one wonder about its ubiquitous applicability. Can we really assume that all weights have the same prior variance?

The hierarchical basis functions approximate different frequencies and we want to avoid learning noise that usually has high frequencies. Hence, we may want to constrain the fluctuations of the basis functions on higher levels by assuming lower prior variance. For example, we can reduce the variance by the factor 4 as the level sum of the grid basis functions increases:

$$\mathbf{S}_0 := \operatorname{diag}((2^{-2(|\mathbf{l}_j|_1 - D + 1)}\sigma^2_{\mathbf{w}})_{j=1}^{M}).$$

This covariance matrix is based on the expected decrease of coefficients for functions from $\mathcal{H}^2_{\text{mix}}$ (Bungartz & Griebel, 2004, Lemma 3.3). However, if we cannot make such a strict assumption or if the function exhibits some local irregularities, this prior is ill-suited.

Instead, we consider a more general form

$$\mathbf{S}_0 := \operatorname{diag}(\boldsymbol{\varsigma}), \quad \text{with}\, \boldsymbol{\varsigma} = (\varsigma_1, \ldots, \varsigma_M)^T. \tag{2.65}$$

and we maximise the type-2 marginal likelihood (2.57) with respect to the hyperparameter vector $\boldsymbol{\theta} := (\boldsymbol{\varsigma}, \sigma_\varepsilon^2)^T$. We can now simply compute the derivatives with respect to hyperparameters $\boldsymbol{\varsigma}$ and σ_ε^2 and set them equal to zero. This leads to the re-estimation equations (Tipping, 2001)

$$\gamma_j = 1 - \varsigma_i (\mathbf{S}_N)_{jj}, \tag{2.66}$$

$$\varsigma_j^{\text{new}} = \frac{\gamma_j}{w_j^2}, \tag{2.67}$$

$$(\sigma_\varepsilon^2)^{\text{new}} = \frac{\|\mathbf{y} - \boldsymbol{\Phi}\mathbf{w}\|^2}{N - \sum_{j=1}^{M} \gamma_j}, \tag{2.68}$$

where $(\mathbf{S}_N)_{jj}$ is the j-th diagonal element of $\mathbf{S}_N$ from (2.47), w_j denotes the j-th element of $\mathbf{w}_N$ defined in (2.48), and γ_j reflects how well the parameter w_j is determined by the data.

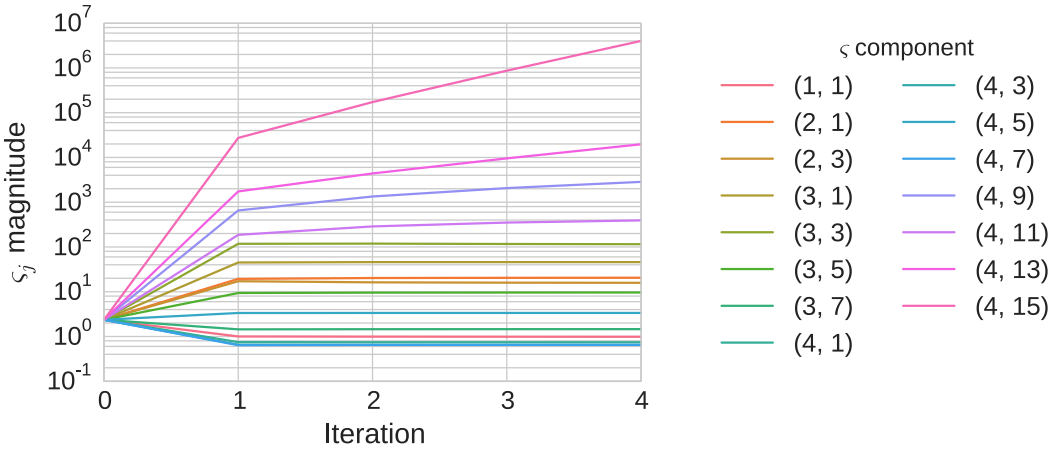
In the optimisation procedure, we initialise $\boldsymbol{\varsigma}$ and σ_ε^2 and estimate the posterior mean and the posterior covariance matrix using (2.47) and (2.48). Then we alternate between re-estimating of the hyperparameters using (2.67) and (2.68) and the posterior statistics using (2.47) and (2.48).

Figure 2.10 illustrates the components of the hyperparameter vectors $\boldsymbol{\varsigma}$, as well as of the vectors $\boldsymbol{\gamma}$ and $\mathbf{w}_N$ for the automatic relevance selection algorithm for the problem in Fig. 2.8 and 5 000 data samples. Optimisation procedure forces some of the hyperparameters ς_j to large values, effectively leading the prior variance of this components to zero. Those are exactly the components of a regular level 4 sparse grid that the generating function is missing.

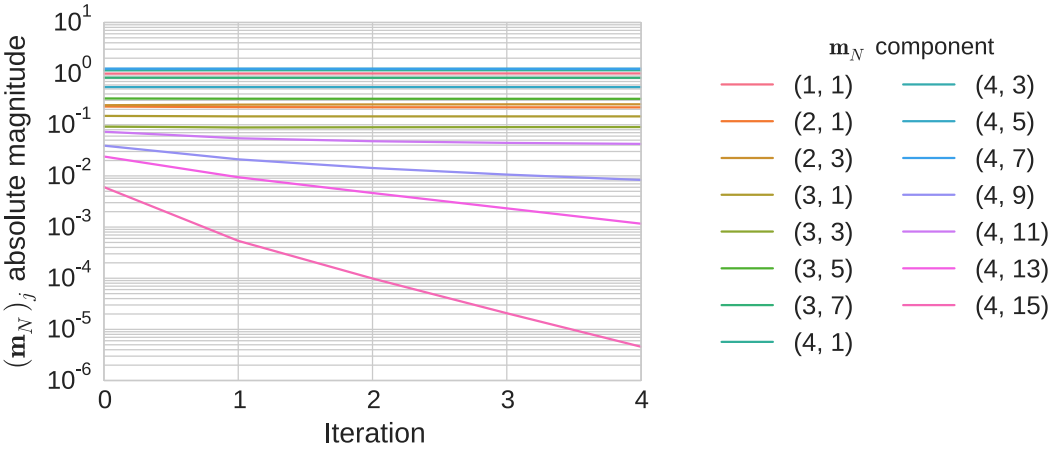
Automatic relevance determination algorithm allows us to identify the optimal structure of the sparse grid model. Unfortunately, the required computational costs are substantial. In the example above it took just a few iterations to identify the irrelevant basis functions, but in a general case the procedure can be expensive due to the need of computing the diagonal of the inverse matrix $(\mathbf{S}_N)_{jj}$. The procedure can be improved, which leads to a much more efficient Sequential Sparse Bayesian Learning Algorithm (Tipping & Faul, 2003). The required amount of computation, however, still scales as $\mathcal{O}(M^3)$. Moreover, Wipf and Nagarajan (2008) have shown that the relevance detection problem can also be approached as an iteratively reweighted ℓ_1 minimisation problem

$$\mathbf{w}^{t+1} = \arg\min_{\mathbf{w}} \|\mathbf{y} - \boldsymbol{\Phi}\mathbf{w}\|_2^2 + \sum_{j=1}^{M} \lambda_j^{(t)} |w_j|. \tag{2.69}$$

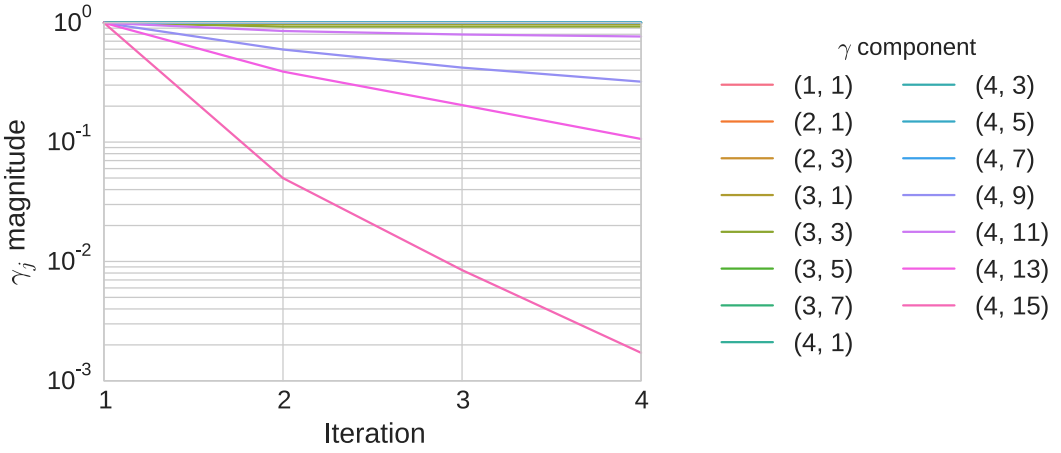
We will discuss the approach of ℓ_1-regularisation in the following chapter.



(a) $\boldsymbol{\varsigma}$



(b) $\mathbf{w}_N$



(c) $\boldsymbol{\gamma}$

Figure 2.10.: Relevance components estimation. Optimisation procedure (2.66)–(2.68) increases ς_j of the unimportant components, effectively leading the prior variance of this components to zero.

2.4.4. Kernel Machines

The feature map (2.41) together with the dot product on the underlying vector space yields a kernel function (Scholkopf & Smola, 2001)

$$k : [0,1]^D \times [0,1]^D \to \mathbb{R}, \quad (\mathbf{x}, \mathbf{x}') \mapsto \langle \boldsymbol{\phi}(x), \boldsymbol{\phi}(x') \rangle. \tag{2.70}$$

This kernel function leads to a symmetric positive definite Gram matrix $\mathbf{K}$ with elements $\mathbf{K}_{ij} := (k(\mathbf{x}_i, \mathbf{x}_j))_{ij}$. One can easily validate this, since for all $c_i \in \mathbb{R}, x_i \in [0,1]^D, i = 1, \ldots, N$, we have:

$$\begin{aligned} \mathbf{c}^T \mathbf{K} \mathbf{c} &= \sum_i c_i \sum_j c_j \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle = \sum_i c_i \left\langle \boldsymbol{\phi}(\mathbf{x}_i), \sum_j c_j \boldsymbol{\phi}(\mathbf{x}_j) \right\rangle \\ &= \left\langle \sum_i c_i \boldsymbol{\phi}(\mathbf{x}_i), \sum_j c_j \boldsymbol{\phi}(\mathbf{x}_j) \right\rangle = \left\| \sum_i c_i \boldsymbol{\phi}(\mathbf{x}_i) \right\|^2 \geq 0. \end{aligned}$$

Let f and g be two functions defined as linear combinations of kernels

$$f(\cdot) = \sum_{i=1}^{N} \alpha_i k(\cdot, \mathbf{x}_i), \quad g(\cdot) = \sum_{j=1}^{N'} \beta_j k(\cdot, \mathbf{x}'_j) \tag{2.71}$$

with arbitrary $N, N' \in \mathbb{N}$, $\alpha_i, \beta_j \in \mathbb{R}$ and $\mathbf{x}_i, \mathbf{x}'_j \in [0,1]^D$. Let us now define a dot product between f and g as

$$\langle f, g \rangle := \sum_{i=1}^{N} \sum_{j=1}^{N'} \alpha_i \beta_j k(\mathbf{x}_i, \mathbf{x}'_j). \tag{2.72}$$

One can show that the space of function (2.71) with the dot product (2.72) completes a Hilbert space $\mathcal{H}$ (a complete vector space with a dot product), called the reproducing kernel Hilbert space (RKHS) associated with kernel k, called a *reproducing kernel* (Scholkopf & Smola, 2001). The Representer theorem warrants us an ability to solve machine learning problems in the kernel form.

Theorem 2.1 (Representer theorem (Kimeldorf & Wahba, 1971; Scholkopf & Smola, 2001))**.** *Let $\Omega : [0, \infty] \to \mathbb{R}$ be a strictly monotonic increasing function, let $\mathcal{X}$ be the input set and $L : (\mathcal{X} \times \mathbb{R}^2)^N \to \mathbb{R} \cup \{\infty\}$ be a loss function. Each minimiser $f \in \mathcal{H}$ of the regularised risk functional*

$$L((\mathbf{x}_1, y_1, f(\mathbf{x}_1)), \ldots, (\mathbf{x}_N, y_N, f(\mathbf{x}_N))) + \Omega(\|f\|_{\mathcal{H}}) \tag{2.73}$$

can be represented as

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}).$$

The use of a regularisation operator Γ in (2.73) modifies the dot product and hence the RKHS:

$$\langle f, g\rangle_{\mathcal{H}} = \langle \Gamma f, \Gamma g\rangle_{L_2} = \int_{\mathcal{X}} \Gamma f(\mathbf{x}) \Gamma g(\mathbf{x}) \mathrm{d}\mathbf{x}.$$

We hence can think the regularisation operators as of ways to enforce certain function properties such as smoothness, bounded derivatives, etc. Note that this is an alternative interpretation to the one given in Sec. 2.4.2.

The corresponding reproducing kernel has the form

$$k(\mathbf{x}, \mathbf{x}') := \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\rangle_{\mathbf{M}} = \langle \boldsymbol{\phi}(\mathbf{x}), \mathbf{M}\boldsymbol{\phi}(\mathbf{x}')\rangle \tag{2.74}$$

with a positive definite matrix $\mathbf{M}$, which is more general than the one defined in (2.70). What is $\mathbf{M}$ for the sparse grid ridge regression from Sec. 2.3? Following proposition gives the answer.

Proposition 2.2. *Let $\mathbf{x}_i \in [0,1]^D$ be an input vector, $\boldsymbol{\phi}_j(\mathbf{x}) : [0,1]^D \to [0,1]$ the sparse grid basis functions, $\boldsymbol{\Phi}$ be the $N \times M$ feature matrix*

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}(\mathbf{x}_1) & \boldsymbol{\phi}(\mathbf{x}_2) & \cdots & \boldsymbol{\phi}(\mathbf{x}_N) \end{bmatrix}^T = \begin{bmatrix} \phi_1(\mathbf{x}_1) & \cdots & \phi_M(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_N) & \cdots & \phi_M(\mathbf{x}_N) \end{bmatrix}, \tag{2.75}$$

and $\boldsymbol{\Gamma} \in \mathbb{R}^{M\times L}$ with $M \leq L$ such that $\mathbf{D} := \boldsymbol{\Gamma}^T\boldsymbol{\Gamma}$, $\mathbf{D} \in \mathbb{R}^{M\times M}$ non-singular. Then minimisation of

$$\min_{\mathbf{w}\in\mathbb{R}^M} (\mathbf{y} - \boldsymbol{\Phi}\mathbf{w})^T(\mathbf{y} - \boldsymbol{\Phi}\mathbf{w}) + \lambda\|\boldsymbol{\Gamma}\mathbf{w}\|_2^2 \tag{2.76}$$

corresponds to the solution of the kernel equation

$$(\mathbf{K} + \lambda\mathbf{I})\boldsymbol{\alpha} = \mathbf{y}, \tag{2.77}$$

where $\mathbf{K}$ is defined by

$$k(\mathbf{x}, \mathbf{x}') := \left\langle \boldsymbol{\phi}(\mathbf{x}), \mathbf{D}^{-1}\boldsymbol{\phi}(\mathbf{x}')\right\rangle. \tag{2.78}$$

Proof. The solution of the minimisation problem (2.76) leads to

$$\mathbf{w} = \left(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\mathbf{D}\right)^{-1}\boldsymbol{\Phi}^T\mathbf{y}. \tag{2.79}$$

Assuming that $\mathbf{D}$ is invertible, we can use the result (A.4) to obtain

$$\mathbf{w} = \mathbf{D}^{-1}\boldsymbol{\Phi}^T\left(\boldsymbol{\Phi}\mathbf{D}^{-1}\boldsymbol{\Phi}^T + \lambda\mathbf{I}\right)^{-1}\mathbf{y}. \tag{2.80}$$

Notice that from (2.78) the Gram matrix can be calculated as

$$\mathbf{K} = \mathbf{\Phi}\mathbf{D}^{-1}\mathbf{\Phi}^T. \tag{2.81}$$

Hence, by solving the kernel equations (2.77), we can obtain the representation of the function

$$\mathbf{f} = \mathbf{\Phi}\mathbf{w} = \mathbf{K}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y} = \mathbf{K}\boldsymbol{\alpha} \tag{2.82}$$

and in general we have

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}, \mathbf{x}_i) = \sum_{i=1}^{N} \alpha_i \langle \boldsymbol{\phi}(\mathbf{x}), \mathbf{D}^{-1}\boldsymbol{\phi}(\mathbf{x}_i)\rangle. \tag{2.83}$$

□

From (2.80) it also follows that if we find the kernel coefficients $\boldsymbol{\alpha} := (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}$, we can obtain $\mathbf{w} = \mathbf{D}^{-1}\mathbf{\Phi}^T\boldsymbol{\alpha}$. This is especially easy if $\mathbf{D} = \mathbf{I}$. In this case, we get

$$\mathbf{w} = \mathbf{\Phi}^T\boldsymbol{\alpha}.$$

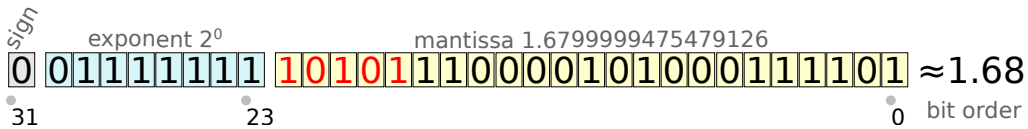
Kernel k described in (2.83) is the *regularised sparse grid kernel* associated with the regularisation matrix $\mathbf{D}$. Note that the inverse matrix $\mathbf{D}^{-1}$ is usually full even if $\mathbf{D}$ is sparse. An exception is $\mathbf{D} = \mathbf{I} = \mathbf{D}^{-1}$.

Sparse Grid Kernel Trick

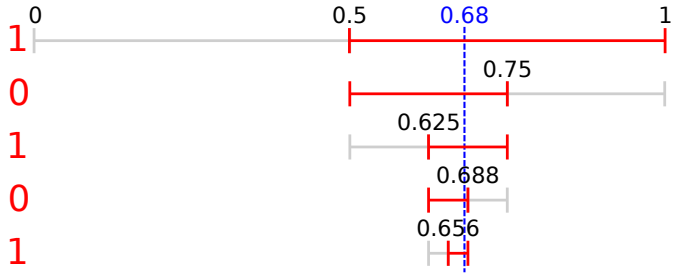
The advantage of using a kernel as a similarity measure is that it allows us to create algorithms in dot product spaces. Some of the kernels can be evaluated efficiently even though they correspond to dot products in infinite dimensional dot product spaces. In such cases, evaluation of $k(\mathbf{x}, \mathbf{x}')$ without the explicit evaluation of $\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\rangle$ is crucial. In the machine learning community this substitution is called *kernel trick*. Given an efficient way to compute k, it can be more efficient than feature maps especially when $N \ll M$.

Exploiting the structure of hierarchical subspaces significantly accelerates the evaluation of the sparse grid kernel (2.78). We use the idea that every grid point lies on the support of only one function in every hierarchical subspace. Buse (2015, Sec. 4.3.2) exploits this scheme for evaluation of dimensional adaptive sparse grids. Depending on the size and dimensionality of the grid, he achieves a speedup of up to 7.4 compared to the naive subspace-wise evaluation of basis functions.

As for the kernel evaluation the points are considered pairwise, we can limit the evaluation to only those basis functions that support both points. For the fast identification of these functions we make use of the binary representation of real numbers as $\sum_{i=1} b \cdot 2^{-i}, b \in \{0, 1\}$. This representation can be computed explicitly using a simple Alg. 2 or it is given implicitly, e.g. as elements of the IEEE 754 format for the floating-point number representation.



(a) Representation of a floating-point number using single-precision IEEE 754-1985 format



(b) Binary search using representation in IEEE 754 format

Figure 2.11.: Usage of binary representation of a floating point number for efficient grid search. In this example the point coordinate is 0.68.

The use of binary encoding is illustrated in Fig. 2.11. In order to fix the exponent for all numbers, we add 1.0 to the coordinate. It is safe to do this, if we know that all coordinates are between 0 and 1 (Fig. 2.11a).[1]

To determine the hierarchical basis functions supporting the point, we use the binary pattern of the mantissa. We start with the highest bit and hierarchically descend to the left descendant basis function if the bit is 0 or to the right function if the bit is 1 (Fig. 2.11b).

We can also use the bit operations on the representations of two coordinates to estimate the finest most basis function that covers both points.

We can also estimate the finest basis function that covers both points from the overlap of the two bit patterns. Alternatively, for every pair of coordinates x and x' we can estimate the maximum level p such that a sparse grid function at this level contains both points:

$$p = \lceil -\log_2(|x - x'|) \rceil. \tag{2.84}$$

Sometimes, p will indicate the level where the points already lie on different supports and the effective maximal level is smaller. Having calculated the maximum level, we can determine the binary search path that identifies hierarchical descendent functions that are nonzero at the coordinate. As shown in Alg. 2, we successively divide the coordinates by the powers of 2 to determine the binary representation.

[1] We need to exclude the coordinate 1.0 for simplicity of presentation. This trivial case can be handled explicitly.

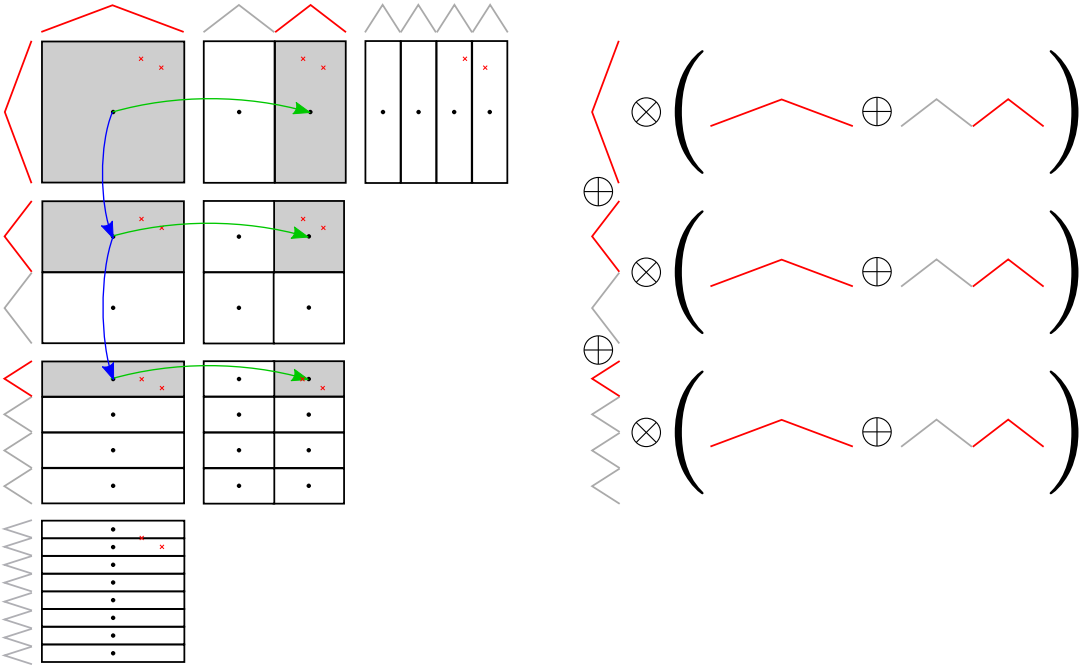


Figure 2.12.: Illustration of the recursive fast kernel evaluation scheme using distributive law. The algorithm starts with iteration through the dimension 0 and then continues the recursive descent into other dimensions for all function levels that still cover both points.

Having established the binary form (it is enough to establish it for only one of two points), we recursively evaluate the basis functions in both points and accumulate the products as described in Alg. 3.

We illustrate the recursive path of Alg. 3 and the underlying mathematical evaluation form in Fig. 2.12. We start by iterating through the dimension 0 and then continue the recursive descent into other dimensions for all function levels that still cover both points. Mathematically, this is equivalent to applying the distributive law.

What is the complexity of Alg. 3? The recursive function considers every hierarchical subspace where both input points rest on the same support. Hence, the complexity of the algorithm is proportional to the number of these subspaces.

Since every hierarchical subspace is identified by a level vector $\mathbf{l}$ and the sparse grids are defined such that the sum of levels may not exceed $\ell + D - 1$, we know that the *total* number of hierarchical subspaces of a sparse grid is (Bungartz & Griebel, 2004)

$$\begin{aligned} |\{W_{\mathbf{l}} \mid |\mathbf{l}|_1 \leq \ell + D - 1\}| &= \sum_{i=0}^{\ell-1} \binom{D-1+i}{D-1} \\ &= \binom{\ell + D - 1}{D} \in \mathcal{O}\left(\frac{(\ell + D)^D}{D!}\right). \end{aligned}$$

Algorithm 2: `GetBinaryForm`: Compute the common binary form of the coordinates

input : Point coordinates $x, x' \in [0,1]$, grid maximum level $l_{\max}$
output: Binary path $\mathbf{p}$

initialise $\mathbf{p} \leftarrow (0)$, $l \leftarrow 2^{-1}$
while $x > 0 \wedge l > 2^{l_{max}}$ **do**
  $b \leftarrow \lfloor x/l \rfloor$; $b' \leftarrow \lfloor x'/l \rfloor$
  if $b \neq b'$ **then**
    break
  $\mathbf{p} \leftarrow (\mathbf{p}^T, b)^T$
  $x \leftarrow x - b \cdot l$
  $l \leftarrow l/2$

Algorithm 3: EvalKernelRec($\mathbf{x}$,$\mathbf{x}'$,d,$l_{\max}$) Recursive kernel evaluation

input : Points $\mathbf{x}, \mathbf{x}' \in [0,1]^D$, current dimension d, grid maximum level $l_{\max}$,
output: Value of the sparse grid kernel k

initialise $k \leftarrow 0, i \leftarrow 1$
compute 1d search path $\mathbf{p} \leftarrow$`GetBinaryForm`$(\mathbf{x}[d], \mathbf{x}'[d], l_{\max})$
Let p_d *be the length of path* $\mathbf{p}$*:* $p_d = \dim(\mathbf{p})$
for $l \leftarrow 0$ **to** $p_d - 1$ **do**
  compute the next index on the path: $i \leftarrow 2i - (-1)^{\mathbf{p}[l]}$
  $\kappa \leftarrow \phi(\mathbf{x}[d], l+1, i) \cdot \phi(\mathbf{x}'[d], l+1, i)$
  if $d < D - 1$ **then**
    $k \leftarrow k + \kappa \cdot$`EvalKernelRec` $(\mathbf{x}, \mathbf{x}', d+1, l_{\max} - l)$
  else
    $k \leftarrow \kappa$

However, as in every dimension d the level can only vary between 1 and p_d, the number of actually considered subspaces is smaller. Calculating the number of such subspaces reduces to a combinatorial counting problem of restricted compositions as the following proposition suggests.

Proposition 2.3. *Let $p_1, \ldots, p_D$ be the lengths of the binary representations as returned by* `GetBinaryForm` *for the dimensions $1, \ldots, D$. Then Alg. 3 evaluates*

$$\sum_{l=D}^{\ell+D-1} F(l, D; p_1, \ldots, p_D) \tag{2.85}$$

hierarchical subspaces with

$$F(l, D; p_1, \ldots, p_D) = \binom{l-1}{D-1} + \sum_{j=1}^{D} (-1)^j \sum^{*} \binom{l - 1 - p_{i_1} - p_{i_2} - \cdots - p_{i_j}}{D-1}, \tag{2.86}$$

and the summation $\sum^{*}$ taken over all j-combinations $i_1 < i_2 < \cdots < i_j$ of D.

Proof. Let $F(l, D; p_1, \ldots, p_D)$ be the number of hierarchical subspaces where the components of the level vector $\mathbf{l} = (l_1, \ldots, l_D)^T$ such that

1. the components sum up to l:

$$\sum_{d=1}^{D} l_d = l;$$

2. the components are restricted:

$$1 \leq l_d \leq p_d, \text{ for each } 1 \leq d \leq D.$$

In combinatorical sense, $F(l, D; p_1, \ldots, p_D)$ calculates the number of restricted compositions (ordered partitions) of l. Then Equation (2.86) follows directly from (Abramson, 1976, Example (B)).

Since the sum of the level vector components l can take any value between D and $\ell + D - 1$, the total number of hierarchical subspaces equals the sum (2.85). □

Proposition 2.3 gives the recipe for calculating the exact numer of function evaluations in Alg. 3 by establishing the connection to a standard combinatorial problem and applying the known results from combinatorial mathematics. Unfortunately, it is not straightforward to bound (2.85) tightly from above to obtain the complexity class of the Alg. 3. The summation in (2.85) over the first binomial coefficient in (2.86) suggests exponential dependency on D. The same is evident from Fig. 2.13, which shows the growth of the number of subspaces

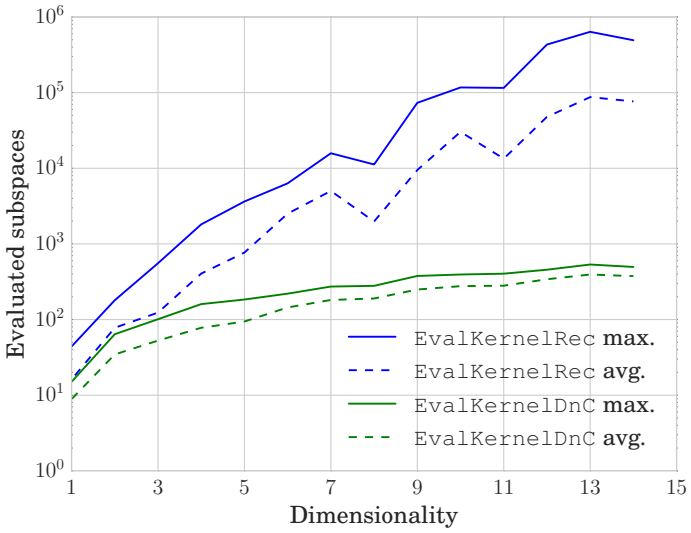


Figure 2.13.: Number of evaluated subspaces for Algorithms 3 and 4. For the function `EvalKernelRec` of Alg. 3 the number of evaluated subspaces depends exponentially on the dimensionality, while for the function `EvalKernelDnC` of Alg. 4 this dependency is quadratic. Here we see an empirical estimation of the number of evaluated subspaces obtained for 2 000 randomly sampled point pairs with the dimensionality between 1 and 15. The maximum and the average number of subspaces is shown.

obtained using Formula (2.85) for 2 000 randomly sampled values of $(p_1, \ldots, p_D)$ with different dimensionality.

The exponential dependency on D brings us back to the curse of dimensionality. This dependency, however, is characteristic for many algorithms for grid evaluation, where every hierarchical subspace is visited at least once. Our analysis yields two further conclusions:

1. Let us imagine a sparse grid with an arbitrary large maximal level ℓ such that $\sum_{d=1}^{D} p_d < \ell + D - 1$ and we are only limited by the distance between two points and not by the maximum level of the particular sparse structure.[2] Then we can apply the distributive law: estimate the sums of one-dimensional basis function evaluations and multiply these sums to obtain the kernel value (as Fig. 2.12 (right) suggests). This leads to a linear dependency in D. We discuss this result in detail in Proposition 2.4.

2. The recursive path of the algorithm repeatedly runs into evaluation of the same branches. We hence can store the intermediate results from

[2]This construction is equivalent to a full grid.

those branches dramatically reducing the amount of required computations. This idea leads to Alg. 4.

Proposition 2.4. *For a sparse grid with arbitrary large maximum level $\ell \to \infty$ the averaged complexity of kernel evaluation is bounded by $(1.5/\ln 2 + 1) \cdot D$ evaluations.*

Proof. It is easy to see, and Fig. 2.12 demonstrates this explicitly, that for an arbitrarily deep sparse grid we can use the distributive law, summing up the evaluations in individual dimensions and multiplying the sums. Hence, the expected number of pairwise evaluations of functions is equal to

$$\begin{aligned}\mathbb{E}\left[\sum_{d=1}^{D} p_d\right] &= \sum_{d=1}^{D} \mathbb{E}\left[\lceil -\log_2(|x_d - x'_d|)\rceil\right] \leq \sum_{d=1}^{D} \mathbb{E}\left[-\log_2(|x_d - x'_d|) + 1\right] \\ &= D \cdot \left(-\int_0^1 \int_0^1 \log_2(|x - x'|)\mathrm{d}x \; \mathrm{d}x' + 1\right). \end{aligned} \tag{2.87}$$

In the last transformation we used the assumption that the variables in individual dimensions are sampled i.i.d. and uniformly.

Now, let us consider the integral above. We first transform the logarithm into a natural one and omit absolute values by considering only the positive part of (symmetrical) difference and then evaluate the internal and external integrals:

$$\begin{aligned}\int_0^1 \int_0^1 \log_2(|x - x'|)\mathrm{d}x \; \mathrm{d}x' &= \frac{2}{\ln 2} \int_0^1 \int_x^1 \ln(y - x)\mathrm{d}y \; \mathrm{d}x \\ &= \frac{2}{\ln 2} \int_0^1 \left[(y - x)\ln(y - x) - y\right]_x^1 \mathrm{d}x \\ &= \frac{2}{\ln 2} \int_0^1 (1 - x)\ln(1 - x) - 1 + x \,\mathrm{d}x = -\frac{2}{\ln 2} \cdot \frac{3}{4}.\end{aligned}$$

Substituting this result into (2.87) leads directly to the upper bound. □

In practice we observed that the average number of evaluations increases rather as $(1.5/\ln 2 + 0.5) \cdot D$ but we do not have a formal proof for this observation. Hence, for a sparse grid without the limiting ℓ the use of the distributive law leads to an average complexity that is linear in the number of dimensions! And for a large portion of point pairs and a sparse grid with higher maximum level this is what effectively happens.

For the other pairs, the number of evaluations in the same dimension varies with ℓ and hence changes in every call of `EvalKernelRec` for the same dimensions d.

The path of the recursive algorithm resembles a tree and many branches of this tree (calls to `EvalKernelRec` with the same parameters) would give the

same results. Hence, we derive a new divide-and-conquer Algorithm 4 for fast kernel evaluation where we store the intermediate results of evaluated branches.

To further accelerate the evaluation we reverse the **for**-loop evaluating the functions with the larger level l first. This way we can identify the largest-most branch of the tree right away. Algorithm 4 illustrates the resulting procedure. We store the intermediate results in the dictionary `cache` hashed using a `key`—a tuple with the current dimension, the level, and the remaining possible level sum for a regular sparse grid.

The formula for computation of the index in Line 6 may not be obvious at first glance. In fact, it is obtained from estimating the index from the path same as in Alg. 3. Let $\mathbf{p}$ be the vector with binary forms obtained from Alg. 2. Then we have

$$
\begin{aligned}
&2^{p_d} - \sum_{j=0}^{p_d-1} (-1)^{\mathbf{p}[j]} 2^{p_d-j-1} = 2^{p_d} - \sum_{j=0}^{p_d-1} (-2\mathbf{p}[j]+1) \cdot 2^{p_d-1-j} = \\
&2^{p_d} - \sum_{j=0}^{p_d-1} (-\mathbf{p}[j]) \cdot 2^{p_d-j} - \sum_{j=0}^{p_d-1} 2^{p_d-1-j} = 2^{p_d} - \sum_{j=0}^{p_d-1} 2^j + \sum_{j=0}^{p_d-1} 2^{p_d-j} \cdot \mathbf{p}[j] = \\
&2 - \frac{1-2^{p_d}}{1-2} + \sum_{j=0}^{p_d-1} 2^{p_d-j} \cdot \mathbf{p}[j] = 2^{p_d} - 2^{p_d} + 1 + 2^{p_d} \sum \mathbf{p}[j] 2^{-j} = \\
&1 + 2^{p_d} \left(x - (x \mod 2^{p_d-1}) \right) = 1 + 2 \cdot \lfloor 2^{p_d-1} x \rfloor .
\end{aligned}
$$

Lemma 2.5. *Let $p_1, \ldots, p_D$ be the maximum number of possible evaluations in the respective dimensions. The size of the intermediate result table* `cache` *and hence the number of evaluations of Alg. 4 is*

$$
\sum_{d=1}^{D-1} \sum_{t=0}^{p_{:d-1}-(d-1)} \lfloor \min\{p_d, \ell - t\} \rfloor_+ + p_d \tag{2.88}
$$

with $p_{:d-1} := \sum_{i=0}^{d-1} p_i$.

Proof. The proof is constructive. The number of different `key`s for the dimension d is limited by the sum of levels of the basis function of the previous $d-1$ dimensions. This sum of levels is at least $d-1$ and at most $p_{:d-1}$. For a particular sum of levels so far, let us call it λ, the level of the basis function in the dimension d can vary between 1 and $\min\{l_{\max} + D - 1 - \lambda - (D-d), \ell\}$. The subtraction of $(D-d)$ is due to the fact that the basis functions in the following $D-d$ dimensions need to have at least level 1. In the final dimension we do not worry about the sum of levels (we set it to -1 anyway). So the number of

Algorithm 4: `EvalKernelDnC` Fast divide-and-conquer kernel evaluation.

input : Points $\mathbf{x}, \mathbf{x}' \in [0,1]^D$, current dimension d, grid maximum level $l_{\max}$,

output: Value of the sparse grid kernel k

initialise $k \leftarrow 0, i \leftarrow 1, r \leftarrow 1$

compute l_{max} *from* $\mathbf{x}[d]$ *and* $\mathbf{x}'[d]$ *using* (2.84)

$l_{\max} = \min\{l_{\max}, l_{\max}\} - 1$

compute 1d search path $\mathbf{p} \leftarrow$`GetBinaryForm`$(\mathbf{x}[d], \mathbf{x}'[d], l_{\max})$

Let p_d *be the length of path* $\mathbf{p}$*:* $p_d \leftarrow \dim(\mathbf{p})$

compute the last index $i = 2\lfloor 2^{p_d - 1}\mathbf{x}[d] \rfloor + 1$

for $l \leftarrow p_d - 1$ **to** *0* **do**

  $\texttt{key} = \begin{cases} (d, l, l_{\max} + D - 1 - l) & d < D \\ (d, l, -1) & \text{else} \end{cases}$

  if *cache*[*key*] *is already precomputed* **then**

    $k \leftarrow k + \texttt{cache}[\texttt{key}]$

    break

  else

    if $d < D$ **then**

      $r \leftarrow$ `EvalKernelDnC`$(\mathbf{x}, x', d+1, l_{\max} - l)$

    $\kappa \leftarrow \phi(\mathbf{x}[d], l+1, i) \cdot \phi(\mathbf{x}'[d], l+1, i)$

    backward update: add the result from smaller levels, if available

    $\texttt{key}_{\text{next}} = \begin{cases} (d, l-1, l_{\max} + D - 1 - (l-1)) & d < D \\ (d, l-1, -1) & \text{else} \end{cases}$

    if *cache*[$\texttt{key}_{next}$] *is already precomputed* **then**

      $\texttt{cache}[\texttt{key}] += \texttt{cache}[\texttt{key}_{\text{next}}]$, $k += \texttt{cache}[\texttt{key}_{\text{next}}]$

      break

    else

      forward update: add the result to large levels if those were precomputed before

      $\texttt{key}_{\text{prev}} = \begin{cases} (d, l+1, l_{\max} + D - 1 - (l+1)) & d < D \\ (d, l+1, -1) & \text{else} \end{cases}$

      while *cache*[$\texttt{key}_{prev}$] *is already precomputed* **do**

        $\texttt{cache}[\texttt{key}_{\text{prev}}] += \texttt{cache}[\texttt{key}]$

        $\texttt{key}_{\text{prev}} = \begin{cases} (d, \texttt{key}_{\text{prev}}[2] + 1, \texttt{key}_{\text{prev}}[3] - 1) & d < D \\ (d, \texttt{key}_{\text{prev}}[2] + 1, -1) & \text{else} \end{cases}$

    $k \leftarrow k + \texttt{cache}[\texttt{key}]$

    compute the next index on the path $i \leftarrow (i + (-1)^{\mathbf{p}[l]})/2$

different `cache` entries is p_D. Altogether we have:

$$\sum_{d=1}^{D} \sum_{\lambda=d-1}^{L_{:d-1}} |\min\{p_d, \ell + D - 1 - \lambda - (D-d)\}|_+ + L[D] =$$
$$\sum_{d=1}^{D} \sum_{\lambda=d-1}^{L_{:d-1}} |\min\{p_d, \ell - (\lambda - (d-1))\}|_+ + L[D] =$$
$$\sum_{d=1}^{D-1} \sum_{t=0}^{L_{:d-1}-(d-1)} |\min\{p_d, \ell - t\}|_+ + L[D].$$

□

From Lemma 2.5 it follows directly that the complexity of Alg. 4 is in $\mathcal{O}(D^2\ell)$ in the worst case. Figure 2.13 provides an empirical evidence that the divide-and-conquer Algorithm 4 scales well with the number of dimensions.

The kernel trick algorithm combines the sparse grid models and the kernel machines. As the complexity of our kernel trick is only quadratic in the number of attributes, it effectively breaks the curse of dimensionality making the combination of sparse grids and kernel machines especially attractive for high-dimensional problems with few data points.

2.4.5. Linear Smoothers

If the dependence of the target variable y and the input $\mathbf{x}$ is assumed to be smooth, the underlying nonparametric regression method is called a scatterplot smoother (Buja et al., 1989). The linear smoother's estimate $\mathbf{f}$ can be expressed as a linear transformation of the observed target values $\mathbf{y}$ with the so-called smoother matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$:

$$\mathbf{f} = \mathbf{S} \cdot \mathbf{y}.$$

Examples of linear smoothers are running means, locally weighted running lines, kernel smoothers, and smoothing splines (Hastie et al., 2011).

With (2.79) and (2.82) we can express sparse grid model as a linear smoother with estimates

$$\mathbf{f} = \mathbf{S}\mathbf{y} := \mathbf{\Phi} \left(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{D}\right)^{-1} \mathbf{\Phi}^T\mathbf{y}.$$

Hence, the sparse grid smoother matrix is defined as

$$\mathbf{S} := \mathbf{\Phi} \left(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{D}\right)^{-1} \mathbf{\Phi}^T. \tag{2.89}$$

An analysis of the eigenvalues and eigenvectors of the smoother matrix gives us an alternative view of the data transformation for a sparse grids approximation.

Proposition 2.6. *The eigenvalues of the smoother matrix* $\mathbf{S}$ *defined in* (2.89) *lies in the interval [0,1).*

Proof. Let $\mathbf{D}$ be an arbitrary symmetric positive definite matrix. From (2.89) we can write the eigenvalue problem as

$$\mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{D})^{-1}\mathbf{\Phi}^T\mathbf{w} = \gamma\mathbf{w}, \tag{2.90}$$

where γ is an eigenvalue and $\mathbf{w}$ is an eigenvector. After multiplying both sides of the equation by $\mathbf{\Phi}^T$ from left we obtain

$$\mathbf{\Phi}^T\mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{D})^{-1}\mathbf{\Phi}^T\mathbf{w} = \gamma(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{D})(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{D})^{-1}\mathbf{\Phi}^T\mathbf{w}.$$

Hence, with $\mathbf{z} := (\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{D})^{-1}\mathbf{\Phi}^T\mathbf{w}$ we arrive at the generalised eigenvalue problem

$$\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{z} = \gamma(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{D})\mathbf{z} \tag{2.91}$$

with the same eigenvalue γ as that of $\mathbf{S}$.

The Rayleigh quotient associated with (2.91) satisfies

$$0 \leq \frac{\mathbf{z}^T\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{z}}{\mathbf{z}^T(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{D})\mathbf{z}} = \frac{a}{a + \lambda b} < 1.$$

The inequality holds since both $\mathbf{\Phi}^T\mathbf{\Phi}$ and $\mathbf{D}$ are positive definite and hence a and b are positive. It follows that the spectrum of $\mathbf{S}$ lies in the interval [0,1). □

As the matrix $\mathbf{S}$ explains the transformation of target values to the predictions, let us look at the eigenvalues and eigenvectors of this matrix to gain a better insight into the sparse grid regression.

Since the image of $\mathbf{S}$ is the same as of $\mathbf{\Phi}$, it is clear that $\mathbf{S}$ would have at most rank($\mathbf{\Phi}$) nonzero eigenvalues. Figure 2.14 plots these eigenvalues for the one-dimensional sparse grid with the maximum level 3, computed using 2 000 points uniformly distributed at $[0, 1]$. It is surprising that for even large values of the regularisation parameter λ of 0.1 and 0.01 the nonzero eigenvalues are almost 1. We made the same observations for higher dimensions as well.

Figure 2.15 shows the eigenvectors of $\mathbf{S}$ against the respective coordinates in x dimension. Regularisation does not change the forms of the eigenvectors. In fact, the eigenvectors correspond to the different frequencies m of the sinus function

$$a \cdot \sin(mx\pi).$$

This becomes evident from the results of the discrete sinus transformation depicted to the right of the respective eigenvectors. The form of eigenvalues and eigenvectors leads us to the conclusion that our one-dimensional sparse grid works as a low-pass filter.

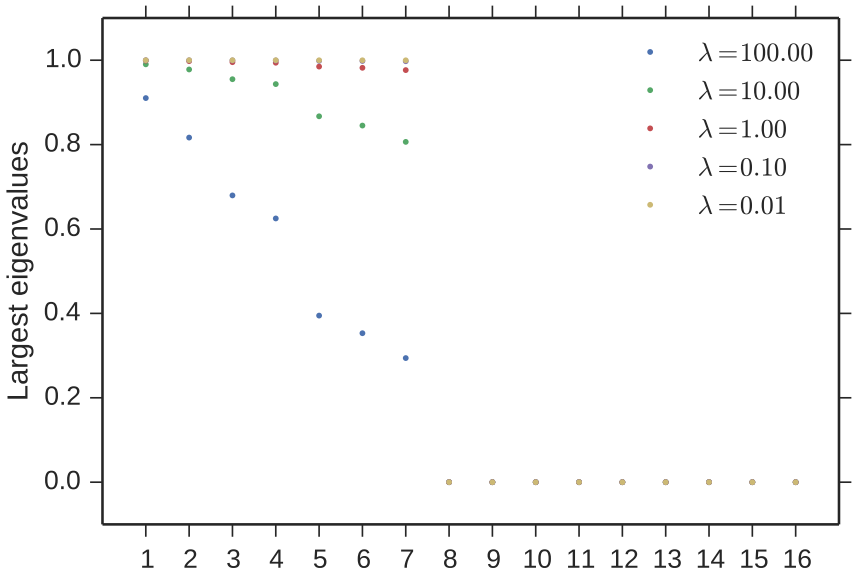


Figure 2.14.: Eigenvalues for one-dimensional sparse grid with different regularisation parameters λ. As regularisation matrix we used identity matrix. The nonzero eigenvalues are very close to 1 even for relatively high λ of 0.1 and 0.01.

As illustrated in Figures 2.16 – 2.18, for higher dimensions the eigenvectors correspond to the mixed frequencies

$$a_1 \cdot \sin(mx_1\pi)\sin(nx_2\pi) + a_2 \cdot \sin(nx_1\pi)\sin(mx_2\pi)$$

of the hyperbolic cross (see (Hallatschek, 1992) for the relationship to sparse grids). Despite the mixed frequencies, the eigenspace of the two-dimensional sparse grid is the same as if those were the pure frequencies and hence it is also a low-pass filter.

Consider a particular case of $\mathbf{D} = \mathbf{I}$ as we have in (2.31). Let $\mathbf{\Phi} = \mathbf{U\Sigma V}^T$ the singular value decomposition of the feature matrix.

Following (Hastie et al., 2011), we consider the prediction

$$\begin{aligned} \mathbf{f} = \mathbf{\Phi}\hat{\mathbf{w}} &= \mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{\Phi}^T\mathbf{y} \\ &= \mathbf{U\Sigma}(\mathbf{\Sigma}^2 + \lambda\mathbf{I})^{-1}\mathbf{\Sigma U}^T\mathbf{y} \\ &= \sum_{j=1}^{p} \mathbf{u}_j \frac{\sigma_j^2}{\sigma_j^2 + \lambda}\mathbf{u}_j^T\mathbf{y}, \end{aligned} \tag{2.92}$$

where the $\mathbf{u}_j$ are the columns of $\mathbf{U}$. Note that since $\lambda \geq 0$, we have $\sigma_j^2/(\sigma_j^2+\lambda) \leq 1$ as in Prop. 2.6. Ridge regression computes the coordinates of $\mathbf{y}$ with respect to the orthonormal basis $\mathbf{U}$. It then shrinks these coordinates by the factors $\sigma_j^2/(\sigma_j^2 + \lambda)$. This means that a greater amount of shrinkage is applied to

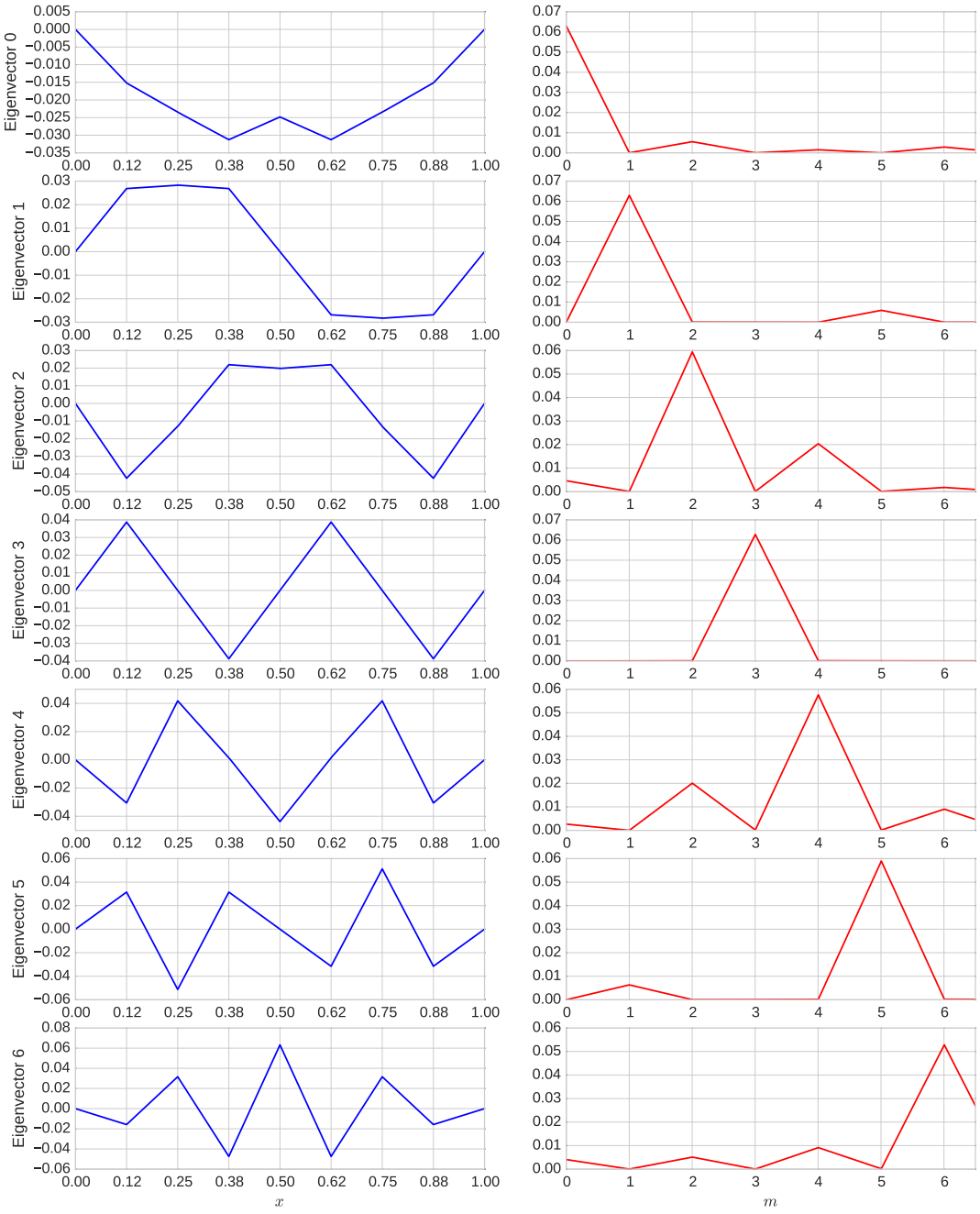


Figure 2.15.: Eigenvectors of a one-dimensional sparse grid smoother. One-dimensional eigenvectors correspond to the different sinus frequencies.

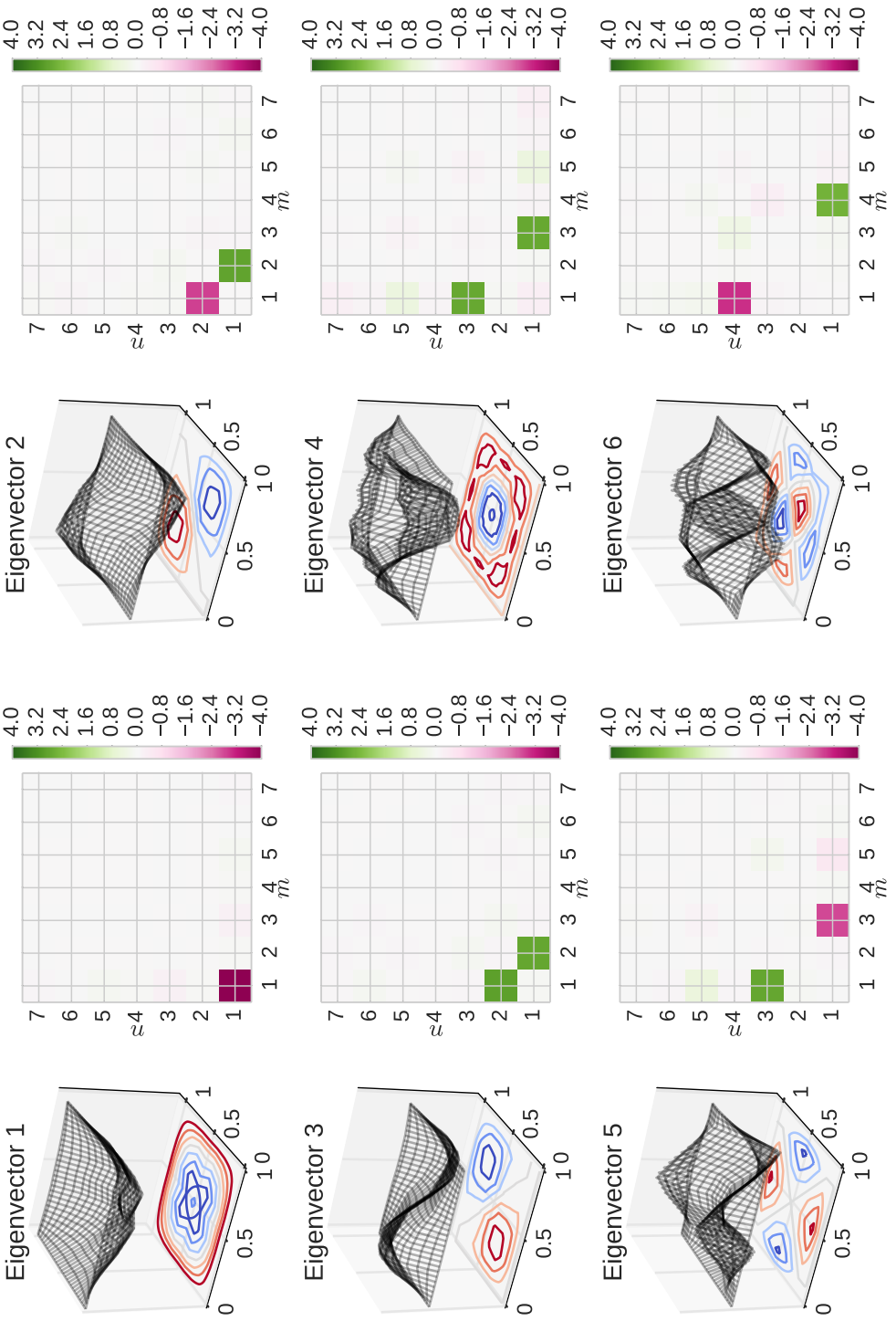


Figure 2.16.: Eigenvalues and eigenvectors 1 through 6 of a two-dimensional sparse grid smoother. The eigenvectors correspond to the mixed frequencies $a_1 \cdot \sin(mx_1\pi)\sin(nx_2\pi) + a_2 \cdot \sin(nx_1\pi)\sin(mx_2\pi)$ of the hyperbolic cross.

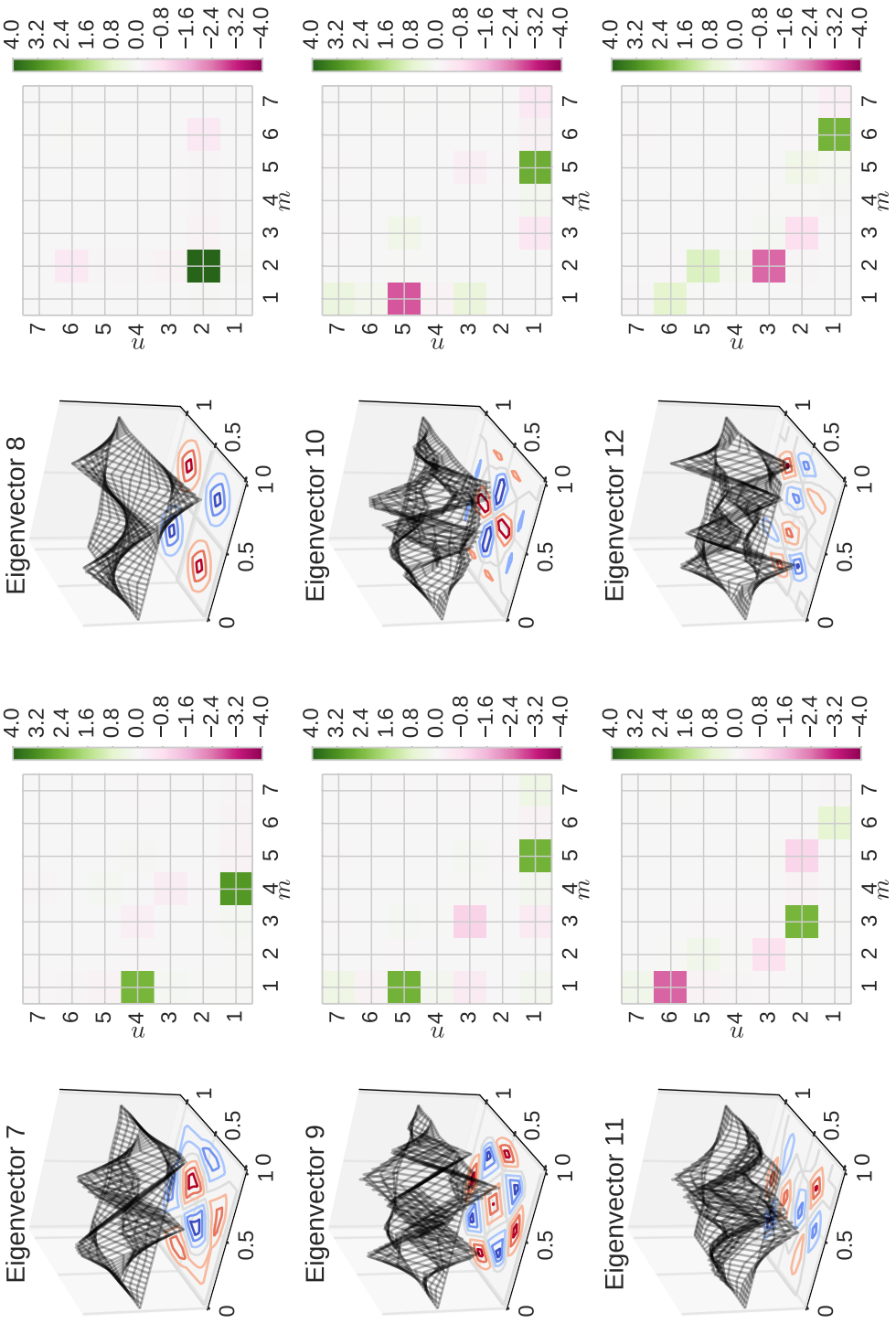


Figure 2.17.: Eigenvalues and eigenvectors 7 through 12 of the two-dimensional sparse grid smoother. The eigenvectors correspond to the mixed frequencies $a_1 \cdot \sin(mx_1\pi)\sin(nx_2\pi) + a_2 \cdot \sin(nx_1\pi)\sin(mx_2\pi)$ of the hyperbolic cross.

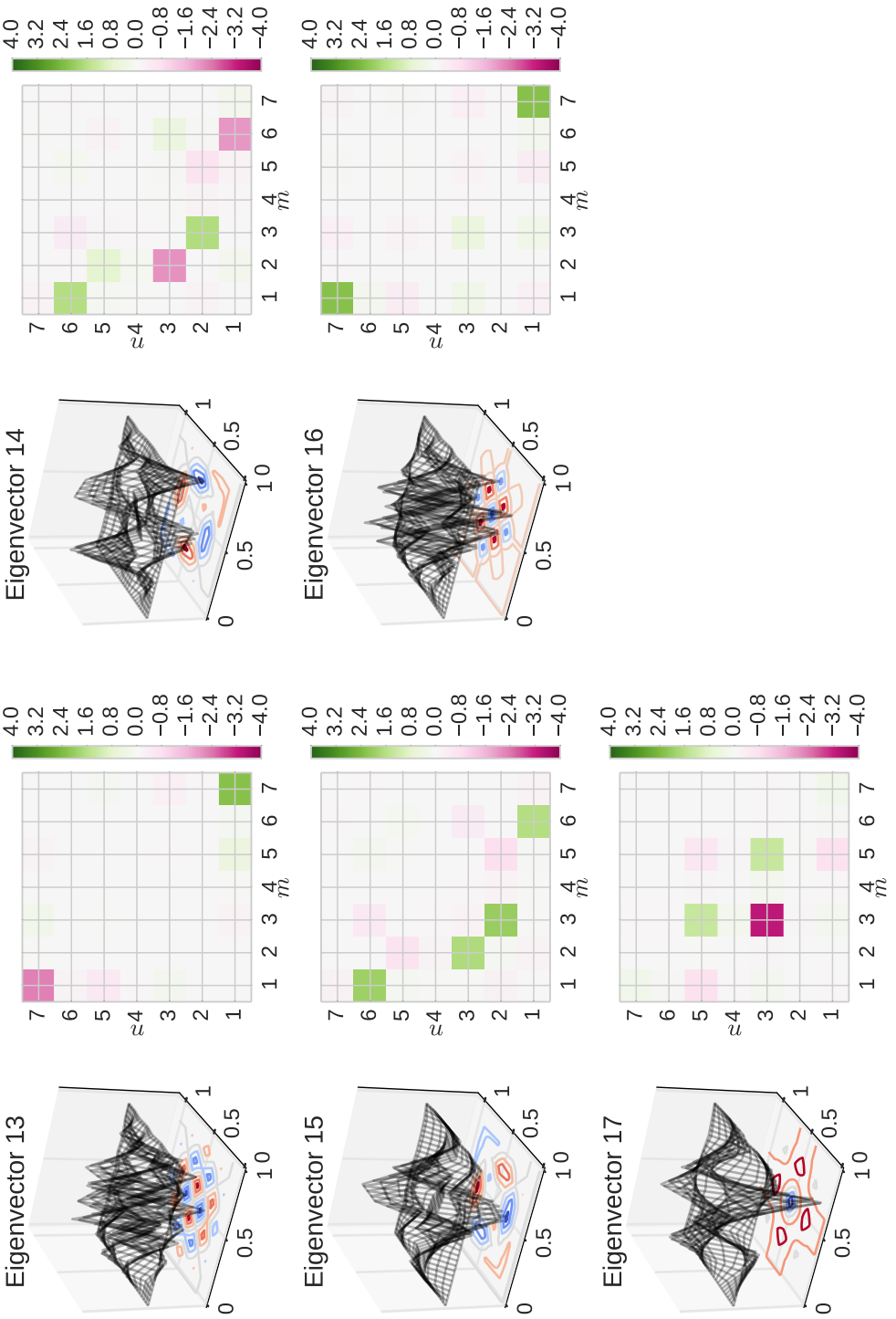


Figure 2.18.: Eigenvalues and eigenvectors 13 through 17 of the two-dimensional sparse grid smoother. The eigenvectors correspond to the mixed frequencies $a_1 \cdot \sin(mx_1\pi)\sin(nx_2\pi) + a_2 \cdot \sin(nx_1\pi)\sin(mx_2\pi)$ of the hyperbolic cross.

the coordinates of basis vectors with smaller σ_j^2. The small singular values σ_j correspond to directions in the column space of $\mathbf{\Phi}$ having small variance.

We define a quantity called *effective degrees of freedom* of the ridge regression fit

$$\begin{aligned} \mathrm{df}(\lambda) &= \mathrm{tr}[\mathbf{S}_\lambda] = \mathrm{tr}[\mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{I})^{-1}\mathbf{\Phi}^T] \\ &= \sum_{j=1}^{M} \frac{\sigma_j^2}{\sigma_j^2 + \lambda}. \end{aligned} \tag{2.93}$$

This is a monotone decreasing function of λ. Usually in a linear regression fit with M variables, the degrees of freedom of the fit is M, the number of free parameters. Although all M coefficients in a sparse grid model will not be zero, they are fit in a restricted way controlled by λ. We have $\mathrm{df}(\lambda) = M$ when $\lambda = 0$ (no regularisation) and $\mathrm{df}(\lambda) \to 0$ as $\lambda \to \infty$.

Despite their numerous advantages as universal nonparametric models for supervised learning, adaptive sparse grids are more often used in computaional science and engineering or uncertainty quantification than in machine learning. Nonetheless, there are strong ties between sparse grids and other popular methods from statistics and machine learning, which provide new insights into sparse grids theory and motivate the creation of new algorithms and applications.

3. Exploiting Low-dimensional Manifolds

> *This is the Death whose particular sphere of operations is, well, not a sphere at all, but the Discworld, which is flat and rides on the back of four giant elephants who stand on the shell of the enormous star turtle Great A'Tuin, and which is bounded by a waterfall that cascades endlessly into space. Scientists have calculated that the chance of anything so patently absurd actually existing are millions to one. But magicians have calculated that million-to-one chances crop up nine times out of ten.*
>
> — Terry Pratchett, *Mort*

He who guards his *sparse grid points* and his *sample size*, guards his *prediction model* from trouble.[1] Sparse grids allow to postpone the onset of the curse of dimensionality, typical for other grid-based nonparametric models. However, as the dimensionality of problems grows, this advantage becomes insufficient. Besides exploding computational costs we face the problem of sample size: the number of required training samples also grows exponentially with the dimensionality. This, in turn, further increases the computational costs.

Given certain regularity assumptions, Stone (1982) showed that the optimal rate of convergence[2] for nonparametric regression grows as $\mathcal{O}(N^{-p/(2p+D)})$, where N is the sample size, D is the dimensionality of the input variables, and the regression function is assumed to be p times differentiable.

Consider the following example: we want to approximate a regression function with $p = 2$ from a dataset with $D = 5$ and $N = 1\,000$, which results in the optimal convergence rate of approximately 0.22. Suppose that D is now

[1] Interpretation of *Proverbs 21:23 NASB* after four years in the PhD program.

[2] The optimal rate of convergence defined by Stone (1982) is unusual for a computer scientist and a little complicated. We provide it here for completeness, however, the reader is not required to understand the definition in order to follow the main exposition of the chapter. Stone defines the optimal rate of convergence as a sequence $\{b_N\}$ indexed by the sample size. Let Θ be a collection of functions, $\theta \in \Theta$ the unknown regression function, and $\hat{T}_N$ the estimator of θ using N samples. The sequence $\{b_N\}$ is called *lower rate of convergence*, if there is a constant $c > 0$ such that $\lim_N \inf_{\hat{T}_N} \sup_\Theta p(\|\hat{T}_N - \theta\| \geq cb_N) = 1$, where $\inf_{\hat{T}_N}$ is over all possible estimators. If there is a sequence of estimators $\{\hat{T}_N\}$ and $c > 0$ such that $lim_N \sup_\Theta p(\|\hat{T}_N - \theta\| \geq cb_N) = 0$ then $\{b_N\}$ is called *achievable rate of convergence*. The sequence if called an *optimal rate of convergence* if it is both lower and achievable.

increased to 15. Then, in order obtain the same optimal convergence rate, we need to increase the number of samples to roughly 2 000 000!

Thus high-dimensional problems require tailored solutions. In particular, if the data points lie on a low-dimensional manifold instead of filling the complete high-dimensional space, we can exploit this property to produce simpler models with the same prediction qualities at the fraction of computational costs. Even if the existence of a low-dimensional manifold is not known a priori, it may be a necessary assumption, which makes high-dimensional supervised learning problems computationally feasible.

We begin with a review of the literature related to manifold learning and sparse grids. We explain how the manifold identification can be intergrated into the supervised learning with adaptive sparse grids if it is formulated as a constrained optimisation problem. This problem can be approached either in its original constrained non-convex formulation using specific heuristics, as described in Sec. 3.2, or in a relaxed unconstrained convex formulation, as described in Sec. 3.3. The latter class of problems can be efficiently solved using the specialised algorithms presented in Sec. 3.4. Finally, Section 3.5 concludes this chapter with an empirical study of the presented methods for adaptive sparse grid learning on a series of synthetic and real-world problems.

3.1. Learning and Manifolds

Traditionally, supervised learning of problems with low-dimensional manifolds is a two-stage process: first, estimate a low-dimensional manifold, then use standard regression or classification algorithms on the transformed data. A large body of research in the machine learning community is focused on identification and estimation of latent manifolds in data as well as on nonlinear dimensionality reduction (see (Burges, 2009; Lee & Verleysen, 2007) and references therein). For example, sparse grids were used for dimensionality reduction and manifold learning with generative topographic mapping (Griebel & Hullmann, 2014a, 2014b), principal manifold learning (Feuersänger & Griebel, 2009; Bohn et al., 2016), and Laplacian eigenmaps (Peherstorfer et al., 2011).

However, identifying a manifold is far from being trivial and by itself can pose a significant computational burden. For instance, the principal manifold learning method described by Feuersänger and Griebel (2009) by itself requires the solution of a large number of convex and non-convex regression problems.

An alternative approach is to perform regression and classification tasks directly adapting the model to the manifold. Let us illustrate this principle in connection with sparse grid classification. Figure 3.1 depicts a rectangular decision boundary in the hierarchical subspaces of a sparse grid. Inside the rectangular the data points have one class and outside they have another. We can prune the basis functions that support only the points of the same class (depicted white) without loss of accuracy. Hence, adapting to a manifold in a classification prob-

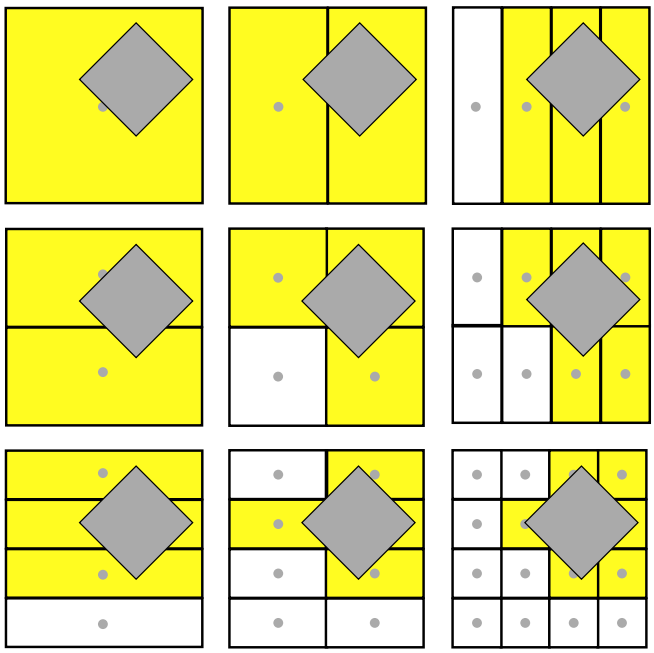


Figure 3.1.: Adaptive sparse grid for decision boundary identification. In order to adapt the model to the manifold, we need to identify only those basis functions which intersect the decision boundaries in every hierarchical subspace.

lem boils down to the identification of decision boundaries that differentiate one class from another. As shown in Fig. 3.1, in every hierarchical subspace we need to identify only those basis functions which intersect the decision boundaries.

This principle is not unique to sparse grids. Scott and Nowak (2005) showed that the decision trees with dyadic partitioning can focus on lower-dimensional manifolds, adapting to the conditions around decision boundaries and eliminating irrelevant features. The authors suggested that their bounding techniques can be applied to other families of trees using the Vapnik-Chervonenkis theory. Kpotufe (2009) and Kpotufe and Dasgupta (2012) used random projections to partition the ambient space and tree-based regressors to construct estimators on these partitions. Adaptive sparse grid models for classification relate to the dyadic decision trees and, intuitively, should inherit some of their properties. However, to the best of our knowledge this inheritance has not been shown formally.

Binev et al. (2005) and Binev et al. (2007) studied partition-based piecewise-constant and piecewise-linear estimators, establishing a general framework that includes the case of low-dimensional manifolds. The optimal convergence in their framework does not depend directly on the ambient dimension, but rather on how well the true function can be approximate by its class of estimators. The adaptive dyadic partitioning and the refinement of the tree-like structures

described by the authors resembles the principles behind the adaptive sparse grids.

More recently, Kpotufe and Garg (2013) presented a kernel regression procedure that adapts to the local smoothness and intrinsic dimensionality of the problem. Y. Yang and Dunson (2016) analysed the regression with Gaussian Processes and used rescaling to adapt the models to a low-dimensional manifold.

Assuming the view of sparse grid basis functions as feature maps (Sec. 2.4.1), we can see the adaptivity as a process of identifying the best features for a particular prediction task. Since the basis functions are locally constrained, the "best" features correspond to the "important" neighbourhoods in the ambient space as shown in Fig. 3.1.

To put it formally, we extend the minimisation of the empirical risk functional (2.5) with a cardinality constraint, limiting the number of nonzero components by k:

$$\begin{aligned} &\min_{\mathbf{w}\in\mathbb{R}^M} \tfrac{1}{N}\sum_{i=1}^{N} L\left(f(\mathbf{x}_i;\mathbf{w}), y_i\right) \\ &\text{subject to } \|\mathbf{w}\|_0 \leq k. \end{aligned} \tag{3.1}$$

We define the ℓ_0 regularisation term[3] as $\|\mathbf{w}\|_0 := |\{j \mid w_j \neq 0\}|$ underlining the similarities to ℓ_1- and ℓ_2-norm.

In the context of regression, the problem (3.1) relates to the *best subset selection* method, which identifies k features that yield the lowest residual sum of squares. Furnival and Wilson (1974) suggested a branch-and-bounds algorithm (which they called "leaps and bounds") for an efficient search in the feature space. The appropriate choice of k involves a trade-off between bias and variance and hence relies on statistical techniques to identify the number of features that lead to the lowest expected prediction error (Hastie et al., 2011).

Despite its efficiency, the leaps-and-bounds method still becomes prohibitively expensive for more than 30 or 40 features, which is not much for a sparse grid in a high-dimensional space. Moreover, the direct application of the best subset selection method to choose the sparse grid basis functions is problematic. For instance, the best subset in Fig. 3.1 consists in the 6 basis functions of the subspace $W_{(3,3)}$ in the bottom right corner. But neglecting the large-scale functions of hierarchical ancestors can lead to overfitting. As we have seen in Sec. 2.4.5, the hierarchical ancestors are important to represent the lower frequencies, while the noise usually hides in the higher frequencies. Hence, we are required to look for another methods of feature selection that preferably can take the hierarchical structure of sparse grids into account.

The minimisation with cardinality constraints (3.1) is known to be an NP-hard problem (Natarajan, 1995) and hence is intractable for large solution

[3]The term $\|\cdot\|_0$ is not a proper norm.

spaces. The multitude of methods to overcome this obstacle can be assigned to one of two categories:

1. *Greedy algorithms* approximate the optimal solution by sequentially selecting the most promising features. This method is very popular in signal processing where it is known under the name "matching pursuit" (Tropp & Gilbert, 2007; Wang et al., 2009; Donoho et al., 2012). We discuss the greedy algorithms in Sec. 3.2

2. Alternatively, one can substitute the non-convex constraint in (3.1) by its convex relaxation or, in more general case, one can regularise the minimisation problem using *sparsity-inducing penalties*. While being known for some time, recently these techniques experienced a renaissance in the machine learning community and are currently a subject of active research (Tibshirani, 1996; Efron et al., 2004; Bach et al., 2011; Bach, 2013). We discuss the application of the sparsity-inducing penalties to sparse grid learning in Sections 3.3 and 3.4.

3.2. Algorithms for Greedy Optimisation

Exhaustive search in the feature space becomes prohibitively expensive even for comparably small grid index-sets of, say, 40 features. Greedy algorithms generate a global solution performing a sequence of locally-optimal decisions. Depending on the direction of this decision path, the solution can either grow, shrink, or alternate between those two operations. This results in forward, backward, and forward-backward algorithms for subset selection.

The solution space of a sparse grid model has to obey weak hierarchy: the induced sparse grid graph has to be connected and contain the "root" feature. Fortunately, this significantly limits the space of feasible solutions. Besides computational efficiency, greedy selection methods have the ability to "grow" a solution that always satisfies the property of weak hierarchy.

We start by examining the standard greedy selection methods: backward and forward selection. This provide us the necessary building blocks. We discuss why forward selection makes sense for sparse grid models and how we can use the properties of hierarchical basis functions to design an efficient greedy algorithm for sparse grids. These building blocks for backward and forward selection can be combined to the superior adaptive greedy algorithm described at the conclusion of this section.

3.2.1. Backward Greedy Selection

The *backward greedy selection* algorithm, illustrated in Alg. 5, trains the full model with all features first and then sequentially removes the features with the smallest impact on the loss function. The method can produce optimal

results, as shown by Hastie et al. (2011), but it is computationally expensive. In addition, if the full model has more features than training examples then it will run into overfitting. In the context of the dimension-adaptive sparse grids, backward greedy selection is used to identify the important sparse grids ANOVA components (Bohn & Griebel, 2013). We come back to this topic in Sec. 4.2.1.

Algorithm 5: Backward Greedy Selection.

input : Set of all possible grid indices $\overline{G}$ and target vector $\mathbf{y} \in \mathbb{R}^N$
output: $G^{(k)}$ and $\mathbf{w}^{(k)}$

Let $M := |\overline{G}|$ and $G^{(M)} = \overline{G}$
for $k = M, M-1, \ldots$ **do**
  Estimate $\mathbf{w}^k := \arg\min_{\mathbf{w} \in \mathbb{R}^{G^{(k)}}} J(\mathbf{w}; G^{(k)})$ as in (2.31)
  Estimate $s^{(k)} = \arg\min_{s \in G^{(k)}} \widetilde{J}(G^{(k)} \setminus \{s\})$
  Let $G^{(k-1)} = G^{(k)} \setminus \{s^{(k)}\}$

3.2.2. Forward Greedy Selection

The *forward greedy selection* algorithm, illustrated in Alg. 6, starts with root node $G^{(0)}$ and generates a sequence $G^{(0)} \subset G^{(1)} \subset \cdots \subset G^{(k)} \subset \cdots$ adding new basis functions to maximise the negative marginal gain:

$$G^{(k)} := G^{(k-1)} \cup \arg\max_{s \in A_G^{(k-1)}} -\widetilde{J}(s|G^{(k-1)}). \tag{3.2}$$

In statistics and signal processing this algorithm is widely used and referred to as boosting or matching pursuit. With some modifications, this is also the most commonly used method for adaptive sparse grids (Pflüger, 2010, 2012; Jakeman & Roberts, 2011; Hegland, 2003; Garcke, 2004; Gerstner & Griebel, 2003).

How efficient is the forward greedy selection algorithm? We can show that Alg. 6 produces a result that is no more than $(1-1/e)$ times worse than optimum. The key to this theoretical bound is the observation that the negative marginal gain for sparse grids has the property of diminishing returns as we will explain in the following.

Let $J^-(G)$ be the difference between the sum of squared target values and $\widetilde{J}(G)$:

$$J^-(G) := \sum_{i=1}^{N} y_i^2 - \widetilde{J}(G). \tag{3.3}$$

The function $J^-(G)$ is a submodular function.

Algorithm 6: Forward Greedy Selection.

input : $\mathbf{y} \in \mathbb{R}^N$ and $\epsilon > 0$
output: $G^{(k)}$ and $\mathbf{w}^k$

Let $G^{(0)} = \{((1,\dots,1),(1,\dots,1))\}$ contain the root node and $\mathbf{w}^{(0)} = (1/N\mathbf{1}^T\mathbf{y})$
for $k = 1, 2, \dots$ **do**
  Find $s^{(k)} := \arg\max_{s \in A_{G^{(k)}}} -\widetilde{J}(s|G^{(k-1)})$
  Let $G^{(k)} = G^{(k-1)} \cup \{s^{(k)}\}$
  Estimate $\mathbf{w}^k := \arg\min_{\mathbf{w} \in \mathbb{R}^{G^{(k)}}} J(\mathbf{w}; G^{(k)})$ as in (2.31)
  if $-\widetilde{J}(s|G^{(k-1)}) \leq \epsilon$ **then**
    break

Definition 3.1. *Let Γ be a set, $\mathcal{P}(\Gamma)$ its powerset, and $h : \mathcal{P}(\Gamma) \to \mathbb{R}$ a set function. Then h is said to be* ***submodular*** *if it satisfies the property of diminishing returns*

$$h(x|X) \geq h(x|Y) \tag{3.4}$$

for all sets $X \subseteq Y \subseteq \Gamma$ and all elements $x \in \Gamma \setminus Y$. The submodular function h is ***monotone non-decreasing*** *if $h(x|X) \geq 0$ for each element $x \in \Gamma \setminus X$ and each set $X \subseteq \Gamma$.*

Theorem 3.1. *The function J^- defined in* (3.3) *is a monotone non-decreasing submodular function. The submodular optimisation problem with cardinality constraint*

$$\max_{G \subset \Gamma, |G| \leq m} J^-(G) \tag{3.5}$$

is NP-hard. It can be solved with a greedy algorithm such that, if $\widehat{G}$ is the greedy solutions and G^* is the optimal solution, we have

$$\frac{J^-(\widehat{G})}{J^-(G^*)} \geq 1 - \left(\frac{m-1}{m}\right)^m \geq \left(1 - \frac{1}{e}\right) \approx 0.632. \tag{3.6}$$

This boundary is tight unless P=NP.

Proof. To simplify the exposition of the chapter, we discuss the proof in Appendix B.1. □

Albeit using the marginal gain for the forward greedy algorithm leads to near-optimal results, computation of the marginal gain can be expensive, since adding a new grid index to the index-set requires re-estimation of all sparse grid coefficients in the model. To make it more practical, the following lemma offers an inexpensive lower bound approximation.

Lemma 3.2. *The negative marginal gain of $\widetilde{J}$ can be bounded from below as*

$$-\widetilde{J}(s|G) = J^-(s|G) \geq \rho^{(0)}(s,G), \quad \text{with } \rho^{(0)}(s,G) = \frac{\left(\sum_{i=1}^N r_i \phi_s(\mathbf{x}_i)\right)^2}{\sum_{i=1}^N \phi_s^2(\mathbf{x}_i) + \lambda N}. \tag{3.7}$$

Proof. Let $\hat{\mathbf{w}}_G \in \mathbb{R}^G$ be the minimiser of (2.36), with $\hat{\mathbf{w}}_G = (\hat{w}_g)_{g\in G}$, and let $r_i := \sum_{g\in G} \hat{w}_g \phi_g(\mathbf{x}_i) - y_i$ be the residual at the point $(\mathbf{x}_i, y_i)$. Then with definitions (2.36), (3.3), and (2.37) the marginal gain function $J^-(s|G)$ satisfies

$$\begin{aligned}
J^-(s|G) =& -\min_{\mathbf{w}\in\mathbb{R}^{G\cup\{s\}}} \left(\sum_{i=1}^N \left(\sum_{g\in G} w_g \phi_g(\mathbf{x}_i) - y_i + w_s \phi_s(\mathbf{x}_i) \right)^2 + \lambda N \sum_{g\in G} w_g^2 + \lambda N w_s^2 \right) + \\
& \sum_{i=1}^N \left(\sum_{g\in G} \hat{w}_g \phi_g(\mathbf{x}_i) - y_i \right)^2 + \lambda N \sum_{g\in G} \hat{w}_g^2 \\
\geq& -\min_{w_s\in\mathbb{R}} \left(\sum_{i=1}^N \left(r_i + w_s \phi_s(\mathbf{x}_i)\right)^2 + \lambda N \sum_{g\in G} \hat{w}_g^2 + \lambda N w_s^2 \right) + \sum_{i=1}^N r_i^2 + \lambda N \sum_{g\in G} \hat{w}_g^2 \\
=& -\min_{w_s\in\mathbb{R}} \sum_{i=1}^N \left(2 w_s r_i \phi_s(\mathbf{x}_i) + w_s^2 \phi_s^2(\mathbf{x}_i) \right) + \lambda N w_s^2 =: \rho^{(0)}(s,G).
\end{aligned} \tag{3.8}$$

We can determine the minimising weight $\hat{w}_s$ of $\rho^{(0)}(s,G)$:

$$\begin{aligned}
2\sum_{i=1}^N \phi_s(\mathbf{x}_i)\left(r_i + \hat{w}_s \phi_s(\mathbf{x}_i)\right) + 2\lambda N \hat{w}_s &\stackrel{!}{=} 0, \\
\sum_{i=1}^N \hat{w}_s \phi_s^2(\mathbf{x}_i) + \lambda N \hat{w}_s &= -\sum_{i=1}^N r_i \phi_s(\mathbf{x}_i), \\
\hat{w}_s &= -\frac{\sum_{i=1}^N r_i \phi_s(\mathbf{x}_i)}{\sum_{i=1}^N \phi_s^2(\mathbf{x}_i) + \lambda N}.
\end{aligned} \tag{3.9}$$

We then evaluate $\rho^{(0)}(s,G)$ explicitly by substituting $w_s = \hat{w}_s$ in (3.8):

$$\begin{aligned}
\rho^{(0)}(s,G) &= -\min_{w_s\in\mathbb{R}} \sum_{i=1}^N \left(2 w_s r_i \phi_s(\mathbf{x}_i) + w_s^2 \phi_s^2(\mathbf{x}_i) \right) + \lambda N w_s^2 \\
&= -\left(2\hat{w}_s \sum_{i=1}^N r_i \phi_s(\mathbf{x}_i) + \hat{w}_s^2 \sum_{i=1}^N \left(\phi_s(\mathbf{x}_i)\right)^2 + \lambda N \hat{w}_s^2 \right) \\
&= 2\frac{\sum_{i=1}^N r_i \phi_s(\mathbf{x}_i)}{\sum_{i=1}^N \phi_s^2(\mathbf{x}_i) + \lambda N} \sum_{i=1}^N r_i \phi_s(\mathbf{x}_i) -
\end{aligned}$$

$$\begin{aligned}
&\frac{\left(\sum_{i=1}^N r_i\phi_s(\mathbf{x}_i)\right)^2}{\left(\sum_{i=1}^N \phi_s^2(\mathbf{x}_i)+\lambda N\right)^2}\sum_{i=1}^N\left(\phi_s(\mathbf{x}_i)+\lambda N\right)^2\\
&=2\frac{\left(\sum_{i=1}^N r_i\phi_s(\mathbf{x}_i)\right)^2}{\sum_{i=1}^N \phi_s^2(\mathbf{x}_i)+\lambda N}-\frac{\left(\sum_{i=1}^N r_i\phi_s(\mathbf{x}_i)\right)^2}{\sum_{i=1}^N \phi_s^2(\mathbf{x}_i)+\lambda N}\\
&=\frac{\left(\sum_{i=1}^N r_i\phi_s(\mathbf{x}_i)\right)^2}{\sum_{i=1}^N \phi_s^2(\mathbf{x}_i)+\lambda N}. \qquad (3.10)
\end{aligned}$$

This establishes Formula (3.7) and proves the lemma. □

Lemma 3.2 implies that instead of maximising the more expensive marginal gain of J^- with respect to the candidate indices s (as required in Line 2 of Alg. 6), we can maximise its lower bound $\rho^{(0)}(s,G)$. The computation of $\rho^{(0)}(s,G)$ is no more expensive than the computation of the refinement indicator (2.40). The resulting refinement procedure selects the grid index s with the highest indicator $\rho^{(0)}(s,G)$ among all indices in the candidate set.

In practice, it is advisable to add multiple new grid indices in every refinement step instead of one. However, it may be insufficient to choose the indices based on the size of the indicators. For example, taking two basis functions with a large support overlap may be less expedient than taking functions with disjoint support. Hence, selecting a grid index may reduce the refinement indicators (potential benefit) of the other candidate indices in the admissible set.

Based on the idea of lazy-greedy selection for submodular optimisation (Minoux, 1978; Leskovec et al., 2007; Wei et al., 2014) we present Alg. 7 for the selection of several basis functions for refinement at once. The algorithm makes use of a priority queue with refinement indicators ρ. At the step j of the refinement procedure the algorithm retrieves the largest grid index v from the queue and updates its refinement indicator $\rho^{(j)}(v,G)$. If the updated indicator is still greater than the upper bound of the following grid index u in the queue, v is accepted for refinement, otherwise v is put back into the queue and the procedure repeats.

What is an efficient way to update the refinement indicators? The following lemma gives us a formula.

Lemma 3.3. *Suppose that $I^{(j)}$ contains j grid indices already selected from the candidate set, and that at the step j Alg. 7 selected the grid index u. Then for every $v \in A \setminus I^{(j)}$ the refinement indicator is updated as*

$$\rho^{(j)}(v,G)=\left[\sqrt{\rho^{(j-1)}(v,G)}+\frac{\sum_{i=1}^N \hat{w}_u\phi_u(\mathbf{x}_i)\phi_v(\mathbf{x}_i)}{\sqrt{\sum_{i=1}^N \phi_v^2(\mathbf{x}_i)+\lambda N}}\right]^2. \qquad (3.11)$$

Algorithm 7: Fast Greedy Refinement.

input : Old grid $G^{(k)}$, admissible index set A, number of new indices n
output: New grid $G^{(k+1)}$

compute the error indicators $\rho^{(0)}(g, G^{(k)})$ for all candidates using (3.7)
store the grid indices in a priority queue $\mathcal{Q}$ such that $\rho^{(0)}(\mathcal{Q}[0], G^{(k)}) \geq \rho^{(0)}(\mathcal{Q}[1], G^{(k)}) \geq \ldots \geq \rho^{(0)}(\mathcal{Q}[|A|-1], G^{(k)})$
select the index with largest error indicator $I^{(1)} \leftarrow \{\mathcal{Q}.\text{pop}()\}$
$j \leftarrow 2$
repeat
    select the largest element $v \leftarrow \mathcal{Q}.\text{pop}()$
    update the indicator refinement indicator of v using (3.14)
    if *$\rho^{(j)}(v, G^{(k)})$ is greater or equal to the refinement indicator stored for the next index in the queue* **then**
        add the index to the sparse grid index-set $I^{(j)} \leftarrow I^{(j-1)} \cup \{v\}$
        increment $j \leftarrow j + 1$
    else
        sort v back into the queue $\mathcal{Q}.\text{push}(v)$
until $j = n$
$G^{(k+1)} \leftarrow G^{(k)} \cup I^{(j)}$

Proof. If every $g \in I^{(j)}$ has the optimal coefficient $\hat{w}_g$ computed as in (3.9), the error terms become

$$r_i^{(j)} = r_i + \sum_{g \in I^{(j)}} \hat{w}_g \phi_g(\mathbf{x}_i) = r_i^{(j-1)} + \hat{w}_u \phi_u(\mathbf{x}_i), \quad \text{such that } u \in I^{(j)} \setminus I^{(j-1)}.$$

For every $v \in A \setminus I^{(j)}$ the refinement indicator is updated as

$$\begin{aligned}
\rho^{(j)}(v, G) &= \frac{\left(\sum_{i=1}^N r_i^{(j)} \phi_s(\mathbf{x}_i)\right)^2}{\sum_{i=1}^N \phi_s^2(\mathbf{x}_i) + \lambda N} \\
&= \left[\frac{\sum_{i=1}^N r_i \phi_v(\mathbf{x}_i)}{\sqrt{\sum_{i=1}^N \phi_v^2(\mathbf{x}_i) + \lambda N}} + \frac{\sum_{i=1}^N \phi_v(\mathbf{x}_i) \sum_{g \in I^{(j)}} \hat{w}_g \phi_g(\mathbf{x}_i)}{\sqrt{\sum_{i=1}^N \phi_v^2(\mathbf{x}_i) + \lambda N}} \right]^2 \qquad (3.12) \\
&= \left[\frac{\sum_{i=1}^N r_i^{(j-1)} \phi_v(\mathbf{x}_i)}{\sqrt{\sum_{i=1}^N \phi_v^2(\mathbf{x}_i) + \lambda N}} + \frac{\sum_{i=1}^N \hat{w}_u \phi_u(\mathbf{x}_i) \phi_v(\mathbf{x}_i)}{\sqrt{\sum_{i=1}^N \phi_v^2(\mathbf{x}_i) + \lambda N}} \right]^2, \text{ s.t. } u \in I^{(j)} \setminus I^{(j-1)}. \qquad (3.13)
\end{aligned}$$

Equation (3.12) implies that the updated refinement indicator can be obtained from the original one using the formula

$$\rho^{(j)}(v,G) = \left[\sqrt{\rho^{(0)}(v,G)} + \frac{\sum_{i=1}^{N} \phi_v(\mathbf{x}_i) \sum_{g \in I^{(j)}} \hat{w}_g \phi_g(\mathbf{x}_i)}{\sqrt{\sum_{i=1}^{N} \phi_v^2(\mathbf{x}_i) + \lambda N}}\right]^2 . \tag{3.14}$$

Similarly, Equation (3.13) implies that it can also be obtained from a previously updated indicator as

$$\rho^{(j)}(v,G) = \left[\sqrt{\rho^{(j-1)}(v,G)} + \frac{\sum_{i=1}^{N} \hat{w}_u \phi_u(\mathbf{x}_i) \phi_v(\mathbf{x}_i)}{\sqrt{\sum_{i=1}^{N} \phi_v^2(\mathbf{x}_i) + \lambda N}}\right]^2 .$$

□

What happens if more than one index is taken in the greedy procedure? While the theoretical guarantees for this case are missing, the idea was successfully applied in a number of algorithms for large sparse inverse problems using matching pursuit methods (Kambadur & Lozano, 2013; Blumensath & Davies, 2009).

By posing sparse grid refinement as a discrete optimisation problem we gained an access to the theory of submodular functions. This allowed us to derive a refinement indicators that, combined with some specific properties of submodular functions, guarantee that our choice of grid points to refine is not too far from the optimal. We derived an algorithm that allowed us to add new basis functions without recomputing all refinement indicators, which should significantly accelerate the process. If this is still not sufficient, we can add more than one new basis function at once. While the optimality bounds of the latter strategy are unknown, the procedure still works well in practice.

3.2.3. Adaptive Forward-Backward Greedy Selection

The forward selection algorithm can be further improved significantly if the features are not nearly orthogonal. Since the algorithm cannot correct the mistakes made in earlier steps, it has a significant drawback if the features are correlated. Unfortunately, this is the case for the features that correspond to the hierarchically related sparse grid basis functions.

Figure 3.2 illustrates an example where simple forward selection fails to obtain the sparsest solution: The target vector $\mathbf{y}$ can be perfectly represented as an affine combination[4] of the feature vectors $\boldsymbol{\phi}_2$ and $\boldsymbol{\phi}_3$. However, since $\boldsymbol{\phi}_1$ is closer

[4]We restrict the example to affine combinations instead of linear combinations. This means that, although the combination coefficients may be positive or negative, its sum is always one. Otherwise, in a two-dimensional plane the vector $\mathbf{y}$ could be represented as a linear combination of any two linear independent features in the example.

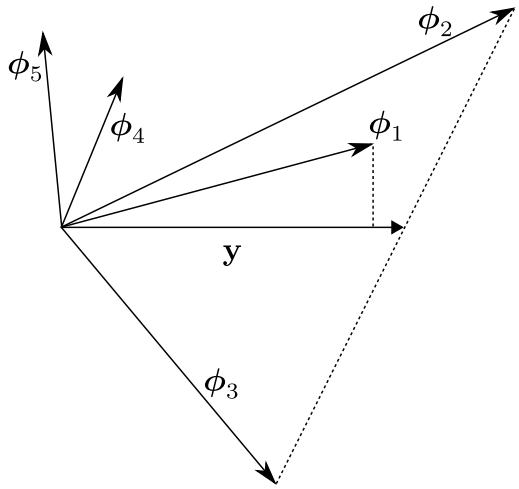


Figure 3.2.: Failure of the forward greedy selection algorithm. The target vector $\mathbf{y}$ can be represented using the feature vectors $\boldsymbol{\phi}_2$ and $\boldsymbol{\phi}_3$. Nevertheless, the superfluous vector $\boldsymbol{\phi}_1$ is closer to $\mathbf{y}$ and is selected first, followed by $\boldsymbol{\phi}_2$ and $\boldsymbol{\phi}_3$. Adapted from T. Zhang (2011).

to $\mathbf{y}$, it is selected first, followed by $\boldsymbol{\phi}_2$ and $\boldsymbol{\phi}_3$. Now, since $\boldsymbol{\phi}_2$ and $\boldsymbol{\phi}_3$ would be sufficient, we would like to remove $\boldsymbol{\phi}_1$ in order to obtain the optimal solution, which is not allowed by the forward selection algorithm.

A natural extension of the forward and backward greedy selection methods is the combination of these two. Feuersänger (2010, Sec. 2.2.1) presented an adaptive sparse grid algorithm that can either refine or compress the sparse grid for function interpolation. However, his backward greedy selection step is sensitive to the choice threshold parameter ϵ', which controls the features to delete. Choose it too small and the algorithm fails to delete spurious features, choose it too large and important features will be lost.

T. Zhang (2009) introduced an adaptive forward-backward greedy selection algorithm for the least squares loss function. This work was extended by Liu et al. (2014) to accommodate general convex smooth functions. We adjust the method for sparse grid regression as shown in Alg. 8. It represents a combination of the Forward Greedy Algorithm 6 in Lines 7–11 and of the Backward Greed Algorithm 5 in Lines 14–21. The algorithm adaptively decides when to take the backward step ensuring that the gain of the previous forward steps is not voided. In fact, a backward step is carried out only if the squared error increase d^- is no more than a half of the squared error decrease d^+ from the previous forward step. This means that after t iterations the squared error is reduced by at least $t\epsilon/2$ independent of the number of the backward steps.

T. Zhang (2011) showed that the adaptive forward-backward selection algorithm can efficiently solve regression problems given the restricted isometry condition. This condition states that the smallest eigenvalue of the $k \times k$ principal submatrices of the $M \times M$ design matrix $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$ as used in (2.33) can be

Algorithm 8: Adaptive Forward-Backward Greedy Selection, adapted from T. Zhang (2011)

input : $\mathbf{y} \in \mathbb{R}^N$ and $\epsilon > 0$
output: $G^{(k)}$ and $\mathbf{w}^{(k)}$

Let $\eta = 0.5$ `// it can also be any number in` $(0, 1)$

Let $G^{(0)} = \emptyset$ and $\mathbf{w}^{(0)} = (1/N\mathbf{1}^T\mathbf{y})$
Let $k = 0$
while *true* **do**
    `// forward greedy selection step`
    Find $s^{(k)} := \arg\max_{s \in A_{G^{(k)}}} -\widetilde{J}(s|G^{(k-1)})$
    Let $\delta^{(k+1)} = -\widetilde{J}(s|G^{(k-1)})$
    Let $G^{(k)} = G^{(k-1)} \cup \{s^{(k)}\}$
    if $\delta^{(k+1)} \leq \epsilon$ **then**
        break
    Let $k = k + 1$

    `// backward greedy selection step`
    while *true* **do**
        Estimate $s^{(k)} = \arg\min_{s \in G^{((k)}} \widetilde{J}(G^{(k)} \setminus \{s\})$
        Let $d^- = -\widetilde{J}(s^{(k)}|G^{(k)} \setminus \{s^{(k)}\})$
        Let $d^+ = \delta^{(k)}$
        if $d^- > \eta \cdot d^+$ **then**
            break
        Let $G^{(k-1)} = G^{(k)} \setminus \{s^{(k)}\}$
        Let $k = k - 1$

bounded away from zero. This assumption is satisfied for the sparse grid models due to the linear independence of the hierarchical basis as long as there are training points on the support of every basis function (no zero feature vectors).

For the optimal performance, one should decrease the refinement threshold ϵ as $\mathcal{O}(\sigma^2 \ln D/N)$ with σ^2 being the noise variance. This means that, at least in theory, adaptive forward-backward greedy selection needs more iterations if N grows.

The adaptive forward-backward algorithm is natural extensions of the adaptive sparse grid learning algorithms. By alternating between the forward and the backward step—between refining and coarsening of a sparse grid model—it adds only the most promising and removes obsolete basis functions. However, this is not the only way to increase parsimony. In the next section we consider a number of convex penalties to facilitate sparsity.

3.3. Convex Problem Formulations with Sparsity-Inducing Penalties

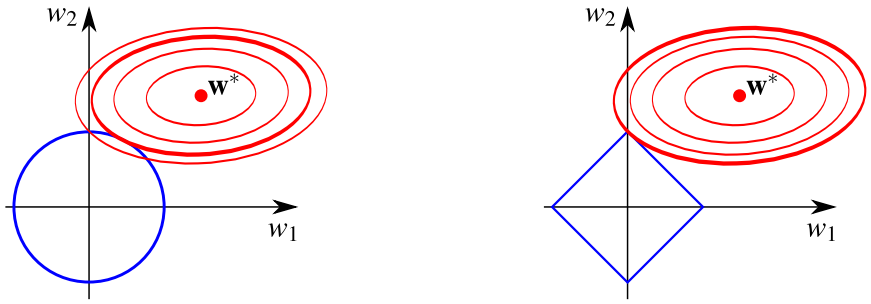
The subset selection methods described in the previous section produce interpretable models by retaining a subset of predicting features and discarding the rest. However, these discrete optimisation methods often exhibit high variance of the results. Due to this high variance, the performance of the resulting models can be far from optimal. Optimisation with *sparsity-inducing penalties*, sometimes called *shrinkage methods*, offers a continuous alternative to the discrete selection methods that does not suffer from high variability.

The idea is to transform the optimisation problem (3.1) with a *discrete* ℓ_0 constraint into an optimisation problem

$$\min_{\mathbf{w}\in\mathbb{R}^M} \tfrac{1}{N} \sum_{i=1}^{N} L(f(\mathbf{x}_i; \mathbf{w}), y_i) + \lambda\Omega(\mathbf{w}) \tag{3.15}$$

with a *continuous* penalty Ω (Bertsekas & Rheinboldt, 1982). The ℓ_1-norm of a vector in a unit cube is the tightest convex hull over the level set of ℓ_0 and hence is well suited as a penalty. Under weak assumptions on L and Ω from Lagrangian multiplier theory, one can show that the solutions for these optimisation problems for certain $\lambda \geq 0$ and $k \geq 0$ are equivalent (see Borwein & Lewis, 2006, Sec. 4.3). Hence, we can move from the realm of constrained discrete optimisation into unconstrained convex optimisation.

Although the sets of solutions where λ and k vary—the *regularisation paths*—are equivalent, the correspondence between particular λ and k is usually unknown. Hence, we cannot select the regularisation parameter λ that corresponds to a particular number of features k. In practice, however, it is rarely a problem as the hyperparameters λ or k are estimated in a validation procedure.



(a) ℓ_2^2 penalty norm (b) ℓ_1 penalty norm

Figure 3.3.: Parameter estimation for lasso and ridge regression. The ball of the penalisation term $\mathcal{B}$ is depicted in blue and the level-set of L is depicted in red. At the optimal point the level-set is tangential to $\mathcal{B}$.

Besides ℓ_1-norm there are a number of other regularisation functions that aim to increase the sparsity and sometimes to enforce a special structure of the minimiser $\mathbf{w}$. We discuss the functions applicable to sparse grid learning in the following. This list, however, is by no means exhaustive. For a comprehensive exposition we refer to the monography by Bach et al. (2011), which we build upon in this section.

3.3.1. ℓ_q-Norms

We consider the family $\Omega_q(\mathbf{w}) := \|\mathbf{w}\|_q^q$ of regularisers by an ℓ_q^q-norm. For $q = 0$ we have the best subset selection problem (3.1), for $q = 2$ the ridge regression problem (2.31).

Consider the *level-set* of the loss function L—the set of parameters $\mathbf{w}$ where the value of L is the same. If $\widehat{\mathbf{w}}$ is the optimal solution of (3.15) then the gradient of L evaluated at $\widehat{\mathbf{w}}$ belongs to the normal cone of $\mathcal{B} = \{\mathbf{w} \in \mathbb{R}^M \mid \Omega(\mathbf{w}) \leq k\}$ (Borwein & Lewis, 2006). For instance, if k is sufficiently small and the constraint is active then the level-set is tangential to $\mathcal{B}$.

This observation relates the properties of the solution $\widehat{\mathbf{w}}$ to the geometry of the ball $\mathcal{B}$. We illustrate this relationship in Fig. 3.3. If Ω is the ℓ_2^2-norm then $\mathcal{B}$ is "round" as depicted in Fig. 3.3a and does not favour any specific direction in space. If Ω is the ℓ_1-norm then $\mathcal{B}$ has singular points along the axis in $\mathbb{R}^M$ as depicted in Fig. 3.3b. The sparse solutions appear when the level-set of L is tangential at one of these singular points.

Figure 3.4 illustrates the level-sets of $\Omega_q(\mathbf{w})$ for different values of q. For $q \geq 1$ these level-sets are convex. One can see that the level-set for $q = 1$ is the convex superset of the level-set for $q = 0$. In fact, one can prove that it is the tightest convex superset (e.g., Bach, 2013). The minimisation problem (3.15) with the least squares loss function and ℓ_1 regularising is well known under the

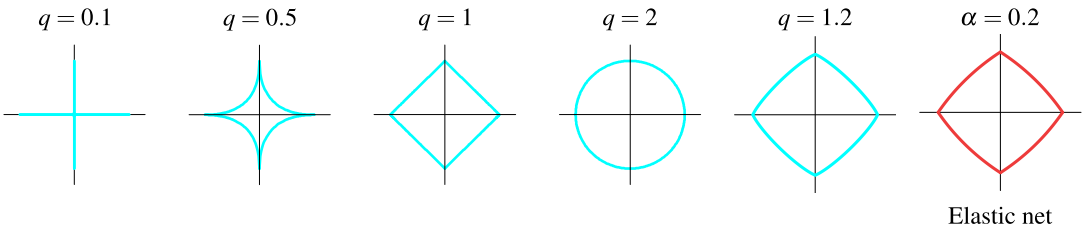


Figure 3.4.: The ball $\mathcal{B}$ of $\Omega_q(\mathbf{w})$ for different values of q. In every graph the shape illustrates the set of coordinates with the same value $\Omega_q(\mathbf{w})$. Adapted from (Hastie et al., 2011).

name *lasso* in the statistical literature (Tibshirani, 1996) and under the name basis pursuit in signal processing (Chen et al., 1998).

Lasso has the Lagrangian form

$$\min_{\mathbf{w}} \tfrac{1}{N}(y_i - f(\mathbf{x}_i; \mathbf{w}))^2 + \lambda \sum_{j=1}^{M} |w_j|. \tag{3.16}$$

The quadratic programming problem (3.16) is convex but non-differentiable. Hence it requires specialised optimisation methods. In recent years we observed the emergence of many efficient methods for optimisation with ℓ_1-penalties (e.g., M. R. Osborne, 2000; Efron et al., 2004; Nesterov, 2007; Beck & Teboulle, 2009; G.-X. Yuan, Chang, Hsieh, & Lin, 2010).

3.3.2. ℓ_1/ℓ_q-Norms

Zou and Hastie (2005) suggested to combine the ℓ_1 and ℓ_2^2 regularisers together as

$$\Omega_\alpha(\mathbf{w}) = \alpha\|\mathbf{w}\|_2^2 + (1-\alpha)\|\mathbf{w}\|_1, \tag{3.17}$$

calling the result an *elastic-net*. It supposes to select features as ℓ_1-norm and shrink the coefficients of correlated predictors as ℓ_2^2-norm, at the same time being computationally advantageous. The last two level-sets in Fig. 3.4 show the Ω_q regulariser with $q = 1.2$ and the Ω_α regulariser with $\alpha = 0.2$. Although they look similar, elastic-net has non-differentiable edges while Ω_q has not.

Another way to combine different ℓ_q norms in a penalty arises when the components of $\mathbf{w}$ can be naturally partitioned into groups such that all components in a group are added or removed simultaneously. Such ℓ_1 group norms can improve the prediction performance and the interpretability of the learned models (Turlach et al., 2005; M. Yuan & Lin, 2006; Roth & Fischer, 2008; Huang & Zhang, 2010). Think of all coefficients of basis functions that belong to the same hierarchical space as a group. Such ℓ_1 group norms can be useful to decide if the complete group of coefficients—the hierarchical subspace—should be added or removed.

In its general form an ℓ_1/ℓ_q-norm penalty has the form

$$\Omega(\mathbf{w}) := \sum_{g\in\mathcal{P}} \gamma_g \|\mathbf{w}_g\|_q = \sum_{g\in\mathcal{P}} \gamma_g \left(\sum_{j\in g} |w_j|^q\right)^{1/q}, \tag{3.18}$$

where $\mathcal{P}$ is a partition of $\{1,\dots,M\}$, $\gamma_g \geq 0$ for the subset $g \in \mathcal{P}$ are some weights, and $\mathbf{w}_g \in \mathbb{R}^g$ is the vector with components from $\mathbf{w}$ indexed by g. Note that Ω defined in (3.18) behaves as ℓ_1-norm in the values $(\|\mathbf{w}_g\|_q)_{g\in\mathcal{P}}$ and hence causes group sparsity.

The most commonly used regularisation term of this class is the ℓ_1/ℓ_2-norm of the form

$$\Omega(\mathbf{w}) := \sum_{g\in\mathcal{P}} \gamma_g \|\mathbf{w}_g\|_2. \tag{3.19}$$

Combined with the square loss, it is referred to as *group lasso* (Turlach et al., 2005; M. Yuan & Lin, 2006).

3.3.3. Penalties for Overlapping Groups

Besides parsimony of the solution, we want to preserve the hierarchical dependency between the selected sparse grid basis functions. This dependency can be encoded into groups of features that are allowed to overlap. These *overlapping groups* can be handled in different ways as suggested by Jacob et al. (2009), Zhao et al. (2009), and Lim and Hastie (2015).

Zhao et al. (2009) introduced the composite absolute penalty (CAP) to handle grouped and hierarchical feature selection using *explicit* formulation of overlapping groups. This formulation goes as follows: Let I_1 and I_2 be two sets of grid indices such that I_1 should be added into the model before I_2. Then we define the groups as $g_1 = I_1$ and $g_2 = I_1 \cup I_2$ and use the CAP

$$\Omega(\mathbf{w}) := \|\mathbf{w}_{g_1}\|_{q_1} + \|\mathbf{w}_{g_2}\|_{q_2}$$

with the values of q_1 and q_2 greater than 1. In the simplest case, if the component w_2 has to appear in the model only after the component w_1, the CAP is $\|(w_1, w_2)\|_2 + \|w_2\|_2$.

This idea can be generalised to an arbitrary hierarchical dependency encoded in a directed acyclic graph: the nodes correspond to groups of features[5] g_n and its descendants are the groups of features that should be added to the model after g_n. Then the CAP has the form

$$\Omega(\mathbf{w}) := \sum_n \gamma_n \|(\mathbf{w}_{g_n}, \mathbf{w}_{\text{all descendants of } g_n})\|_{q_n}. \tag{3.20}$$

[5]Of course, a group can also consist of only one feature.

The positive weights γ_n in this case are important to prevent over-penalisation of features that appear in many groups. To solve the problem (3.15) with Penalty (3.20), Zhao et al. (2009) proposed a modification of the least angle regression (LARS) algorithm discussed in Sec. 3.4.2.

Jacob et al. (2009) proposed to encode the overlapping groups through *latent variables*. Every time a feature that corresponds to the weight vector component w_j is included in a new group g, it receives a new coefficient v_j^g. For instance, if a feature appears in four different groups it also has four different coefficients to be estimated. The final coefficient of the feature is the sum of the four. The resulting *latent group lasso* penalty has the form

$$\Omega_{\text{union}}(\mathbf{w}) := \min_{\mathbf{v}\in\mathbb{R}^{M\times|\mathcal{P}|}} \sum_{g\in\mathcal{P}} \gamma_g \|\mathbf{v}^g\|_q \tag{3.21}$$

such that

$$\sum_{g\in\mathcal{P}} \mathbf{v}^g = \mathbf{w}$$
$$v_j^g = 0 \text{ if } j \notin g \text{ for every } g \in \mathcal{P}.$$

While both penalties (3.20) and (3.21) are referred to as "group lasso with overlapping groups", the semantics of the penalties are different. As pointed out by Mairal and Yu (2013), Penalty (3.20) prefers sparsity patterns that are *intersections* of the groups, while Penalty (3.21) prefers those sparsity patterns that are *unions* of the groups.

The definition of the groups suggested by Zhao et al. (2009) and described in Penalty (3.20) encourages the strong hierarchy. While it is unclear how to promote weak hierarchy with CAP, for the latent group lasso penalty (3.21) we could define groups that consist of the root node edges in DAG. In the next subsection we will see how this idea leads to an efficient algorithm for solving the minimisation problem in (3.21).

3.3.4. Path Coding Penalties and Network Flows

Mairal and Yu (2013) suggested a penalty framework that extends and generalises the notion of overlapping groups. They introduced a notion of groups as paths in a DAG. The path coding penalties they defined encourage sparsity patterns of subgraphs that can be covered by a small number of path. Even though the number of paths in a graph grows exponentially with the number of nodes, reduction of penalty computation to a min-cost-flow problem allows to make use of algorithms with polynomial time complexity.

Let $\mathcal{G}' = (G', E')$ be a directed acyclic graph obtained from a sparse grid DAG $\mathcal{G} = (G, E)$ (see Def. 2.1) as follows: $G' = \bar{G} \cup \{t\}$ a union of the sparse grid index-set and the auxiliary node t and E' describes the set of hierarchical

connections in the sparse grid

$$E' := E \cup \{(u, t) \mid u \in G\}.$$

The flow problems on graphs usually require a source node s, which in our case is the root node with the grid index $((1, \ldots, 1), (1, \ldots, 1))$. It also requires a sink node t which is not present in the sparse grid and so we added it to the DAG representation, connecting it to all normal nodes in the graph.

Every set g in the graph corresponds to a path from s to t, i.e. a collection of vertices $(v_1, \ldots, v_k)$ such that $v_1 = s$, $v_k = t$ and $(v_i, v_{i+1}) \in E'$ for $1 \leq i < k$.

Maybe your lectures on graph theory have been a while ago. So let us recapitulate the definition of a graph flow. It will also allow us to point out the important properties of a flow that we use.

Definition 3.2. *Let $\mathcal{G}' = (G', E')$ be a directed graph with the source node s and the sink node t. A **flow** f on the graph $\mathcal{G}'$ is a non-negative function defined on edges, denoted as $[f_{uv}]_{(u,v)\in E'}$. The value f_{uv} lies between l_{uv} and δ_{uv}—its lower and upper capacities (**capacity constraint**). For all vertices, except of s and t, the sum of their incoming flows is equal to the sum of their outgoing flows (**conservation constraint**).*

Let $\mathcal{P}$ be the set of all paths in $\mathcal{G}'$, $\mathcal{F}$ the set of all flows in $\mathcal{G}'$, and $s_j(f)$ the amount of flow going through the node $j \in G$:

$$s_j(f) := \sum_{u \in G' : (u,j) \in E'} f_{uj}.$$

Mairal and Yu (2013) define two *path coding penalties*

$$\phi_{\mathcal{P}}(\mathbf{w}) := \min_{f \in \mathcal{F}} \left\{ \sum_{(u,v)\in E'} f_{uv} c_{uv} \;\middle|\; s_j(f) \geq 1, \forall j : w_j \neq 0 \right\} \tag{3.22}$$

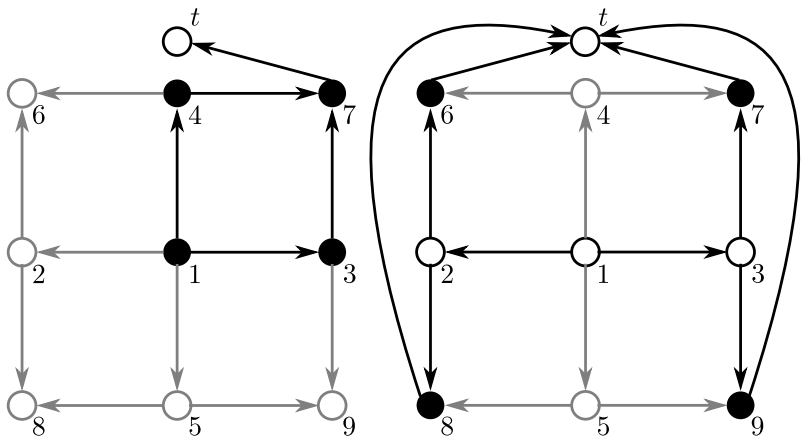
and

$$\psi_{\mathcal{P}}(\mathbf{w}) := \min_{f \in \mathcal{F}} \left\{ \sum_{(u,v)\in E'} f_{uv} c_{uv} \;\middle|\; s_j(f) \geq |\mathbf{w}_j|, \forall j \in \{1, \ldots, M\} \right\}. \tag{3.23}$$

To illustrate these definitions, let us consider two simple examples depicted in Fig. 3.5. The cost c_{uv} of all edges is 1. We consider two different instances of the weight vectors $\mathbf{w}$:

1. The entries 1, 3, 4, 7 of the weight vector $\mathbf{w}$ have the value 0.5, others are 0. Hierarchy of the solution is preserved.

2. The entries 6, 7, 8, 9 of the weight vector $\mathbf{w}$ have the value 0.5, others are 0. Hierarchy of the solution is not preserved.



(a) Hierarchy (b) No hierarchy

Figure 3.5.: Example of minimal flows in a sparse grid DAG. Every graph illustrates a subset of nodes of a two-dimensional sparse grid of level 3. The weight parameters corresponding to the filled nodes have value 0.5, others are 0. Black arrows identify the edges with nonzero flow. Path coding formulations favour the solutions that preserve hierarchy.

In order to calculate Penalty (3.22), we first set the lower bound for the amount of flow through the nodes corresponding to nonzero weight parameters

$$s_3(f) \geq 1, \quad s_4(f) \geq 1, \quad s_7(f) \geq 1. \tag{3.24}$$

It is now easy to see that the minimum flow will be achieved when the values of $s_j(f)$ from Inequalities (3.24) are close to 1, while the capacity constraint is still satisfied. In this particular case the minimum flow is

$$f_{1,3} = f_{1,4} = f_{4,7} = f_{3,7} = 1, \quad f_{7,t} = 2.$$

The total path coding penalty $\phi_{\mathcal{P}}(\mathbf{w})$ is 6.

Similarly, for Penalty (3.23) we have

$$\begin{aligned} s_3(f) \geq 0.5, \quad s_4(f) \geq 0.5, \quad s_7(f) \geq 0.5, \\ f_{1,3} = f_{1,4} = f_{4,7} = f_{3,7} = 0.5, \quad f_{7,t} = 1. \end{aligned}$$

The total path coding penalty $\phi_{\mathcal{P}}(\mathbf{w})$ is 3.

The second example can be calculated analogously. On Fig. 3.5 we show the edges with nonzero flow in black and the edges with zero flow in grey and leave the calculation of the numerical values as an exercise. Table 3.1 summarises the penalties for both examples. It shows that both path coding penalties favour solutions with preserved hierarchical dependencies.

Besides the traditional ℓ_2^2 regularisation function, traditionally used for training sparse grid models, there exist a number of sparsity-inducing norms with different properties that can be advantageous, for example, by promoting strong sparsity or weak hierarchy in the structure of the sparse grid models. Unfortunately, the standard conjugate gradient method will not work with the most of this penalties. In the next section we review the specialised optimisation algorithms that can be used instead.

3.4. Algorithms for Convex Optimisation with Sparsity-Inducing Penalties

The structure and complexity of the regularisation terms presented in the previous section as well as the tremendous size of the solution space makes simple optimisation methods, i.e. Newton method or conjugate gradients, insufficient. Therefore, in this section we review the state-of-the-art optimisation procedures applicable for training of sparse grid models with proposed regularisers. Later in Sec. 3.5, we compare these methods in a series of numerical experiments.

3.4.1. Proximal Methods

Consider the problem (3.15) where the loss function L is differentiable and the regularisation term Ω is not. In this case minimisation is often carried out using proximal methods, i.e. proximal gradient, forward-backward splitting, alternating direction method of multipliers, etc (Boyd et al., 2010; Bach et al., 2011). We discuss several popular proximal methods later in this section.

In every iteration t the loss function L is linearised around the current estimate $\mathbf{w}^t$:

$$\mathbf{w}^{t+1} := \underset{\mathbf{w}\in\mathbb{R}^M}{\arg\min} \underbrace{L(\mathbf{w}^t) + \nabla L(\mathbf{w}^t)^T(\mathbf{w}-\mathbf{w}^t)}_{\text{linear approximation of } L} + \underbrace{\frac{\rho}{2}\|\mathbf{w}-\mathbf{w}^t\|_2^2}_{\text{proximal term}} + \underbrace{\lambda\Omega(\mathbf{w})}_{\text{non-smooth part}}, \tag{3.25}$$

where $\rho > 0$ is an upper bound of ∇L. The quadratic term, called *proximal term*, keeps the update in the neighbourhood of $\mathbf{w}^t$ where L is close to its linearised approximation.

Since adding constants to the objective function in (3.25) or dividing it by a constant does not change the minimiser, we can add $1/(2\rho)\|\nabla L(\mathbf{w}^t)\|_2^2 - L(\mathbf{w}^t)$

Example	$\phi_{\mathcal{P}}(\mathbf{w})$	$\psi_{\mathcal{P}}(\mathbf{w})$
Strong hierarchy	6	3
No hierarchy	12	6

Table 3.1.: Path coding penalties.

Penalty Ω	**Operator** $\text{prox}_{\lambda\Omega}$	**Source**
ℓ_1-norm	$[\text{prox}_{\lambda\|\cdot\|_1}(\mathbf{w})]_j = \text{sgn}(\mathbf{w}_j)\,\|\|\mathbf{w}_j\| - \lambda\|_+$	(Bach et al., 2011)
ℓ_2^2 ridge regression	$\text{prox}_{\lambda/2\|\cdot\|_2^2}(\mathbf{w}) = (1+\lambda)^{-1}\mathbf{w}$	(Bach et al., 2011)
$\ell_1 + \ell_2^2$ elastic-net	$\text{prox}_{\lambda(\|\cdot\|_1+\gamma/2\|\cdot\|_2^2)}() = \text{prox}_{\lambda\gamma/2\|\cdot\|_2^2} \circ \text{prox}_{\lambda\|\cdot\|_1}$ $= (\lambda\gamma+1)^{-1}\,\text{prox}_{\lambda\|\cdot\|_1}$	(Bach et al., 2011)
ℓ_1/ℓ_q "group lasso"	$[\text{prox}_{\lambda\Omega}(w)]_g = \|1 - \lambda/\|\mathbf{w}_g\|_2\|_+\,\mathbf{w}_g, g \in \mathcal{G}$	(Bach et al., 2011)
$\phi_{\mathcal{P}}$ penalty (3.22)	$[\text{prox}_{\phi_{\mathcal{P}}}(\mathbf{v})]_j = \begin{cases}\mathbf{v}_j & \text{if } s_f(f^\star) > 0 \\ 0 & \text{otherwise}\end{cases}$ with $f^\star \in \arg\min_{f\in\mathcal{F}} \sum_{(u,v)\in E} f_{uv}c_{uv} + \sum_{j=1}^{M} \frac{1}{2}\left\|\mathbf{v}_j^2(1-s_j(f))\right\|_+$	(Mairal & Yu, 2013)
$\psi_{\mathcal{P}}$ penalty (3.23)	$[\text{prox}_{\psi_{\mathcal{P}}}(\mathbf{v})]_j = \text{sgn}(\mathbf{v}_j)\min\{\|\mathbf{v}_j\|, s_j(f^\star)\}$ with $f^\star \in \arg\min_{f\in\mathcal{F}} \sum_{(u,v)\in E} f_{uv}c_{uv} + \sum_{j=1}^{M} \frac{1}{2}\,\|\|\mathbf{v}_j\| - s_j(f)\|_+^2$	(Mairal & Yu, 2013)

Table 3.2.: Proximal operators for penalties from Sec. 3.3.

and divide the objective function by ρ obtaining the following form:

$$\begin{aligned}\mathbf{w}^{t+1} &:= \arg\min_{\mathbf{w}\in\mathbb{R}^M} \tfrac{1}{2}\|\tfrac{1}{\rho}\nabla L(\mathbf{w}^t)\|_2^2 + \tfrac{1}{\rho}\nabla L(\mathbf{w}^t)^T(\mathbf{w}-\mathbf{w}^t) + \tfrac{1}{2}\|\mathbf{w}-\mathbf{w}^t\|_2^2 + \tfrac{\lambda}{\rho}\Omega(\mathbf{w}) \\ &= \arg\min_{\mathbf{w}\in\mathbb{R}^M} \tfrac{1}{2}\|\mathbf{w} - \left(\mathbf{w}^t - \tfrac{1}{\rho}\nabla L(\mathbf{w}^t)\right)\|_2^2 + \tfrac{\lambda}{\rho}\Omega(\mathbf{w}). \end{aligned} \tag{3.26}$$

This allows us to define the proximal operator associated with a particular regularisation term $\lambda\Omega$.

Definition 3.3. *The* ***proximal operator*** *associated with the regularisation term* $\lambda\Omega$ *is the map*

$$\begin{aligned}\text{prox}_{\lambda\Omega} : \mathbb{R}^M &\to \mathbb{R}^M \\ \mathbf{v} &\mapsto \arg\min_{\mathbf{w}\in\mathbb{R}^M} \tfrac{1}{2}\|\mathbf{v}-\mathbf{w}\|_2^2 + \lambda\Omega(\mathbf{w}). \end{aligned} \tag{3.27}$$

Using the definition above we can compute the update (3.26) as

$$\mathbf{w}^{t+1} := \text{prox}_{\frac{\lambda}{\rho}\Omega}(\mathbf{w}^t - \tfrac{1}{\rho}\nabla L(\mathbf{w}^t)).$$

Understandably, the definition of efficient proximal operators for different penalties Ω is of great importance and was the focus of active research in the recent years. The proximal operators for the regularisation terms from Sec. 3.3 are summarised in Tab. 3.2 along with the references.

Algorithm 9: ISTA with Backtracking, adapted from (Beck & Teboulle, 2009)

Initialise $\rho_0 > 0, \eta > 1, \mathbf{w}^0 \in \mathbb{R}^n$
for $t = 1, \ldots$ **do**
  Find the smallest non-negative i_t such that for $\rho^* := \eta^{i_t}\rho_{t-1}$

$$F(\text{prox}_{\frac{\lambda}{\rho^*}\Omega}(\mathbf{w}^{t-1})) \leq Q_{\rho^*}(\text{prox}_{\frac{\lambda}{\rho^*}\Omega}(\mathbf{w}^{t-1}), \mathbf{w}^{t-1}). \tag{3.30}$$

  Set $\rho_t = \rho^*$ and compute

$$\mathbf{w}^t = \text{prox}_{\frac{\lambda}{\rho_t}\Omega}(\mathbf{w}^{t-1}). \tag{3.31}$$

Choice of ρ and acceleration

In general, the upper bound ρ is unknown and has to be estimated in a line-search procedure. Beck and Teboulle (2009) suggested a backtracking procedure called iterative shrinkage-thresholding algorithm (ISTA) and presented in Alg. 9. Let $F(\mathbf{w})$ be the objective function of the minimisation problem (3.15)

$$F(\mathbf{w}) := \tfrac{1}{N}\sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i; \mathbf{w})) + \lambda\Omega(\mathbf{w}) \tag{3.28}$$

and $Q(\mathbf{v}, \mathbf{w})$ the objective function of the minimisation problem (3.25):

$$Q_\rho(\mathbf{v}, \mathbf{w}) := L(\mathbf{w}) + \nabla L(\mathbf{w})^T(\mathbf{v} - \mathbf{w}) + \tfrac{\rho}{2}\|\mathbf{v} - \mathbf{w}\|_2^2 + \lambda\Omega(\mathbf{v}). \tag{3.29}$$

ISTA produces a sequence of positive non-increasing values $\{F(\mathbf{w}^t)\}_t$ such that

$$F(\mathbf{w}^t) \leq Q_{\rho_t}(\mathbf{w}^t, \mathbf{w}^{t-1}) \leq Q_{\rho_t}(\mathbf{w}^{t-1}, \mathbf{w}^{t-1}) = F(\mathbf{w}^{t-1}).$$

Beck and Teboulle (2009) also suggested an accelerated version called fast iterative shrinkage-thresholding algorithm (FISTA) and presented in Alg. 10. FISTA maintains two additional variables and combines them with the information from the previous step to achieve faster convergence at marginal computational overhead.

If L is Lipschitz-continuous, like the square loss function, ISTA has the global convergence rate in $\mathcal{O}(1/t)$, while FISTA has the convergence rate in $\mathcal{O}(1/t^2)$, which is optimal for this class of techniques (Nesterov, 2004).

Algorithm 10: FISTA with backtracking, adapted from (Beck & Teboulle, 2009)

Initialise $\rho_0 > 0, \eta > 1, \mathbf{w}^0 \in \mathbb{R}^M$
Set $\mathbf{v}^1 = \mathbf{w}^0, \gamma_1 = 1$
for $t = 1, \ldots$ **do**
  Find the smallest non-negative i_t such that for $\rho^* := \eta^{i_t}\rho_{t-1}$

$$F(\text{prox}_{\frac{\lambda}{\rho^*}\Omega}(\mathbf{v}^t)) \leq Q_{\rho^*}(\text{prox}_{\frac{\lambda}{\rho^*}\Omega}(\mathbf{v}^t), \mathbf{v}^t). \tag{3.32}$$

  Set $\rho_t = \rho^*$ and compute

$$\mathbf{w}^t = \text{prox}_{\frac{\lambda}{\rho_t}\Omega}(\mathbf{v}^t), \tag{3.33}$$

$$\gamma_{t+1} = \frac{1 + \sqrt{1 + 4\gamma_t^2}}{2}, \tag{3.34}$$

$$\mathbf{v}^{t+1} = \mathbf{w}^t + \frac{\gamma_t - 1}{\gamma_{t+1}}(\mathbf{w}^t - \mathbf{w}^{t-1}). \tag{3.35}$$

3.4.2. Active-Set Methods and Least Angle Regression

Active-set methods explicitly take into account the sparse structure of the solution allowing it to grow incrementally (Nocedal & Wright, 2006). This increases the size of tractable problems and makes active-set methods especially well-suited for training sparse grid models.

An active-set algorithm minimises the overall problem by solving a sequence of subproblems. It strongly resembles the method for training adaptive sparse grid models presented in Sec. 2.3 and, in fact, it generalises Alg. 1. Therefore, Alg. 11 presents the active-set method in a slightly unconventional form to underline these similarities.

An active-set method requires the definition of three components:

Inner-loop solver: At each iteration step we solve Problem (3.15) considering only the grid indices from $G^{(k)}$. Hence, we automatically guarantee that $w_g = 0$ for $g \notin G^{(k)}$. The solution of the subproblem can be done, for example, using FISTA from the previous subsection. Further acceleration is achieved using the solution from the previous iteration as warm-start.

Optimality condition: Conventionally, the global convergence of the active-set algorithm is verified using the duality gap. Let R^*_{emp} be the Fenchel conjugate of the empirical risk functional defined in (2.4) and Ω^* the dual of the sparsity-inducing regularisation term. If $\mathbf{w}^k$ is the solution from

Algorithm 11: Active-set optimisation

input : $G^{(0)}$
output: $G^{(k)}$ and $\mathbf{w}^k$
for $k = 0, 1, \ldots$ **do**
  Fit-Step: ***solve*** (3.15) on $G^{(k)}$ to obtain $\mathbf{w}^k$
  if ***optimality condition*** *satisfied* **then**
    break
  Adjust-Step:
  - identify the set of new features I and ***extend*** the active set
  - identify the set of elements J from $G^{(k)}$ with zero components in $\mathbf{w}$ and remove them from the active set

$$G^{(k+1)} = \left(G^{(k)} \setminus J\right) \cup I$$

inner-loop solver then it follows (Bach et al., 2011)

$$L(\mathbf{w}^k) + \lambda\Omega(\mathbf{w}^k) + R^*_{\text{emp}}(\nabla R_{\text{emp}}(\mathbf{w}^k)) = 0 \tag{3.36}$$

and $\mathbf{w}^k$ is the solution of the full problem if and only if

$$\Omega^*(\nabla R_{\text{emp}}(\mathbf{w}^k)) \leq \lambda. \tag{3.37}$$

Unfortunately, the closed form of the dual function Ω^* is not always available or the computation cannot be done efficiently. In this case we can recourse to other convergence criteria such as MSE threshold, failure to improve, or time restrictions.

Feature selection strategy: A natural way to select the new features is to look at the violators of Condition (3.37). If this is unfeasible, one can try to maximise the negative marginal gain as discussed in Sec. 3.2.2. In this case the refinement indicators for the regularisation terms from Sec. 3.3 can be derived similarly to the one for ℓ_2^2-norm in Lemma 3.2. The question of whether to select a single variable or a group depends on the structure of Ω and computational considerations.

Least angle regression (LARS) is an efficient algorithm for computing the regularisation path for lasso (Efron et al., 2004). Similar to the forward greedy selection method, LARS sequentially identifies the important features taking only "as much as need" of each. It starts by identifying the feature that is most correlated with the target vector. But instead of fitting the feature completely to the target, LARS moves the coefficient of the feature continuously towards its optimal value, which decreases the correlation with the residual vector. At some

point the second feature becomes the same correlation coefficient as the first one and joins the active set. Now two coefficients are moved together, which causes the correlation to further decrease. And so forth.

The change of coefficients in LARS is piecewise linear, which enables one to calculate the exact step size at the beginning of each step. To obtain the entire lasso path, LARS drops the variables from active set when the corresponding coefficient becomes zero and recomputes the direction.

Even though the LARS algorithm computes the complete regularisation path, it is as efficient a single least squares regression. In the model with M features LARS takes exactly M steps to obtain the full least squares estimates (Hastie et al., 2011, Sec. 3.4.4).

3.4.3. Coordinate Descent

Friedman et al. (2010) showed empirically that the regularisation path can be computed using the coordinate descent algorithm (CD) with active set much faster than using LARS. At the same time, Bach et al. (2011) showed that a block coordinate descent algorithm (BCD) with active set outperformed ISTA, FISTA, and subgradient descent on medium- and large-scale problems with ℓ_1/ℓ_2 and ℓ_1/ℓ_∞ regularisation penalties. Hence, CD with active set is a promising method for adaptive sparse grid learning.

The principle behind CD is fairly simple: it cycles through the variables optimising only one at a time. In the context of linear equation solvers it is known as the Gauss-Seidel procedure. While not necessarily convergent in general, CD converges for problems with square loss and separable regularisation terms (Tseng, 2001).

Following Friedman et al. (2010), we consider CD for the elastic-net penalty

$$\Omega_\alpha(\mathbf{w}) = (1-\alpha)\tfrac{1}{2}\|\mathbf{w}\|_2^2 + \alpha\|\mathbf{w}\|_1$$

that combines ℓ_1- and ℓ_2^2-norm penalties (see Sec. 3.3.2). Suppose that we already optimised (3.15) with respect to $\widetilde{w}_h$ for $h \neq j$ and now we want to partially optimise it with respect to w_j. In order to do this, we need to compute the gradient at $\widetilde{w}_j$ from the previous estimation. This is only possible if $\widetilde{w}_j \neq 0$. For example for $\widetilde{w}_j > 0$ we have

$$\left.\frac{\partial R_\Omega}{\partial w_j}\right|_{w_j=\widetilde{w}_j} = -\tfrac{1}{N}\sum_{i=1}^{N}\phi_j(\mathbf{x}_i)(y_i - f(\mathbf{x}_i;\widetilde{\mathbf{w}})) + \lambda(1-\alpha)w_j + \lambda\alpha. \tag{3.38}$$

In general, the coordinate-wise update has the form

$$\widetilde{w}_j := \frac{S\left(1/N\sum_{i=1}^{N}\phi_j(\mathbf{x}_i)(y_i - \tilde{y}_i^{(j)}), \lambda\alpha\right)}{1+\lambda(1-\alpha)}, \tag{3.39}$$

where

- $\widetilde{y}_i^{(j)} = \sum_{h \neq j} \phi_h(\mathbf{x}_i)\widetilde{w}_h$ is the fitted value excluding the contribution from $\phi_j(\mathbf{x}_i)$, and hence $y_i - \widetilde{y}_i^{(j)}$ is the partial residual for fitting w_j,
- $S(z, \gamma) := \operatorname{sgn}(z)|z| - \gamma|_+$ is the soft-threshold operator.

To obtain the regularisation path, we compute the solutions for a decreasing sequence of values λ, starting with the largest value $\lambda_{\max}$ which causes the entire vector $\widetilde{\mathbf{w}}$ to be zero. When $\widetilde{\mathbf{w}}$ is zero, Equation (3.39) indicates that $\widetilde{w}_j$ remains zero if $1/N|\langle \boldsymbol{\phi}_j, \mathbf{y}\rangle| < \lambda\alpha$. This is true if we select $\lambda_{\max} = \max_h |\langle \boldsymbol{\phi}_h, \mathbf{y}\rangle|/(N\alpha)$.

Regularisation path is constructed from a decreasing sequence of K values between $\lambda_{\max}$ and $\lambda_{\min} = \epsilon\lambda_{\max}$ on the logarithmic scale. Friedman et al. (2010) suggested typical values $\epsilon = 0.001$ and $K = 100$. Apart from giving us the regularisation path, this scheme exploits warm-starts and leads to a more stable algorithm.

Minimisation with sparsity-inducing penalties requires specialised optimisation algorithms. Fortunately, many efficient algorithms for solving these problems appeared in the recent years. As we will see in the next section, these algorithms can be readily adapted to our purposes, which leads to improved learning techniques for adaptive sparse grid models.

3.5. Numerical Experiments

It is now time to subject our refinement strategies and sparsity-inducing penalties to a test. We begin by reconstructing the decision boundaries of a two-dimensional rhombus. Then we consider a six-dimensional problem of photometric redshift prediction frequently discussed in connection with sparse grid regression. Finally, we illustrate how using the new adaptive sparse grid methods we can approach a ninety-dimensional regression problem of predicting a year of creation of different songs.

In order to obtain the numerical results in this and the following chapters, I implemented the described methods in SG^{++}—an open source sparse grid software framework developed in the research groups at the chair for scientific computing at the Technische Universität München and the institute for parallel and distributed systems at the Universität Stuttgart and available at `sgpp.sparsegrids.org`.

3.5.1. Two-Dimensional Rhombus

At the beginning of Sec. 3.1 we motivated the sparse grid refinement as the process of manifold learning. As an illustration, in Fig. 3.1 we considered the

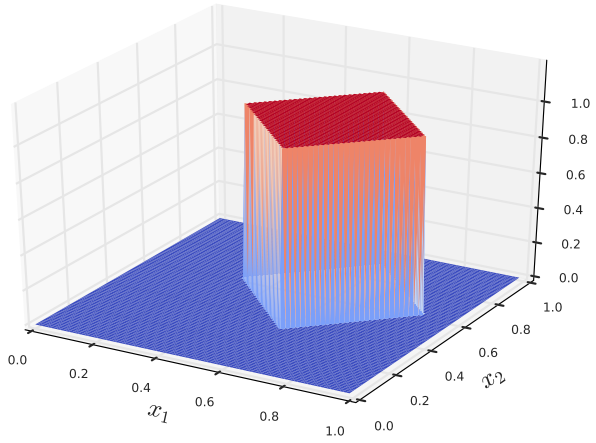


Figure 3.6.: Two-dimensional function with rhombus decision boundaries. This problem poses a challenge for sparse grids since the boundaries are not axis-aligned.

approximation of decision boundaries of a rhombus. Now we return to this example and consider it as a benchmark for evaluating the performance of different regularisation functions and refinement strategies.

Figure 3.6 illustrates the function we want to approximate. It represents a two-dimensional rhombus such that all data points outside of the rhombus have the value 0 and the points inside the rhombus have the value 1. While this two-dimensional synthetic problem may seem simple, in fact it is not. The boundaries of the rhombus are non-smooth and unaligned with respect to the axis. In contrast, the sparse grid models are usually more successful at approximating the function irregularities located parallel to the axis since the position of the basis functions is axis-aligned.

Table 3.3 summarised the methods we compare in this section. The method *elastic-net* uses the coordinate descent (CD) optimisation method (Sec. 3.4.3) combined with the elastic-net regularisation penalty (Sec. 3.3.2). The method *forward-backward* is an implementation of the adaptive forward-backward greedy selection algorithm (Sec. 3.2.3), where we compute the potential contribution of new grid points using formula (3.7) and the parameter fitting is done using the conjugate gradients (CG) method. The *ridge* regression method is the standard regression method by Pflüger (2010) that combines ℓ_2^2 regularisation penalty, CG parameter fitting, and the spatially-adaptive refinement of children indices in all dimensions (2.39) using the refinement indicator (2.40). The *lasso* method differs from *ridge* in the application of the ℓ_1 penalisation term and FISTA optimisation method (Alg. 10). The methods *greedy ridge* and *greedy lasso* are the modifications of the *ridge* and *lasso* methods where we apply spatially-dimension-adaptive refinement adding only children indices in particular dimensions and using the refinement indicator (3.7).

To conduct the experiment we sampled 2000 data points on the function and split them into a training set (80%) and a validation set (20%).[6] The validation set was used to identify the optimal values for various hyperparameters of the respective methods. We identified the optimal hyperparameters presented in Table 3.4 using a Bayesian optimisation procedure (Martinez-Cantin, 2014).

Usually, the hyperparameters would include the regularisation parameter λ, the number of fit-refine steps we conduct until convergence and the number of grid functions we refine. We would stop the alternation of the fit-refine steps if the sparse grid size becomes larger than 1000, which means we surpass the budget, or if the validation error increases, which means the model starts to overfit. Hence, the final grid sizes in Fig. 3.7 below may differ.

For the methods with spatially-adaptive refinement strategy the parameter *refinement points* denotes the number of parent grid indices that are to refine, hence in two-dimensional space on average four times as many new grid indices are create at every refinement step. For the methods with spatially-dimension-adaptive refinement strategy this parameter denotes the number of children pairs in a particular dimension that are to be created, hence on average twice as many new grid indices are created at every refinement step.

Table 3.4 shows that the optimal regularisation parameter λ is invariant to the choice of the refinement strategies: for ridge and greedy ridge as well as for lasso and greedy lasso the optimal values of λ are very similar. For the elastic-net method we do not estimate the optimal hyperparameters since the coordinate descent algorithm is used to compute the complete regularisation path.

Once the optimal hyperparameters were chosen, we retrained the models on the complete training dataset and computed a test error on 2000 data points of the test dataset. We present the test errors in Fig. 3.7. The greedy lasso method outperforms others once the grid becomes larger than 400 points; greedy ridge and forward-backward method show very similar performance; ridge and lasso methods also show similar performance. For more than 1000 grid points elastic-net shows superior results.

Method	**Sparsity**	**Fit-Step**	**Ref. Strategy**	**Ref. Indic.**
Elastic-Net	Elastic-Net	CD	Spat.-Dim.	
Forward-Backward	Greedy Sel.	CG	Spat.-Dim.	(3.7)
Ridge	ℓ_2^2	CG	Spat.-Adapt.	(2.40)
Greedy Ridge	ℓ_2^2	CG	Spat.-Dim.	(3.7)
Lasso	ℓ_1	FISTA	Spat.	(2.40)
Greedy Lasso	ℓ_1	FISTA	Spat.-Dim.	(3.7)

Table 3.3.: Summary description of methods in the experiment.

[6] The use of a cross-validation procedure did not change the results, which suggests that the sample size was sufficiently large.

Method	λ	Ref. points	Ref. steps
Forward-Backward	$1.02 \cdot 10^{-4}$	27	5
Ridge	$1.27 \cdot 10^{-4}$	35	11
Greedy Ridge	$9.56 \cdot 10^{-5}$	42	10
Lasso	$5.15 \cdot 10^{-5}$	46	10
Greedy Lasso	$4.37 \cdot 10^{-5}$	41	11

Table 3.4.: Optimal hyperparameters for the 2-d rhombus benchmark.

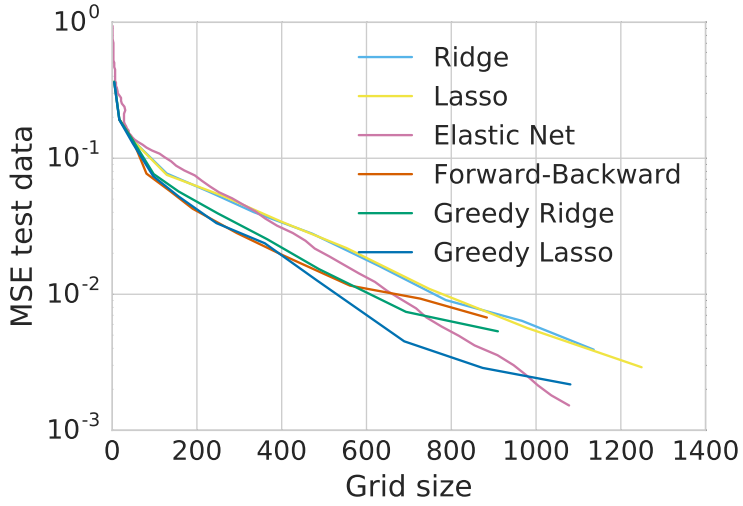


Figure 3.7.: Test error for different refinement methods. The greedy lasso method outperforms other methods once the grid becomes larger than 400 grid points. For more than 1000 grid points elastic-net shows superior results.

Method	λ	Ref. points	Ref. steps
Ridge w/o Rosenblatt	$1 \cdot 10^{-6}$	20	20
Ridge w. Rosenblatt	$1 \cdot 10^{-6}$	20	20
Greedy Ridge w/o Rosenblatt	$1 \cdot 10^{-6}$	100	20
Greedy Ridge w. Rosenblatt	$1 \cdot 10^{-6}$	50	20

Table 3.5.: Optimal hyperparameters for SDSS DR5 benchmark.

This experiment shows that the methods for fitting and refining the sparse grid models presented in this chapter are able to outperform the standard ridge regression method for spatially-adaptive sparse grids. Altogether, greedy ridge regression for spatially-dimension-adaptive sparse grids shows promise, producing a good compromise between accuracy and computational efficiency. Let us now look at how this method compares to the standard ridge regression in a real-world setting.

3.5.2. Photometric Redshift Prediction

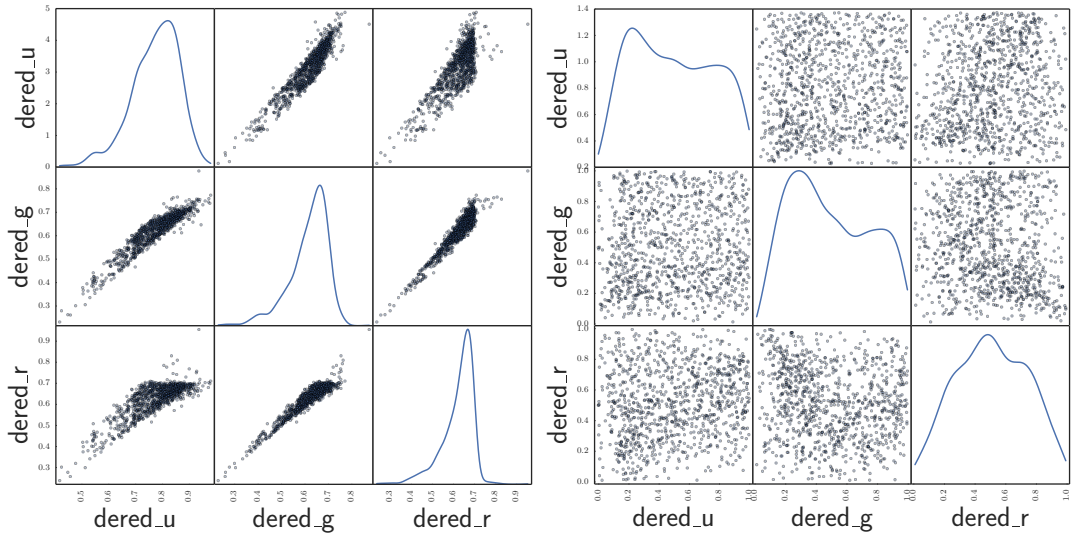
We now assess the performance of the greedy ridge and standard ridge regression methods using the data from the Sloan Digital Sky Survey (SDSS) from Data Release 5 (Adelman-McCarthy et al., 2007). The goal is to predict the redshift of galaxies from the Main Galaxy Sample using 6 features from the dataset: dereded intensities in five broadbands (*ugriz*) together with a meta-parameter *eClass* (Pflüger, 2010). Both training and test datasets contain 60 000 data points each.

Usually, sparse grid regression utilises the normalisation technique where input variables are mapped into a unit hypercube by shifting and scaling (see Fig. 3.8a) (Garcke, 2004). High correlation between attributes can alas inhibit the effectiveness of spatial adaptivity. Alternatively, one can use the transformation suggested by Rosenblatt (1952), which effectively makes the data distribution uniform (see Fig. 3.8b).

As we observed in the previous section, different refinement strategies do not influence the choice of the optimal regularisation parameter λ, and we set λ equal to $1 \cdot 10^{-6}$ for all experiment configurations. Table 3.5 summarises the optimal hyperparameters obtained in a validation procedure.

For fitting of the spatially-dimension-adaptive sparse grid models we used the averaged stochastic gradient descent algorithm (Bottou, 2012). As shown in (Khakhutskyy & Hegland, 2016), this method leads to the similar results as the conjugate gradient method but with a faster initial convergence.

Since we only want to compare the new method with the state of the art, we use a hard termination criterion: the training stops after a certain number of refinement steps (even if the global convergence has not been achieved yet).



(a) Shifting and scaling (b) Rosenblatt transformation

Figure 3.8.: Illustration of the normalisation methods for the first three attributes of the SDSS Data Release 5 dataset. The Rosenblatt transformation removes the correlation between attributes and produces almost uniformly distributed input values.

Figure 3.9 illustrates the decline of the training and test errors for different learning scenarios. Note that the algorithms favour different normalisation methods: While the spatially-adaptive ridge regression method produces better results if the data simply shifted and scaled to a unit hypercube, the spatially-dimension-adaptive greedy ridge regression benefits from the uniform distribution of the input data. There is a big gap between the performance of the ridge regression with the Rosenblatt transformation and the performance of the rest. Hence, to compact the representation we break the continuity of the ordinate axis, which depicts the root mean squared error (RMSE). Instead of 0, the ordinate axis starts at $1.8 \cdot 10^{-2}$ which seems to be the optimum for published nonlinear models (Pflüger, 2010, Tab. 6.8).

The best greedy ridge regression model has RMSE of $1.82 \cdot 10^{-2}$ on the test data with 4 247 grid indices. This is significantly better than $2.00 \cdot 10^{-2}$ with 9 940 grid indices of the best standard ridge regression model.

Altogether, the results show that the spatially-dimension-adaptive greedy ridge regression method can build smaller sparse grid models which have the same predictive power as the state of the art. In (Khakhutskyy & Hegland, 2016) we also showed that, combined with stochastic gradient descent optimisation methods, it may require just a fraction of the number of data passes to build and train a new model. The next section shows that with the spatially-dimension-adaptive greedy ridge regression we can approach the large high-dimensional problems that were unattainable by the spatially-adaptive method.

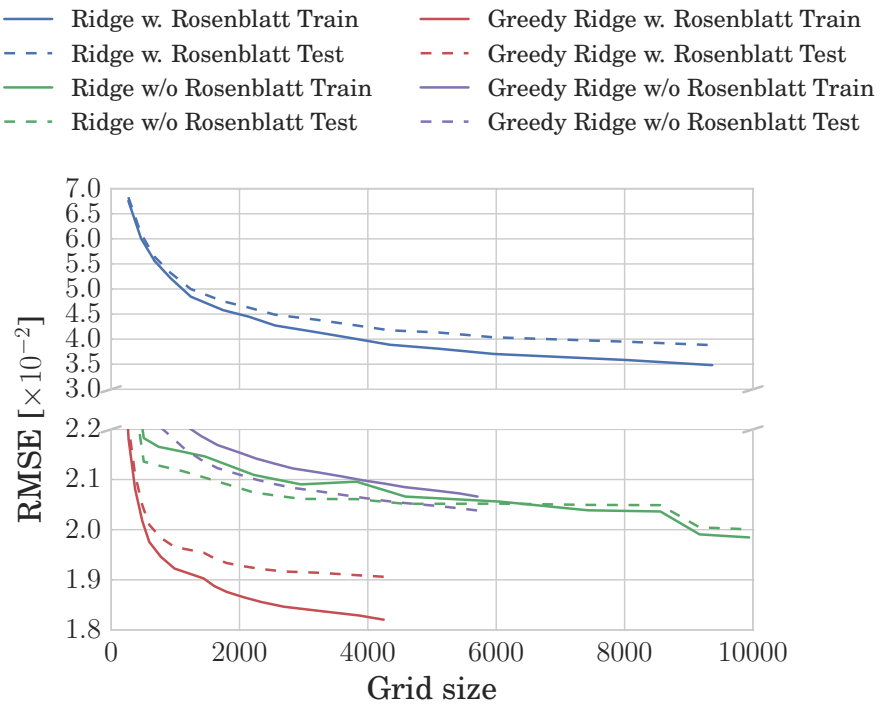


Figure 3.9.: Performance comparison between Ridge and Greedy Ridge regression methods. The Greedy Ridge method clearly shows the ability to create smaller sparse grids with the same representation power.

3.5.3. Million Song Dataset: Year Prediction

Sparse grid models were used to approach problems with over 100 dimensions before. Pflüger (2010) used a level 2 regular sparse grid model for the Musk-1 classification problem. The dataset contains 166 attributes and 476 data points such that the sparse grid contained 333 basis functions. The result compared favourably with another machine learning models. However, the use of a level 2 regular sparse grid cannot capture high nonlinearity. The spatially-adaptive refinement of such a large problem is prohibitively expensive and, due to the small sample size, may lead to premature overfitting.

In this experiment we choose another high-dimensional problem, which has less attributes but significantly more samples. We consider a 90-dimensional regression problem of predicting the year of creation of music titles based on their acoustic characteristics. The Million Song Dataset (MSD) collected by Bertin-Mahieux et al. (2011) contains information about 515 576 music titles representing 28 223 different artists and covering the timeline between 1922 and 2011. Each title is described by 90 values derived from the averages and covariances of the timbre vectors. The dataset is split into the training set with 463 715 titles and the test set with 51 630 titles such that all songs by the same artist will appear either in the training or in the test set (to avoid the "producer effect").

For this problem the use of a regular sparse grid model becomes impractical for any but a trivial case. For a regular sparse grid model with maximum level 2 we have only 181 basis functions, whereas for the level 3 it becomes 16 561 and for the level 4 it becomes 1 021 201 basis functions. Level 3 allows only for low local resolution of the model and level 4 is computationally infeasible and would require much more data samples than are available.

A typical spatially-adaptive refinement also quickly becomes problematic since for every refinement grid point at least 180 new basis functions are added. Hence, the number of basis functions explodes. To counteract this, we apply the spatially-dimension-adaptive refinement with ridge regression that recommended itself in the previous experiments. We start with a level 2 sparse grid and use modified linear basis functions.

As we start with the level 2 sparse grid, we would need to calculate the refinement indicators for more than 16 000 grid points at every refinement step. The time for this computation can quickly dominate the fitting time. Moreover, usually it is sufficient to look for refinement candidates in a few areas with a high local error. Hence, to accelerate the refinement step, the error-based refinement indicator (2.40) was used to sort the refinable parent nodes and then the spatially-dimension adaptivity indicator (3.7) were computed for only a subset of these nodes.

To estimate the optimal hyperparameters, we selected 10% of the points from the training dataset—46 372 data samples. From this subset, 80% of the points were used for training and 20% for validation of the hyperparameters. The

Method	RMSE	Reference
1-NN	13.99	(Bertin-Mahieux et al., 2011)
Constant Pred.	10.80	(Bertin-Mahieux et al., 2011)
50-NN	10.20	(Bertin-Mahieux et al., 2011)
Vowpal Wabbit	8.76	(Bertin-Mahieux et al., 2011)
Linear SVM	11.08	(Ho & Lin, 2012)
RBF SVM	10.11	(Ho & Lin, 2012)
Lasso	9.34	(Arnaldo et al., 2015)
Multiple Linear Regression	9.34	(Arnaldo et al., 2015)
Multiple Regr. Genetic Programm.	9.34	(Arnaldo et al., 2015)
FFNN	8.78	(Arnaldo et al., 2015)
Adaptive Sparse Grids	**9.00**	

Table 3.6.: Results for the year prediction challengen in MSD. Adaptive sparse grids show a competitive performance.

optimal results were achieved with the regularisation term $\lambda = 7 \cdot 10^{-9}$ and refinement of 50 pairs of points in every refinement step (similar to the previous section).

Once the optimal hyperparameters were determined, we trained the model on the complete training set and evaluated the root-mean-square error on the test set. Using the adaptive sparse grids we were able to achieve the test error of 9.00 using a model with 2 555 grid points. Table 3.6 compares this result with other results described in the literature. One can see that adaptive sparse grids are among the best models in the comparison, significantly outperforming most of the linear and nonlinear models. However, as it is often the case, an unambiguous comparison is difficult. For example, the results published by Arnaldo et al. (2015) are used to illustrate their feature selection and generation methods and, hence, the feed-forward neural network (FFNN) were trained on a transformed dataset, whereas for the adaptive sparse grids and for the Vowpal Wabbit in (Bertin-Mahieux et al., 2011) no feature generation is taking place.

Figure 3.10 compares the decrease of the test error with advancing refinement steps and increasing grid size for Greedy Ridge and Ridge methods. It shows that it is unnecessary to have a large grid to achieve good results as the final grid size is still much smaller than the size of a regular sparse grid with level 3. The best test error achieved for the Ridge method was 9.02 for 8 442 greedy points. Moreover, the diagram suggests that the spatially-adaptive sparge grid model run into the overfitting phase and does not allow any further improvement while the spatially-dimension-adaptive methods continues to improve.

We were able too successfully build an adaptive sparse grid model for a regression problem with 90 real-valued attributes. This became possible due to a greedy refinement technique that controls the sparsity of the model in a more granular way than the standard spatially-adaptive method. To improve the com-

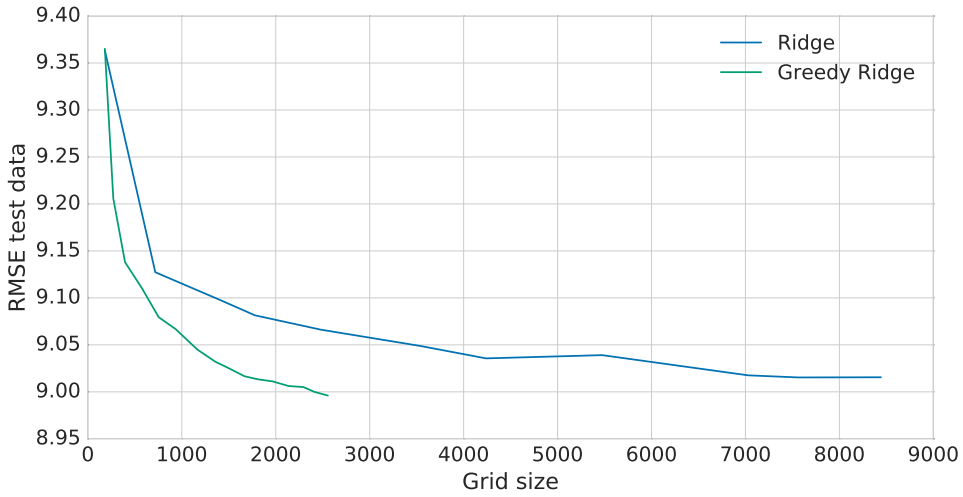


Figure 3.10.: Test error versus sparse grid size for the year prediction challenge MSD. Despite the high dimensionality of the problem large sparse grid models are unnecessary. Greedy Ridge achieves better prediction error with a quarter of the grid size.

putational performance, we were adding several new grid points at once and we were considering only the most promising subsets of the refinement candidates. These measures were necessary for the method to be practical for large-scale learning. In the result, our model performed among the best models reported in the literature.

The key to tackling a high-dimensional problem with a low-dimensional manifold is to identify this manifold and to use it for learning. Spatial adaptivity of the sparse grid models allows us to identify such low-dimensional manifolds directly during the learning process. Consideration of the refinement problem in a formal framework leads to a formulation of a discrete optimisation problem, which is usually exponentially hard to solve directly. To overcome this complexity, we can either relax the optimality of the solution, which leads to the various greedy optimisation methods, or relax the problem formulation, which leads to the convex sparsity-inducing norms.

As we have seen in the experiments, both greedy optimisation methods and convex relaxation methods can yield similar results. Usually, this methods produce models with smaller size and better accuracy than the standard spatially-adaptive sparse grid methods.

If a high-dimensional problem does not possess a low-dimensional manifold, sometimes it still may be solved efficiently. Even if the nominal dimensionality of a problem is high, the effective dimensionality can be low. In the next chapter we discuss this case in detail.

4. Additive Structure and Effective Dimensionality

> *Capt. Kirk: Spock, give me an update on the dark area ahead.*
> *Mr. Spock: No analysis due to insufficient information.*
> *Capt. Kirk: No speculation, no information, nothing? I've asked you three times for information on that thing and you've been unable to supply it. Insufficient data is not sufficient, Mr. Spock! You're the science officer. You're supposed to have sufficient data all the time.*
>
> — Star Trek. *The Immunity Syndrome*

Imagine, you need to assemble a team of experts. How would you do it? This depends on the goal this team has to accomplish. If the goal is to define a new policy, you would invite experts with different backgrounds and values, who would discuss, negotiate, and vote to find a compromise decision. The more diverse your policy makers are, the more different their values and interests—the more agreeable is the result. In contrast, if the goal is, for example, to win a football championship, you would look for players with complementary expertise. They would train relentlessly to work together as a team and to achieve the same goal.

What does this have to do with sparse grids? Just as it is simpler to find a team of experts to cover a broad set of skill and qualifications than to find one super-human that could do everything, it is often simpler to consider a set of low-dimensional models instead of one high-dimensional. In this sense, the two metaphors for creating a team of experts provide an intuition for the two approaches for training sparse grid models discussed in this chapter. Both approaches for reducing the number of degrees of freedom rely on the assumption that the problem possess an additive structure, meaning that the true high-dimensional solution can be approximated as a sum of low-dimensional solutions. This also implies that the interaction between subsets of dimensions are not important for practical solution, such that higher-order interaction terms can be dismissed. This property is called a low effective dimensionality.

The first metaphor of policy makers corresponds to an ensemble of independent low-dimensional sparse grid models that solve a high-dimensional regression problem described in Sec. 4.1. To the best of my knowledge, this work is the first to consider the use of ensembles of sparse grids with random projections.

It shows that having a number of low-dimensional sparse grids instead of one high-dimensional can reduce the number of degrees of freedom—and hence the time and space complexities—by several magnitudes. Empirical results show that such ensembles produce satisfactory results and hence provide a simple and practical way to tackle high-dimensional problems.

The second metaphor of a football team relates to identifying and training important parts of a single sparse grid model necessary for the solution of a high-dimensional learning problem. We discuss these methods in Sections 4.2 and 4.3. We gain two major insights: First, the use of modified linear basis leads to correct identification of functional components and inherits (at least from the practical point of view) the rich theory available for prewavelet basis, while prewavelet basis itself is impractical for spatial adaptivity. Second, extending the sparsity-inducing penalties to sparse grid subspaces, we derive a new indicator for identifying important functional components.

Formally, the two approaches discussed in this chapter offer an approximate solution for (2.5) when it is infeasible to solve the problem exactly in the sparse grids function space V. In the first case, we approximate V as a sum of smaller spaces $V_1, \ldots, V_k$ with smaller input dimensionalities and perform an inexact minimisation. In the second case, we reduce the function space V keeping the minimisation exact.

We will see that the latter task is significantly more complicated than the first one, which may explain why some professional football players become politicians, but no politician becomes a professional football player.

4.1. Ensemble of Independent Models

In the previous chapter we mentioned that dimensionality reduction can serve as a pre-processing step before model fitting to prevent the curse of dimensionality, although the data-aware methods for dimensionality reduction can become prohibitively expensive. Unlike the dimensionality reduction and manifold learning methods, subspace projections are data-oblivious, which means that they do not rely on data to produce the mapping from a high-dimensional space to a low-dimensional one. Random forests is the paragon of this class of methods.

Subspace projections can be performed in several ways. Commonly, these projection methods aim to preserve certain properties important for the models. For example, they may aim to preserve the distance between the data points or their scalar products. We focus on two particular methods: random projections and feature hashing.

Johnson and Lindenstrauss (1984) showed that any set S of N points embedded in D dimensions can be projected down to $K \in \mathcal{O}(\varepsilon^{-2} \log N)$ dimensions such that the distortion from this projection is not worse than $1+\varepsilon$. This means that for every pair of points $\mathbf{x}, \mathbf{x}'$ from S and a projection matrix $\mathbf{P}$ of size $D \times K$

it holds that

$$\sqrt{\tfrac{K}{D}}\|\mathbf{x}-\mathbf{x}'\|_2(1-\varepsilon) \leq \|\mathbf{P}\mathbf{x}-\mathbf{P}\mathbf{x}'\|_2 \leq \sqrt{\tfrac{K}{D}}\|\mathbf{x}-\mathbf{x}'\|_2(1+\varepsilon). \tag{4.1}$$

The entries of the matrix $\mathbf{P}$ should be independent and identically distributed. The efficient computation of the projection matrices is a subject of an ongoing research. Major efforts focus either on finding very sparse projection matrices (Achlioptas, 2001) or on utilising the matrix structure for a fast matrix-vector product computation (Ailon & Chazelle, 2009).

The property (4.1) is important for the machine learning methods that rely on Euclidean distance as a similarity metric, i.e. k-nearest-neighbours, kernels, or support vector machines.

As the dimensionality of the input space grows, storage of the projection matrix can become a bottleneck. Here, hash functions offer a remedy. Random hashing relies on two hash functions $h : \mathbb{N} \to \{1, \ldots, m\}$ and $\xi : \mathbb{N} \to \{\pm 1\}$ that together define the hashed feature map

$$\psi_i^{(h,\xi)}(\mathbf{x}) := \sum_{j:h(j)=i} \xi(j)x_j.$$

Function h maps the entry j of the original vector to the entry i of the new vector and function ξ decides whether to add or to subtract the entry. If written in matrix notation, the combination of functions h and ξ corresponds to a projection matrix $\mathbf{P}$ where every column has only one nonzero entry and all nonzero entries are either $+1$ or -1.

These feature maps can be used to define an inner product

$$\langle \mathbf{x}, \mathbf{x}' \rangle_\psi := \langle \psi^{(h,\xi)}(\mathbf{x}), \psi^{(h,\xi)}(\mathbf{x}') \rangle.$$

similar to the procedure in Sec. 2.4.4. Weinberger et al. (2009) showed that this construction leads to an unbiased kernel, which means

$$\mathbb{E}_\psi \left[\langle \mathbf{x}, \mathbf{x}' \rangle_\psi \right] = \langle \mathbf{x}, \mathbf{x}' \rangle.$$

For kernel methods, hashing can be integrated into the kernel trick implementation (see Sec. 2.4.4).

Data-oblivious random subspace projections can certainly cause the loss of important information. That is why the supervised learning methods rarely rely on a single projection but rather construct ensembles of numerous simple models from differently projected data and aggregate the predictions as depicted in Fig. 4.1. Rooney et al. (2004) showed that this method can outperform other ensemble building methods such as bagging and boosting.

The complexity overhead of an ensemble is moderate and vastly smaller than the complexity of the high-dimensional model. Figure 4.2 compares the number of grid indices of a regular sparse grid with level 4 and an ensemble of low-

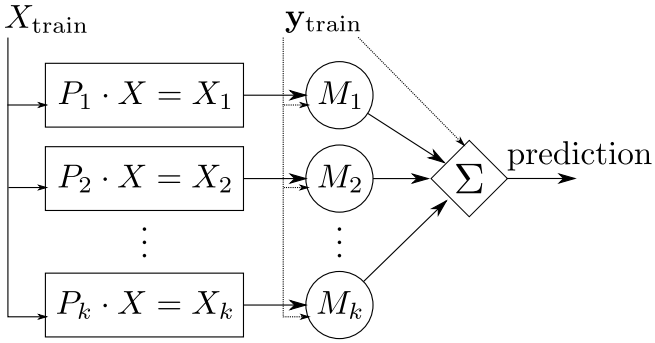


Figure 4.1.: Construction procedure for an ensemble of random projections. A high-dimensional training dataset X_{train} is projected using k independent matrices $P_1, \ldots, P_k$ to obtain lower-dimensional training datasets $X_1, \ldots, X_k$. These datasets together with the target values $\mathbf{y}_{\text{train}}$ are used to fit the models $M_1, \ldots, M_k$. At the prediction phase, the output of the individual models is combined to produce an overall ensemble prediction. Adapted from (Batzner, 2015).

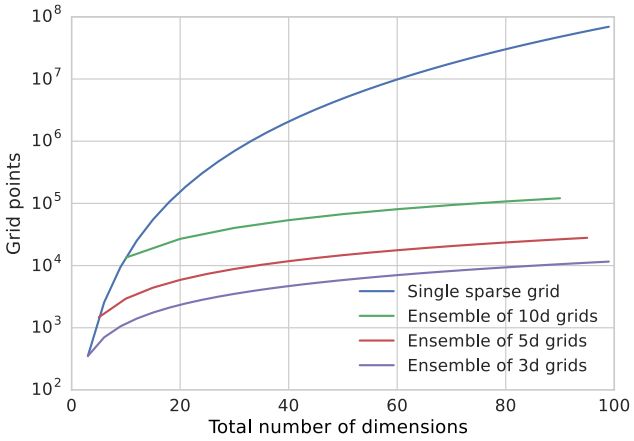


Figure 4.2.: Growth of the total number of grid points for a single sparse grid model and ensembles of sparse grids with fixed dimensionality. All sparse grids have level 4. The number of sparse grid models in an ensemble is the ratio between the total number of dimensions for a single sparse grid model and the number of dimensions of ensemble members, e.g. a 30-dimensional single sparse grid is compared with ensembles of three 10-dimensional, six 5-dimensional, and ten 3-dimensional sparse grids. Single model grows exponentially while ensembles grow linearly.

dimensional sparse grids with level 4 as the problem dimensionality increases. The size of a regular sparse grid grows exponentially while the total number of grid points in an ensemble grows linearly.

Both, random projection and feature hashing, have linear complexity in the number of ambient and projected dimensions. In the bachelor project with Batzner (2015) we observed that the random projection method is faster as long as the projection matrix, which grows linearly with d, can fit into the main memory.

To find the optimal combination of individual models in an ensemble, Wolpert (1992) suggested the *stacking* procedure. For regression ensembles, it results in a convex combination, which means that all combination coefficients are non-negative and add up to 1. These coefficients are chosen in the way that minimises the generalisation error of a previously unseen validation dataset or as a part of a cross-validation procedure:

$$\begin{aligned} &\min_{\alpha_1,\dots,\alpha_K} \sum_{i=1}^{N} \left(y_i - \sum_{k=1}^{K} \alpha_k f_k(\mathbf{x}_i) \right)^2 \\ &\text{subject to } \alpha_k \geq 0,\ k = 1, \dots, K \\ &\sum_k \alpha_k = 1. \end{aligned} \tag{4.2}$$

For classification ensembles, this procedure amounts to a weighted majority voting to obtain the overall prediction, which is the same as convex combination and subsequent thresholding. Breiman (1996) observed that in most cases this scheme leads to a lower prediction error than a single predictor f_k with the lowest cross-validation error.

We observed that these random projection ensembles can work very well for real-world problems. In particular, these ensembles showed the state-of-the-art performance for the internet advertisement prediction problem.[1] The Internet Advertisements dataset is a standard classification benchmark dataset compiled by Kushmerick (1999).[2] It addresses the problem of identification advertisement banners on web pages. The dataset consists of meta information about 3 279 hyperlinked images found on web pages. The meta information is described by 1 557 attributes such as the geometry of the image (if available), the phrases occurring in the URL of the hyperlink, the URL and alt text of the image, the anchor text, and words occurring near the anchor text. The goal is to predict whether an image is an advertisement or not. Only 14% of all data entries are advertisements, which means that if a model predicts that neither of images is an ad, it will be correct 86% of the time. Fradkin and Madigan (2003) studied the influence of dimensionality reduction techniques in the dataset. To keep the

[1] The results for other benchmarks can be found in (Batzner, 2015).

[2] The dataset is available from UCI Machine Learning Repository at `archive.ics.uci.edu/ml`.

Method	Stacking	Method	PCA	RP
SG+RP	95.4	C 4.5	94.7	90.7
SG+FH	95.8	NN	95.8	95.8
SG+PCA	96.1	SVM	96.6	96.5

Table 4.1.: Internet Advertisement benchmark. Sparse grids results on the left are compared to the results from (Fradkin & Madigan, 2003) on the right.

settings comparable, in Table 4.1 (right) we present their results from principle component analysis (PCA) and random projections (RP) onto 500 dimensions.

The sparse grid results presented in Table 4.1 (left) were obtained using ensemble stacking with majority voting for 10 sparse grid regressors with level 3 and 50 dimensions using random projection (SG+RP) and feature hashing (SG+FH) methods. We also compare the results to PCA a singular sparse grid model with level 2 and 500 dimensions (SG+PCA). The computation of a level 3 sparse grid with 500 dimensions would be intractable.

The idea of using ensembles of sparse grids is not completely new. Heinecke, Peherstorfer, Pflüger, and Song (2012) used a procedure called AdaBoost to create an ensemble of weak regular sparse grids that could match the accuracy of a spatially adaptive sparse grid. The major difference is that for Heinecke et al. "weak" means having full dimensionality and reduced level, while in this work "weak" means reduced dimensionality and high level. Besides, the creation of an ensemble using independent random projections bears more potential for parallelisation than inherently sequential AdaBoost.

Although subspace projections may not have a strong mathematical appeal, they constitute a working horse behind numerous machine learning applications where the trade-off between working sufficiently good in most cases and working mediocre in the worst case is made in favour of the former.

4.2. Single Model with Additive Structure

How to predict the price and quality of a Bordeaux wine? All you need to know is the age of the vintage, the average temperature during the growing season, as well as the rainfall in the preceding winter and during harvest months. According to Ashenfelter (2010), the weighted sum of these numbers explains about 80% of the wine price variation.[3] This is only one in a multitude of examples for simple linear regression models used in science and industry.

The secret of success of many linear regression models lies in the property of many real-world problems where most of the variation of the target variable

[3]Ashenfelter (2010) also calculated the rate of return to holding wine at about 2-3% p.a. which vastly surpassed the return rates of German government bonds at the moment of writing.

can be explained by the input variables directly or together with few interactions between the input variables. Sparse grid models are able to preserve this property. This means, in consequence, that for a given problem certain parts of the sparse grid can be omitted without any significant loss of accuracy. This in turn allows to tackle higher-dimensional problems.

We begin this section by showing that every solution for a supervised learning problem—simply every function— possesses an additive structure in form of functional analysis of variance (ANOVA) decomposition (Sec. 4.2.1). This ANOVA decomposition alone does not bring us far. However, using ANOVA decomposition we can formalise the property of a high-dimensional function to be represented as a sum of low-dimensional ones. We call this property an effective dimensionality (Sec. 4.2.2). Finally, we show that when certain sparse grids discretise a function, the parts of the sparse grid model correspond to discretised ANOVA components (Sec. 4.1). This implies that low effective dimensionality of the original problem reflects in low effective dimensionality of the sparse grid model. We build upon this observation in Sec. 4.3.

4.2.1. Functional ANOVA Decomposition and Additive Models

Functional ANOVA decomposition represents a high-dimensional function as a sum of functions of the form

$$\begin{aligned} f(x_1, x_2, \ldots, x_D) &= f_0 + \sum_{i}^{D} f_i(x_i) + \sum_{i<j}^{D} f_{ij}(x_i, x_j) \\ &+ \sum_{i<j<k}^{D} f_{ijk}(x_i, x_j, x_k) + \cdots + f_{1,\ldots,D}(x_1, \ldots, x_D). \end{aligned} \tag{4.3}$$

The first term f_0 is constant, f_i denotes one-dimensional functions, f_{ij} denotes two-dimensional functions—also called second-order interactions—and so on.

Our exposition of the ANOVA decomposition and its relation to sparse grids follows Griebel (2006) and Feuersänger (2010). Let $V^{(D)}$ denote a D-dimensional tensor product function space and $f(x_1, \ldots, x_D) \in V^{(D)}$ be a mapping from $[0,1]^D$ to $\mathbb{R}$. Let μ be a product measure,[4] which means that

$$\mathrm{d}\mu = \prod_{j=1}^{D} \mathrm{d}\mu_j(x_j),$$

[4]Do not be afraid if you missed the lecture on measurable spaces in your curriculum. A measure can be intuitively understood as a generalisation of the concepts of length, area, and volume. It assigns a real number to a subset and should satisfy several properties, such as non-negativity and assignment of 0 to an empty set. Definition of a measure is not unique. We will learn two important measures that allow us to define projections with different properties.

with unit mass such that

$$\int_0^1 \mathrm{d}\mu_j(x_j) = 1 \quad \text{for each } j = 1, \ldots, D.$$

In the one-dimensional case, we are interested in the decomposition of the form

$$f(x) = f_0 + f_1(x) \tag{4.4}$$

with $f_0 \in \mathrm{span}\{1\}$ and $f_1 \in W^{(1)}$. Hence, we decompose $V^{(1)}$ in a constant and a one-dimensional contribution

$$V^{(1)} = \mathbf{1} \oplus W^{(1)}, \tag{4.5}$$

where $\mathbf{1} = \mathrm{span}\{1\}$ denotes the one-dimensional space of constant functions and $W^{(1)}$ denotes its complement to $V^{(1)}$.

We use the projection operator P

$$Pf := \int_{[0,1]} f(x)\mathrm{d}\mu(x) \tag{4.6}$$

to formulate the functions f_0 and f_1 explicitly. It is straightforward to compute the constant as

$$f_0 = Pf(x) = \int_{[0,1]} f(x)\mathrm{d}\mu(x). \tag{4.7}$$

The one-dimensional remainder then becomes

$$f_1(x) = f(x) - f_0 = (I - P)f(x) = f(x) - \int_{[0,1]} f(x)\mathrm{d}\mu(x). \tag{4.8}$$

The measure $\mu(x)$ yields the definition $\langle f, g\rangle := \int_{[0,1]} f(x)g(x)\mathrm{d}\mu(x)$, which determines certain properties of the ANOVA components. In particular, $\langle f_1, f_0\rangle = 0$ with respect to this inner product, since the subspace increment $W^{(1)} = (I - P)V^{(1)}$ is the orthogonal compliment to $\mathbf{1}$ in $V^{(1)}$.

The tensor product construction allows the generalisation of the splitting (4.5) to a D-dimensional case:

$$\begin{aligned} V^{(d)} &= \bigotimes_{i=1}^{D} (\mathbf{1}_i \oplus W_i^{(1)}) \\ &= \mathbf{1}_1 \otimes \ldots \otimes \mathbf{1}_D \\ &\oplus \bigoplus_{i=1}^{D} \mathbf{1}_1 \otimes \cdots \otimes W_i \otimes \ldots \otimes \mathbf{1}_D \\ &\oplus \bigoplus_{i<j} \mathbf{1}_1 \otimes \cdots \otimes W_i \otimes \cdots \otimes W_j \otimes \cdots \otimes \mathbf{1}_D \end{aligned} \tag{4.9}$$

$$\oplus \quad \cdots$$
$$\oplus \quad W_1 \otimes \cdots \otimes W_D. \tag{4.10}$$

The subscripts i and j identify the corresponding coordinate directions.

Using subsets $\mathbf{u} \subseteq \{1, \ldots, D\}$, we can establish a shorthand notation for ANOVA component, where $f_{\mathbf{u}} \in W_{\mathbf{u}}$ and $\mathbf{x}_{\mathbf{u}}$ represents a subvector of $\mathbf{x}$ with components x_i, $i \in \mathbf{u}$. Hence, we can write (4.3) as

$$f(x_1, \ldots, x_D) = \sum_{\mathbf{u} \subseteq \{1,\ldots,D\}} f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}), \tag{4.11}$$

and (4.9) as

$$V^{(D)} = \bigoplus_{\mathbf{u} \subseteq \{1,\ldots,D\}} W_{\mathbf{u}} := \bigoplus_{\mathbf{u} \subseteq \{1,\ldots,D\}} \left(\bigotimes_{k \in \{1,\ldots,D\} \setminus \mathbf{u}} \mathbf{1}_k \right) \otimes \left(\bigotimes_{j \in \mathbf{u}} W_j \right). \tag{4.12}$$

Different measures induce different projections and hence different decompositions. For example, the Lebesgue measure $\mathrm{d}\mu(x) = \mathrm{d}x$ induces

$$Pf = \int_{[0,1]} f(x) \mathrm{d}x$$

and the corresponding classical ANOVA decomposition is well-known in statistics (Efron & Stein, 1981; Wahba, 1990).

The Dirac measure located at a point $a \in [0, 1]$ induces

$$Pf = \int_{[0,1]} f(x)\delta(x - a)\mathrm{d}x = f(a).$$

and the corresponding decomposition is known under the names *anchored* ANOVA decomposition (Sloan et al., 2004; Dick et al., 2004) and cut-HDMR (Rabitz & Alis, 1999).

For a fixed choice of the one-dimensional mapping P_j, the decomposition (4.12) is unique. The empty set projection $P_\emptyset = \prod_{j=1}^{D} P_j$ is the unconditional mean of f with respect to the measure μ, while the partial projections $P_{\mathbf{u}} = \prod_{j \in \mathbf{u}} P_j$ for $\mathbf{u} \neq \emptyset$ describe the conditional means $\int \ldots \int f(\mathbf{x}) \prod_{k \in \{1,\ldots,D\} \setminus \mathbf{u}} \mathrm{d}\mu_k$ that marginalise out the complementary directions $j \in \{1, \ldots, D\} \setminus \mathbf{u}$.

The individual terms $f_{\mathbf{u}}$ in (4.11) can be computed by successive application of projections and subtraction of low order terms (Griebel, 2006):

$$f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}) = \left(\prod_{j \in \{1,\dots,D\} \setminus \mathbf{u}} P_j f \right) - \sum_{\mathbf{v} \subset \mathbf{u}} f_{\mathbf{v}}(\mathbf{x}_{\mathbf{v}}) = \sum_{\mathbf{v} \subseteq \mathbf{u}} (-1)^{|u|-|v|} \left(\prod_{j \in \{1,\dots,D\} \setminus \mathbf{v}} P_j f \right) (\mathbf{x}_{\mathbf{v}}). \tag{4.13}$$

Note that ANOVA components are orthogonal with respect to $\mu(\mathbf{x})$. This means that for every $f_{\mathbf{u}}$ and $f_{\mathbf{v}}$ with $\mathbf{u} \neq \mathbf{v}$ we have

$$\langle f_{\mathbf{u}}, f_{\mathbf{v}} \rangle = \int_{[0,1]^D} f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}) f_{\mathbf{v}}(\mathbf{x}_{\mathbf{v}}) \mathrm{d}\mu(\mathbf{x}) = 0. \tag{4.14}$$

Also, for every component $f_{\mathbf{u}}$ with $\mathbf{u} \neq \emptyset$ we have

$$\int_{[0,1]^D} f_{\mathbf{u}} \mathrm{d}\mu(\mathbf{x}) = 0. \tag{4.15}$$

ANOVA decomposition contains 2^D terms and does not lead to the computation improvement per se. It was observed, however, that for many relevant approximation problems, including data mining, the importance of the ANOVA components decays with the increasing order of interactions (Friedman, 1991). This means that the high-order terms in (4.3) can be omitted without a significant loss of approximation quality.

4.2.2. Effective Dimensionality

For practical considerations, the computation of ANOVA components is only useful if the number of informative components is limited and small. In contrast, if high-order interaction terms carry essential information about the approximant, identification of low dimensional terms yields no benefits. Following Holtz (2011), we introduce the notion of effective dimensionality to characterise how well low-dimensional terms can express an approximant.[5]

Classical ANOVA decomposition allows to bound the approximation error with respect to effective dimensions. Let

$$\sigma^2(f) := \int_{[0,1]^D} \left(f(\mathbf{x}) - \int_{[0,1]^D} f(\mathbf{z}) \mathrm{d}\mathbf{z} \right)^2 \mathrm{d}\mathbf{x} \tag{4.16}$$

[5]Holtz (2011) considers two kinds of effective dimensionality: superposition, where $|\mathbf{u}| \leq d_s$ the interaction order is restricted, and truncated, where $\mathbf{u} \subseteq \{1, \dots, d_t\}$ can include any interaction order but the input space dimensionality is restricted. We only consider the first kind.

be the *total variance* of the function f. This total variance can be decomposed into the variances of individual ANOVA components, since

$$\begin{aligned}
\sigma^2(f) &= \int_{[0,1]^D} (f(\mathbf{x}) - f_0)^2 \,\mathrm{d}\mathbf{x} = \int_{[0,1]^D} f^2(\mathbf{x})\mathrm{d}\mathbf{x} - 2\int_{[0,1]^D} f(\mathbf{x}) f_0 \mathrm{d}\mathbf{x} + f_0^2 \\
&= \int_{[0,1]^D} \left(\sum_{\mathbf{u}\subseteq\{1,\dots,D\}} f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}) \right) \left(\sum_{\mathbf{u}\subseteq\{1,\dots,D\}} f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}) \right) - \\
&\quad 2\int_{[0,1]^D} \sum_{\mathbf{u}\subseteq\{1,\dots,D\}} f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}) f_0 \mathrm{d}\mathbf{x} + f_0^2 \\
&= \sum_{\mathbf{u}\subseteq\{1,\dots,D\}} \|f_{\mathbf{u}}\|_2^2 + \sum_{\mathbf{u}\subseteq\{1,\dots,D\}} \sum_{\substack{\mathbf{v}\subseteq\{1,\dots,D\} \\ \mathbf{v}\neq\mathbf{u}}} \|f_{\mathbf{u}}\|_2^2 - 2 \sum_{\mathbf{u}\subseteq\{1,\dots,D\}} \langle f_{\mathbf{u}}, f_0 \rangle + f_0^2 \\
&= \sum_{\substack{\mathbf{u}\subseteq\{1,\dots,D\} \\ \mathbf{u}\neq\emptyset}} \|f_{\mathbf{u}}\|_2^2 + \sum_{\substack{\mathbf{u}\neq\emptyset, \mathbf{v}\neq\emptyset \\ \mathbf{u}\neq\mathbf{v}}} \langle f_{\mathbf{u}}, f_{\mathbf{v}} \rangle \overset{(4.14)}{=} \sum_{\substack{\mathbf{u}\subseteq\{1,\dots,D\} \\ \mathbf{u}\neq\emptyset}} \|f_{\mathbf{u}}\|_2^2 \\
&\overset{(4.15)}{=} \sum_{\substack{\mathbf{u}\subseteq\{1,\dots,D\} \\ \mathbf{u}\neq\emptyset}} \int_{[0,1]^D} \left(f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}) - \int_{[0,1]^D} f_u(\mathbf{z}_u)\mathrm{d}\mathbf{z} \right)^2 \mathrm{d}\mathbf{x} \\
&= \sum_{\substack{\mathbf{u}\subseteq\{1,\dots,D\} \\ \mathbf{u}\neq\emptyset}} \sigma^2(f_{\mathbf{u}}).
\end{aligned}$$

The *effective dimensionality* d_s describes the maximum interaction order of the ANOVA components necessary to preserve the variance up to a proportion $\alpha \in [0,1]$:

$$d_s := \underset{t\in\{1,\dots,D\}}{\arg\min} \sum_{\substack{|\mathbf{u}|\leq t \\ \mathbf{u}\neq\emptyset}} \sigma^2(f_{\mathbf{u}}) \geq \alpha\sigma^2(f). \tag{4.17}$$

All ANOVA components with interaction order larger than d_s yield only an insignificant improvement of the approximation quality.

Holtz (2011, Lemma 2.3) showed that the *approximation error* from truncation of ANOVA components is bounded by the total variance. This means that for the effective dimensionality d_s with a proportion α we have

$$\|f - \sum_{|\mathbf{u}|\leq d_s} f_{\mathbf{u}}\|_2^2 \leq (1-\alpha)\sigma^2(f). \tag{4.18}$$

A choice of $\alpha = 0.99$ would imply that only 1% of the total variance was lost due to the truncated interaction order.

In contrast, *anchored* ANOVA decomposition yields an error bound for *quadrature*. Let I be a quadrature operator

$$I(f) := \int_{[0,1]^D} f(\mathbf{x}) \mathrm{d}\mathbf{x}.$$

Furthermore, let

$$\tilde{\sigma}(f) := \sum_{\substack{\mathbf{u} \subseteq \{1,\dots,D\} \\ \mathbf{u} \neq \emptyset}} |I f_{\mathbf{u}}| \tag{4.19}$$

denote the sum of absolute values of all anchored ANOVA components.

In the anchored ANOVA case, the *effective dimensionality* d_s describes the maximum interaction order of the ANOVA components required to preserve a proportion $\alpha \in [0, 1]$ of $\tilde{\sigma}(f)$:

$$d_s := \underset{t \in \{1,\dots,D\}}{\arg\min} \sum_{\substack{|\mathbf{u}| \leq t \\ \mathbf{u} \neq \emptyset}} |I f_{\mathbf{u}}| \geq \alpha \tilde{\sigma}(f). \tag{4.20}$$

Holtz (2011, Lemma 2.8) showed that this definition allows to bound the *quadrature error* from truncation of ANOVA components similar to (4.18). For the effective dimensionality d_s with a proportion α we have

$$\left| I(f) - I\left(\sum_{|\mathbf{u}| \leq d_s} f_{\mathbf{u}} \right) \right| \leq (1 - \alpha)\ \tilde{\sigma}(f). \tag{4.21}$$

Altogether, functional ANOVA is a form of function subspace decomposition. It only requires a choice of a suitable measure function (e.g. Lebesgue or Dirac delta function), then the projection operation, the form of individual ANOVA components, and the properties such as effective dimensionality follow automatically. The ability to rely on theoretical properties of ANOVA components is very useful. For example, estimate of effective dimensionality tells a lot about the problem at hand. However, working with measure functions, integrals, and component formulas of the form (4.13) directly is cumbersome and inefficient. Fortunately, sparse grids can help us (of cause they can!) to estimate ANOVA components directly through an appropriate function space discretisation.

4.2.3. Sparse Grids and ANOVA Decomposition

Until this point the discussion of the functional ANOVA decomposition was very general and without any connection to sparse grids. An attentive reader, however, may have noticed the similarity between the function space structures (4.9) and (2.13). This similarity is not coincidental.

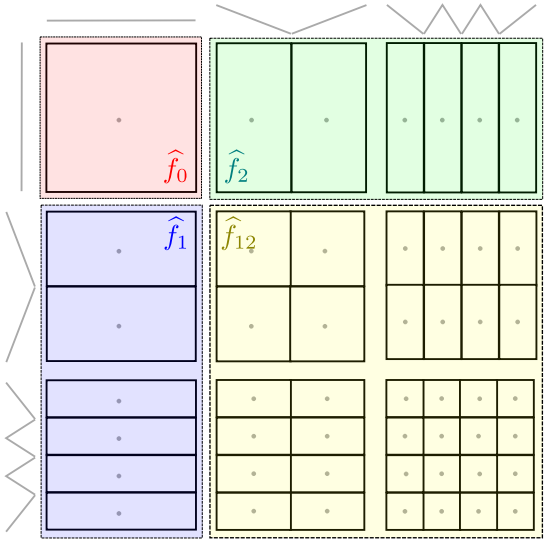


Figure 4.3.: ANOVA components discretised with a hierarchical basis. Basis functions covered in one colour correspond to the same ANOVA component.

If a sparse grid basis set contains a one-dimensional constant function, due to the tensor product construction rule this constant function becomes a factor of some high-dimensional function. Assume that we have the modified linear basis. Then the value of high-dimensional basis functions that contains the constant function $\phi_{1,1}$ in d-th dimension does not depend on the coordinate x_d in d-th dimension:

$$\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) = \phi_{l_1,i_1}(x_1) \times \cdots \times \phi_{l_{d-1},i_{d-1}}(x_{d-1}) \times 1 \times \phi_{l_{d+1},i_{d+1}}(x_{d+1}) \times \cdots \times \phi_{l_D,i_D}(x_D).$$

As Fig. 4.3 suggests, we can then combine all basis functions $\phi_{\mathbf{l},\mathbf{i}}$ with corresponding weights $w_{\mathbf{l},\mathbf{i}}$ that have a constant in the dimension d and not a constant in all other dimensions into a function $\widehat{f}_d$. This combination gives us a part of the total approximant that does not depend on the coordinate x_d in d-th dimension. In particular, for the example in Fig. 4.3 we have

$$\begin{aligned}
\widehat{f}_0(\mathbf{x}) &= 1 \cdot 1, \\
\widehat{f}_1(\mathbf{x}) &= \prod_{(\mathbf{l},\mathbf{i}) \in G_\ell} w_{\mathbf{l},\mathbf{i}} \phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}), \text{ where } l_1 = 1 \text{ and } l_2 > 1, \\
\widehat{f}_2(\mathbf{x}) &= \prod_{(\mathbf{l},\mathbf{i}) \in G_\ell} w_{\mathbf{l},\mathbf{i}} \phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}), \text{ where } l_1 > 1 \text{ and } l_2 = 1, \\
\widehat{f}_{12}(\mathbf{x}) &= \prod_{(\mathbf{l},\mathbf{i}) \in G_\ell} w_{\mathbf{l},\mathbf{i}} \phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}), \text{ where } l_1 > 1 \text{ and } l_2 > 1.
\end{aligned}$$

The sparse grid model $\widehat{f}$ then evaluates to

$$\widehat{f}(\mathbf{x}) = \widehat{f}_0(\mathbf{x}) + \widehat{f}_1(\mathbf{x}) + \widehat{f}_2(\mathbf{x}) + \widehat{f}_{12}(\mathbf{x}),$$

which is identical to the form of the functional ANOVA decomposition (4.3).

The strong connection between the ANOVA decomposition and the sparse grids technique has been known for years. Hegland (2003) motivated dimensional adaptivity of sparse grids for approximation to recover the underlying ANOVA structure. This idea was furthered by Gerstner and Griebel (2003) for high-dimensional quadrature and by Garcke (2004) for machine learning problems. Later, Holtz (2011) and Feuersänger (2010) developed constructive approaches to represent ANOVA components with sparse grids for quadrature and function approximation.

Feuersänger (2010) showed how the use of multiscale basis functions with vanishing zeroth moment (i.e. the second order piecewise linear Neumann prewavelet basis) leads to a natural decomposition of hierarchical subspaces into ANOVA components. Hence, wavelets could be a first choice as sparse grid basis functions. Unfortunately, the efficiency of standard wavelet basis functions, such as those used by Feuersänger, is limited due to growing cost for refinement (Buse, 2015). It turns out that representation of these basis functions requires introduction of so called "shadow points". The number of these auxiliary points grows fast with the refinement level and the dimensionality. Hence, the storage of these points becomes a limiting factor.

For an anchored ANOVA decomposition, it is sufficient to have a hierarchical basis that contains a constant function. For example, the modified linear basis contains a constant function on its first level. Hence, if we use the modified linear basis and the Dirac measure (which means that we evaluate the function at the sparse grid points and use these values to compute the sparse grid coefficients) we obtain sparse grid subspaces that correspond to discretised anchored ANOVA components. Then using (4.20) and (4.21) we could calculate the effective dimensionality and error bounds for quadrature.

Unfortunately, this is useless for our goals for two particular reasons: First, we are unable to evaluate the functions behind most supervised learning problems at an arbitrary point. And second, in supervised learning we are usually interested in approximation of function output and not in computing its quadrature.

Fortunately, if the approximant $\widehat{f}$ is a solution to a regression problem, we can establish an alternative definition of effective dimensionality that allows us to bound the mean squared error from leaving out certain high-order interaction components $\widehat{f}_{\mathbf{u}}$ like in the case of functional ANOVA decomposition.

Similar to the definitions of contributions of the ANOVA components in (4.16) and (4.19), for the dataset $\{\mathbf{x}_i, y_i\}_{i=1}^N$ we define

$$\hat{\sigma}^2(f) := \tfrac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i) \cdot y_i. \tag{4.22}$$

And, consequently, we define effective dimensionality d_s in terms of preserving the α's portion of $\hat{\sigma}^2(f)$:

$$d_s := \underset{t\in\{1,\dots,D\}}{\arg\min} \sum_{|\mathbf{u}|\leq t} \hat{\sigma}^2(f_{\mathbf{u}}) \geq \alpha\hat{\sigma}^2(f). \tag{4.23}$$

Following lemma suggests that using this notion of effective dimensionality we can obtain the error bounds from the removal of superfluous subspaces as we did in (4.18).

Lemma 4.1. *Let $S = \{\mathbf{x}_i, y_i\}_{i=1}^N$ be a dataset. Let $\widehat{f}$ and $\widehat{f}'$ be two functions fitted to S such that $\widehat{f}'$ contains the ANOVA components of $\widehat{f}$ with effective dimensionality d_s. The mean squared error from removing the subspaces above effective dimensionality d_s defined in* (4.23) *is bounded as*

$$\tfrac{1}{N}\sum_{i=1}^{N}\left(y_i - \widehat{f}'(\mathbf{x}_i)\right)^2 - \tfrac{1}{N}\sum_{i=1}^{N}\left(y_i - \widehat{f}(\mathbf{x}_i)\right)^2 \leq (1-\alpha)\hat{\sigma}^2(\widehat{f}). \tag{4.24}$$

Proof. First we show that $\hat{\sigma}^2(\widehat{f})$ can be decomposed in the sum of individual components:

$$\begin{aligned}\hat{\sigma}^2(\widehat{f}) &= \tfrac{1}{N}\sum_{i=1}^{N}\left(y_i\widehat{f}(\mathbf{x}_i)\right) = \tfrac{1}{N}\sum_{i=1}^{N} y_i \sum_{\mathbf{u}\subseteq\{1,\dots,D\}} \widehat{f}_{\mathbf{u}}(\mathbf{x}_i) \\ &= \sum_{\mathbf{u}\subseteq\{1,\dots D\}} \tfrac{1}{N}\sum_{i=1}^{N} y_i\widehat{f}_{\mathbf{u}}(\mathbf{x}_i) = \sum_{\mathbf{u}\subseteq\{1,\dots,D\}} \hat{\sigma}^2(\widehat{f}_{\mathbf{u}}).\end{aligned}$$

If $\widehat{f}$ is fitted to S then its residual is orthogonal to predictions. Therefore, we can rewrite the mean squared error as

$$\begin{aligned}\tfrac{1}{N}\sum_{i=1}^{N}\left(y_i - \widehat{f}(\mathbf{x}_i)\right)^2 &= \tfrac{1}{N}\sum_{i=1}^{N}\left(y_i - \widehat{f}(\mathbf{x}_i)\right)\left(y_i - \widehat{f}(\mathbf{x}_i)\right) \\ &= \tfrac{1}{N}\sum_{i=1}^{N} y_i\left(y_i - \widehat{f}(\mathbf{x}_i)\right) - \tfrac{1}{N}\sum_{i=1}^{N}\widehat{f}(\mathbf{x}_i)\left(y_i - \widehat{f}(\mathbf{x}_i)\right) \\ &= \tfrac{1}{N}\sum_{i=1}^{N} y_i^2 - \hat{\sigma}^2(\widehat{f}).\end{aligned} \tag{4.25}$$

Let $\widehat{f}_{\mathbf{u}}$ and $\widehat{f}'_{\mathbf{u}}$ be the ANOVA components of $\widehat{f}$ and $\widehat{f}'$ correspondingly. Then we have

$$\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \sum_{|\mathbf{u}|\leq d_s}\widehat{f}'_{\mathbf{u}}(\mathbf{x}_i)\right)^2 \leq \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \sum_{|\mathbf{u}|\leq d_s}\widehat{f}_{\mathbf{u}}(\mathbf{x}_i)\right)^2, \tag{4.26}$$

since $\widehat{f}'$ does not contain any other components and selecting a subset of components from $\widehat{f}$ leads to an additional truncation error. If we apply (4.25) to (4.26) we immediately obtain

$$\frac{1}{N}\sum_{i=1}^{N} y_i^2 - \sum_{|\mathbf{u}|\leq d_s}\hat{\sigma}^2\left(\widehat{f}'_{\mathbf{u}}\right) \leq \frac{1}{N}\sum_{i=1}^{N} y_i^2 - \sum_{|\mathbf{u}|\leq d_s}\hat{\sigma}^2\left(\widehat{f}_{\mathbf{u}}\right). \tag{4.27}$$

Now we can how the error bound (4.24):

$$\begin{aligned}
&\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \widehat{f}'(\mathbf{x}_i)\right)^2 - \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \widehat{f}(\mathbf{x}_i)\right)^2 \overset{(4.25)}{=} \frac{1}{N}\sum_{i=1}^{N} y_i^2 - \hat{\sigma}^2(\widehat{f}') - \frac{1}{N}\sum_{i=1}^{N} y_i^2 + \hat{\sigma}^2(\widehat{f}) \\
&\overset{(4.27)}{\leq} \frac{1}{N}\sum_{i=1}^{N} y_i^2 - \sum_{|\mathbf{u}|\leq d_s}\hat{\sigma}^2\left(\widehat{f}_{\mathbf{u}}\right) - \frac{1}{N}\sum_{i=1}^{N} y_i^2 + \hat{\sigma}^2(\widehat{f}) = \sum_{|\mathbf{u}|> d_s}\hat{\sigma}^2(\widehat{f}_{\mathbf{u}}) \\
&= (1-\alpha)\hat{\sigma}^2(\widehat{f}).
\end{aligned}$$

□

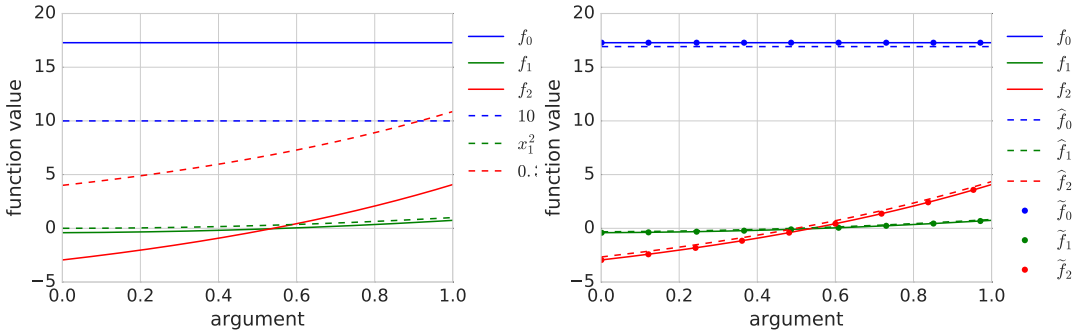
Let us see on a concrete example how the components of a discretised sparse grid relate to the functional ANOVA components. Feuersänger (2010, Fig. 4.4, 4.8) demonstrated that the discretisation of the function

$$f(x,y) = 10 + x_1^2 + 4e^{x_2} + 0.3x_1x_2 \tag{4.28}$$

using prewavelet basis functions recovers correct terms, while the basis set consisting of hat functions and a constant with anchor ANOVA decomposition fails to do so. Using the definition (4.13) we can estimate the ANOVA components exactly[6]

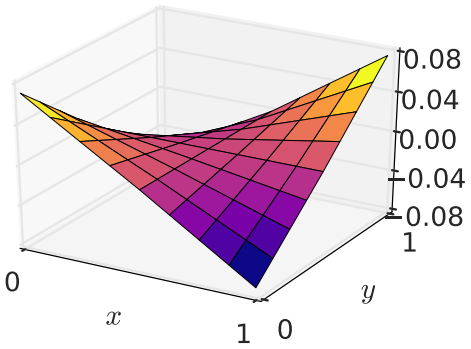
$$\begin{aligned}
f_0 &= 17.28, \\
f_1(x_1) &= x_1^2 + 0.15x_1 - 0.41, \\
f_2(x_2) &= 0.15x_2 + 4e^{x_2} - 6.95, \\
f_{12}(x_1,x_2) &= 0.3x_1x_2 - 0.15x - 0.15x_2 + 0.075.
\end{aligned}$$

[6]The values are rounded to the second significant digit.



(a) Analytical ANOVA

(b) SG discretised ANOVA



(c) SG remainder $\widehat{f}_{12}$

Figure 4.4.: Comparison of analytically computed and numerically estimated ANOVA components for $f(x, y) = 10 + x^2 + 4e^y + 0.3xy$. (a): analytically computed ANOVA components f_0, f_1, f_2 compared with the term of $f(x, y)$, they differ by a constant. (b): analytically computed ANOVA components f_0, f_1, f_2 compared with sparse grid discretisation $\widehat{f}_0, \widehat{f}_1, \widehat{f}_2$ and prewavelet discretisation $\widetilde{f}_0, \widetilde{f}_1, \widetilde{f}_2$, the coincidence is almost exact. (c): the two-dimensional reminder as estimated from the sparse grid discretisation, the amplitude and contribution is negligible compared to the one-dimensional terms.

Figure 4.4a shows that this ANOVA components have the same form as the terms of (4.28) but shifted by a constant.

Now, let us approximate the function (4.28) using a sparse grid with modified linear basis. As we can see in Fig. 4.4b, prewavelet basis functions are perfectly suitable to reconstruct the underlying components of classical ANOVA. In fact, the ANOVA components f_i and prewavelet approximants $\widetilde{f}_i$ are indistinguishable from each other. Surprisingly, our approximants $\widehat{f}_i$ that uses modified linear basis functions are not far away! Finally, modified linear functions succeed to identify the unimportance of the two-dimensional remainder $\widehat{f}_{12}$ shown in Fig. 4.4c.

This example suggests an important conclusion: Although the modified linear basis functions do not possess a vanishing zeros model, in practice they recon-

struct the classical ANOVA components just as well as the prewavelet basis functions. This mean, they could, at least for practical purposes, be applied to identify the importance of the ANOVA components, the effective dimensionality and the bounds for the approximation error.

4.3. Sparse Grids with Additive Structures

Let us know discuss different ways to control the shape of the sparse grid model. This control can be explicit, when the prior knowledge of the problem structure is available or the resolution of higher interaction terms is determined by an external paramete—this is the subject of Sec. 4.3.1. Alternatively, this control can be implicit, identified by putting a penalty on the number of ANOVA components—this is the subject of Sec. 4.3.2. Finally, we conclude this section and this chapter with Sec. 4.3.3 that presents the opportunities for learning a structure in a regression problem using a number of numerical examples.

4.3.1. Integration of Known Structure

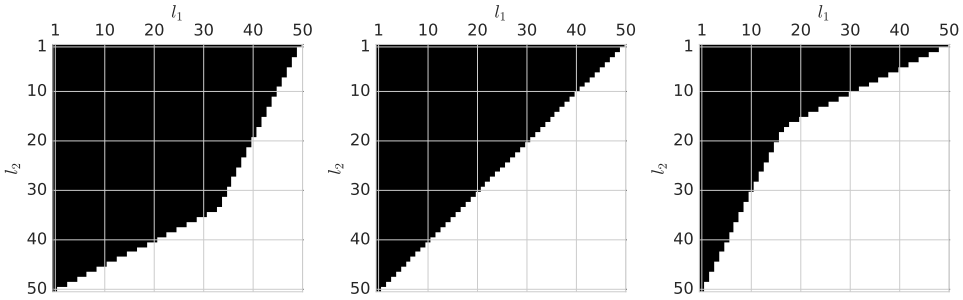
Griebel and Knapek (2000) introduced *generalised sparse grids* that have the flexibility to vary the maximal possible resolution of the interaction terms in a sparse grid.[7] It extends the constraint in the definition (2.17) by limiting the maximum norm of the level vector:

$$G_\ell^T := \bigcup_{\mathbf{l}} G_{\mathbf{l}} \quad \text{such that } |\mathbf{l}|_1 - T|\mathbf{l}|_\infty \leq (\ell + d - 1) - T \cdot \ell. \tag{4.29}$$

The parameter $T \in [-\infty, 1]$ controls the shape of the diagonal and hence the number of degrees of freedom in higher-order interaction terms. Figure 4.5 shows the influence of T on the shape of the subspace tableau. For $T = 0$ we obtain the standard regular sparse grid; $T > 0$ leads to exclusion of hierarchical subspaces from the sparse grid (starting with isotropic ones); $T < 0$ leads to inclusion of new hierarchical subspaces with the special case $T = -\infty$, where it becomes a full grid.

Figure 4.6 compares the number of grid points in different interaction orders for a generalised sparse grid with 5 dimensions and maximum level 5 for different values of the parameter T. Two things are interesting in this diagram. First, for a regular sparse grid most of the points are in the 2-order and 3-order interactions and there are much more points than, for example, in the 1-order or 4-order (as we look on the logarithmic scale). Second, the parameter $T > 0$ quickly reduces the number of points in higher interaction terms, while keeping

[7] The generalised sparse grids introduced by Griebel and Knapek (2000) are not to be confused with the generalised sparse grids which were introduced by Hegland (2003) and denote dimension adaptivity.



(a) $T = -1$ (b) $T = 0$ (c) $T = 1/2$

Figure 4.5.: Subspace tableau for a generalised sparse grid with different values of T from (4.29). For $T = 0$ we obtain the standard regular sparse grid; $T > 0$ leads to exclusion of hierarchical subspaces from the sparse grid (starting with isotropic ones); $T < 0$ leads to inclusion of new hierarchical subspaces with the special case $T = -\infty$, where it becomes a full grid. Adapted from (Griebel & Knapek, 2000).

the form of the distribution same, $T < 0$ explodes the number of points in higher interactions while keeping the distribution form the same.

In addition to limiting the resolution in high-order interaction terms, the domain-specific information can suggest to limit the neighbourhood of allowed interactions. This principle naturally occurs in processing images, such as classification of handwritten digits. Here, the dimension corresponds to a pixel or a superpixel (colour patch) in the images. To recognise a shape in the image, it is therefore more import to consider the information in the local neighbourhood of every pixel and not the information of the other side of the image. In terms of sparse grids this means that certain dimensions (pixels) x_i and x_j do not interact with each other and the component $\widehat{f}_{ij}(x_i, x_j)$ and can be excluded from the sparse grid structure.

Figure 4.7b shows the growth of a 64-dimensional sparse grid model for recognition of 8×8 pixel digits (e.g. Fig 4.7a). The interaction terms are truncated according to the Chebyshev distance between pixels. For example, pixel (1,1) has Chebyshev distance 1 with pixels (0,0), (0,1), (0,2), (1,0), (1,2), (2,0), (2,1), (2,2) and hence the interaction terms between corresponding directions are included in the neighbourhood size 1 in Fig. 4.7b. In contrast, pixel (1,3) has Chebyshev distance 2 and hence the interaction between (0,0) and (1,3) is not included in the bars with neighbourhood size 1 but it is included into the bars with neighbourhood size 2. The advantage of limiting the neighbourhood size increases with the maximum level of the sparse grid model: for example, for a sparse grid level 3 the neighbourhood 7 is about 7 times larger than neighbourhood 1, for a sparse grid level 5 this difference is 452 times.

If the structure of the problem domain is known a priori, sparse grids offer mechanisms to tinker the model structure reducing its size by orders of

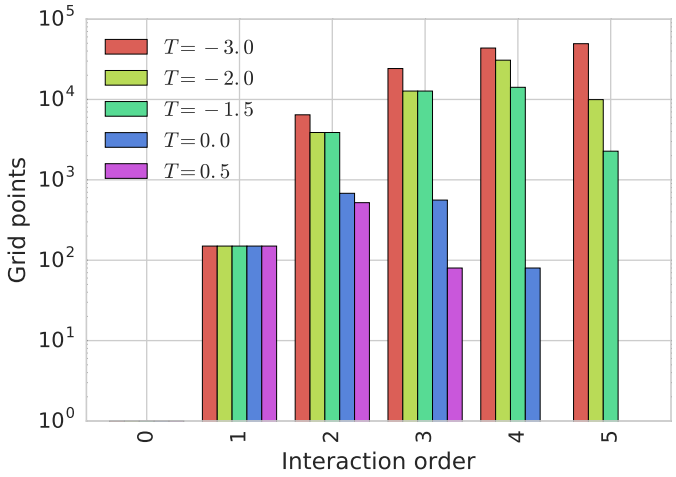
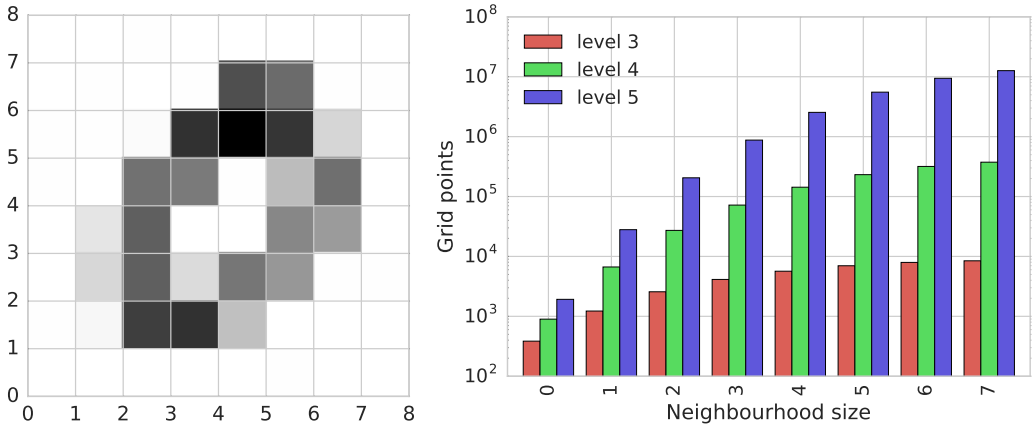


Figure 4.6.: Number of grid points in different interaction orders for generalised sparse grids. For a regular sparse grid with $T = 0$ most of the points concentrate in the 2-order and 3-order interactions. For $T > 0$ the number of points in higher interaction terms diminishes. For $T < 0$ the number of points in higher interactions explodes. The shape of the point distribution remains the same.



(a) Rasterised handwritten digit 0 (b) Sparse grids with limited interactions

Figure 4.7.: Growth of a sparse grid model for recognition of 8×8 pixel handwritten digits. The correlation between collocated pixels is more significant than between far apart. Right: the size of a sparse grid model where only interaction between collocated pixels is considered, the radius of the collocation is denoted as the neighbourhood size. The advantage of limiting the neighbourhood size increases with the maximum grid level.

magnitude. This can facilitate solution solution of high-dimensional problems. However, often this structure is unknown a priori but there is plenty of data. In this case, we should try to identify the structure a posteriori.

4.3.2. Identification of Unknown Structure

Our methods for identifying an additive structure in the problem resemble the methods for identifying low-dimensional manifolds in Chapter 3. Two main classes discussed in this section are greedy approaches and sparsity-inducing norms applied on the level of ANOVA components.

ANOVA greedy approach

The idea of using dimension-adaptive sparse grids to recover the ANOVA structure was initially suggester by Hegland (2003) and Garcke (2004) for data mining and by Gerstner and Griebel (2003) for quadrature computation. The suggested algorithms can be classified as forward greedy selection methods that operate on complete hierarchical subspaces $W_{\mathbf{l}}$ and respect the lattice structure of the solution (you can only add a hierarchical space if all its parents in all dimensions are already added). Usually an algorithm would start with an initial grid as small as consisting only of the subspace $W_{(1,\dots,1)}$. The algorithm would maintain two sets of subspaces:

1. *active set* consisting of hierarchical subspaces that have been made a part of the model and

2. *candidate set* consisting of subspaces to select the new subset from.

The algorithm would identify a hierarchical subspace with the largest error indicator (application dependent) and add it to the active set. The children subspaces of this one are added to the candidate set if all their hierarchical ancestors are already in the active set. Since candidate subspaces are not necessarily orthogonal, the error indicators of the subspaces in the candidate set need to be recomputed with respect to the new model. The iteration continues until some convergence criterion is reached, i.e. training error surpasses a certain threshold.

Numerical experiments conducted by Hegland (2003) showed that this methods can be successfully used for identification of ANOVA components. Alas, this forward selection method on hierarchical subspaces inherits the drawbacks described in Sec. 3.2. Moreover, it can be computationally intractable for higher dimensions.

Feuersänger (2010) suggested a spatially-dimension-adaptive algorithm for function approximation that respects the ANOVA structure of the sparse grid model. The procedure starts with an initial grid $G^{(0)}$ of "small, but not too small, level". The algorithm starts with a backward-greedy procedure where it

fits the initial grid model to the problem, identifies the importance of different ANOVA components in the grid model by summarising the local error indicators of the grid-indices, and eliminates the basis functions with the indicator below a certain threshold value. After this coarsening, the algorithm continues with a standard spatially-adaptive refinement procedure where only children in the same ANOVA component are added. In practice, this means that, if the level vector of a grid index is 1 in a certain dimension, children in this dimension are not created.

By default the algorithm would refine all grid indices with the local error indicator larger than a certain threshold. This makes it a refinement in a breath-first-search manner of traversion through the hierarchical subspaces scheme in contrast to the depth-first-search manner used by Garcke and Hegland.

We can see three problems with this approach:

1. The requirement to consider all important main influences and the interaction terms right from the beginning makes the initialisation step extremely expensive in high dimensions. The failure to make correct assumptions about the choice of interaction order cannot be corrected in the later stage. For example, a regular sparse grid of level level l would be able to capture at most the interaction terms of up to order $l-1$ and the highest orders would have only very few grid points. On the other side, a too generous estimation of the maximum interaction order explodes the size of the initial grid making the coarsening step intractable.

2. The choice of a threshold, either absolute or relative, is critical for the success. Either the same threshold controls refinement and coarsening and makes the whole algorithm more crucially depend on the proper choice, or we have two different thresholds and the cross-validation procedure becomes much more expensive. In fact, nothing in the compression step suggests that *any* of the ANOVA components will be completely eliminated as even within the same ANOVA component the volume of the basis function can vary strongly. It seems to be more reasonable to identify the summarised error indicator of the complete ANOVA component and then decide whether to eliminate it or not.

3. Finally, the breath-first-search procedure for refinement seem to be overly lavish for high-dimensional setting. Spending too many grid indices where it is not necessary is exactly what we want to avoid for high-dimensional supervised learning problems. The author reasonably argues that breath-first-search manner reduces the number of overall iterations. However, this strategy heavily relies on the assumption that the algorithm succeeds at choosing the refinement threshold that selects only the right grid indices.

The methods to deal with ANOVA structure suggested so far belong to forward and backward greedy procedures. And similarly to the greedy approach in

Sec. 3.2, identification of the additive subcomponents of the model can occur by greedy insertion and deletion of the ANOVA components of a sparse grid model. The main difference is that instead of considering individual grid indices, where evaluation is comparably inexpensive, we operate with sets of dozens or hundreds of grid indices. Hence, it may be too expensive to include and remove again the subspaces as easy as individual grid indices in Alg. 8.

ANOVA sparsity-inducing norms

The application of sparsity-inducing norms to facilitate ANOVA structure for nonparametric multivariate structure estimation was studied in the context smoothing splines (see a comprehensive review by Gu (2013)). The idea of smoothing spline ANOVA (SS-ANOVA) goes back to the work by Wahba et al. (1995). In the later study, H. H. Zhang et al. (2004) combined SS-ANOVA and basis pursuit to identify the main one-dimensional ANOVA components and two-dimensional interactions. Their approach uses ℓ_1-norm as the regularisation operator and two kinds of regularisation parameters λ_π and λ_s to differentiate between one-dimensional and two-dimensional ANOVA components. This allows a fine control over regularisation, but makes the hyperparameter configuration more time-consuming.

An alternative ℓ_1-norm regularisation of lasso is to a mixed ℓ_1/ℓ_q-norm group lasso regularisation (3.18). The formulation is easily extendable to sparse grid ANOVA context. We can summarise all basis functions that belong to the same ANOVA component in a group and penalise all corresponding parameters together.

Let $P = \mathcal{P}(\{1, \ldots, D\})$ be a powerset, $\mathbf{u} \in P$, $G_\mathbf{u} = \{(\mathbf{l}, \mathbf{i}) \mid \mathbf{l}_j > 1 \text{ for each } j \in \mathbf{u},\ \mathbf{l}_j = 1 \text{ for each } j \notin \mathbf{u}\}$, and $\mathbf{w}_\mathbf{u} = \{w_g \mid g \in G_\mathbf{u}\}$. Then the penalisation term has the form

$$\Omega(\mathbf{w}) = \sum_{\mathbf{u} \in P} \gamma_\mathbf{u} \|\mathbf{w}_\mathbf{u}\|_q. \tag{4.30}$$

Depending on the method, the normalisation term $\gamma_\mathbf{u}$ is set either to $\sqrt{|G_\mathbf{u}|}$ or to $\lambda / \|\mathbf{w}_\mathbf{u}^{\text{ls}}\|_q$, where $\mathbf{w}_\mathbf{u}^{\text{ls}}$ is the simple least-squares solution without any regularisation terms.

Using a sparsity-inducing norm to recover ANOVA structure in sparse grids offers many advantages. It makes the initial coarsening procedure unnecessary. Important or unimportant ANOVA components can be identified at any step of the learning procedure. Since it can substitute the existing regularisation term in (2.6), it does not introduce any additional hyperparameters that need to be estimated using expensive cross-validation procedure. As the next section shows, some sparsity-inducing norms are more robust to the choice of λ than others.

4.3.3. Numerical Experiments

Now it is time to put our methods for ANOVA identification to a test. We take a well studied benchmark datasets from (Friedman, 1991) and compare three different sparsity-inducing penalties for their abilities to identify important ANOVA components and the stability of the methods to the choice of regularisation parameters.

Friedman 1 is a ten-dimensional dataset where only first five dimensions contribute to the target variable:

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \varepsilon \tag{4.31}$$

with uniformly distributed $x_1, \ldots, x_{10} \sim \mathcal{U}(0,1)$ and a normally distributed white noise $\varepsilon \sim \mathcal{N}(0,1)$.

Friedman 2 is generated using a four-dimensional input as

$$y = \sqrt{x_1^2 + \left(x_2 x_3 - \frac{1}{x_2 x_4}\right)^2} + \varepsilon \tag{4.32}$$

with $x_1 \sim \mathcal{U}(0,100), x_2 \sim \mathcal{U}(40\pi, 560\pi), x_3 \sim \mathcal{U}(0,1), x_4 \sim \mathcal{U}(1,11)$, and a comparably high random noise $\varepsilon \sim \mathcal{N}(0, 15\,625)$.

Friedman 3 is another four-dimensional dataset with inputs distributed identically to Friedman 2 and target values generates as

$$y = \arctan\left(\frac{x_2 x_3 - \frac{1}{x_2 x_4}}{x_1}\right) + \varepsilon \tag{4.33}$$

with a low random noise $\varepsilon \sim \mathcal{N}(0, 0.01)$.

Friedman (1991) compared different noise settings for each dataset. In this evaluation we follow the configuration used in (Pflüger, 2010). Friedman 1 dataset contains 10 000 points while Friedman 2 and Friedman 3 contain 5 000 data points. We are not interested in the test error and so we do not control for it.[8] We are rather interested in testing three regularisation penalties: ℓ_1-norm (lasso), ℓ_2^2-norm, and group lasso for their ability to identify important ANOVA components. While ℓ_2^2-norm has been used for sparse grid regression for years and today is a quasi state-of-the-art regulariser, lasso and group lasso are two new contestants that are expected to produce sparser solutions. Lasso achieves

[8]Note that the variance of the random noise directly implies the optimal MSE that is achievable on the dataset. For example, with our instance of Friedman 2 dataset it is virtually impossible to train a model to achieve test error lower than 15 625, training error that is significantly lower than 15 625 would indicate possible overfitting.

this by forcing individual coefficients to 0, while group lasso is configured in a way to regularise all coefficients from the same ANOVA component together.

For the *Friedman 1* we train a ten-dimensional sparse grid model with level 4. As we know that the effective dimensionality of the dataset is equal to 2, we restrict the interaction order to 2 as well. This alone immediately reduces the grid size from 2001 to 1041 points.

Figure 4.8 compares the contributions of individual ANOVA subspaces for different norms with respect to the choice of different values for the parameter λ. Both lasso and group lasso succeed at identifying important ANOVA subspaces (all left of f_5). In contrast, ℓ_2^2-norm regulariser tends to shrinking all contributions such that it is impossible to find a threshold that could separate important components from unimportant. Moreover, in the list of 10 components with the highest contributions we do not find the component f_5 which, we know, plays an important role in the true function.

For ℓ_1-norm penalty we observe a significant drop in contribution between important components (left of f_5) and unimportant (right of it). For $\lambda > 0.001$ the unimportant contributions go to zero. Even for largest tested regularisation parameter $\lambda = 0.1$ the contribution of all important components is above zero such that we do not miss any important ANOVA components.

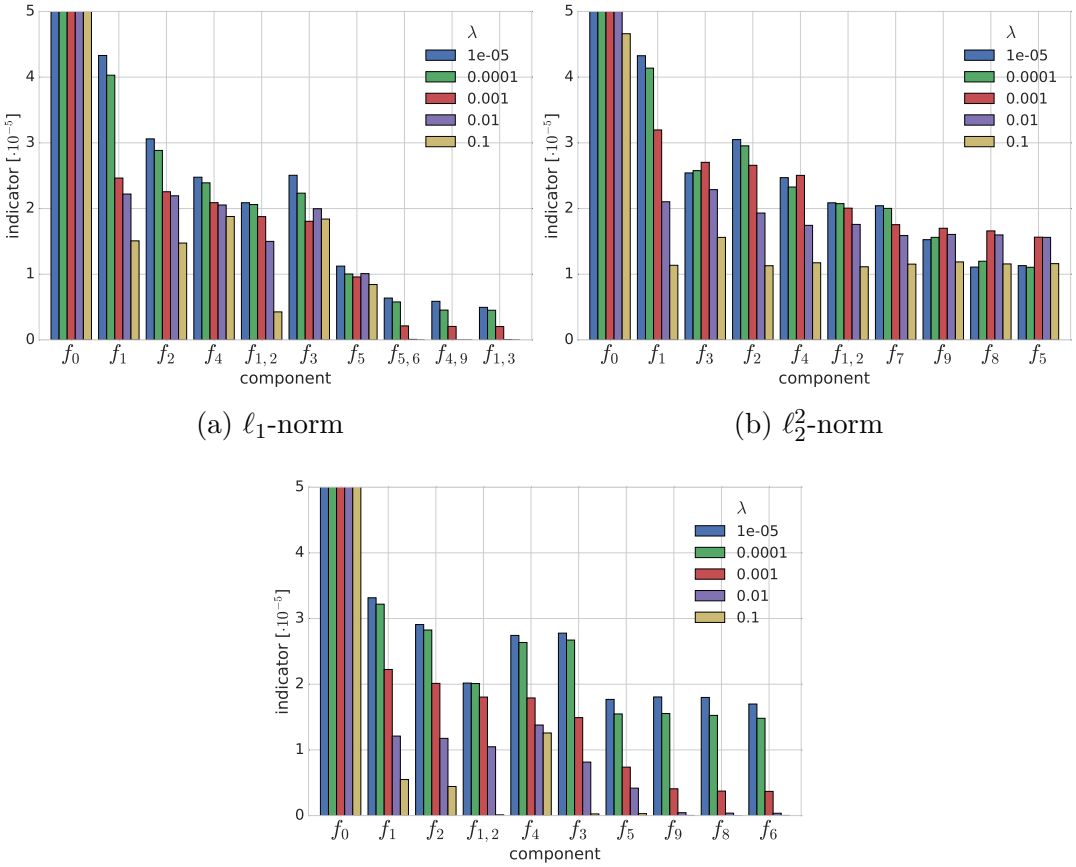
For group lasso this drop can be observed for $\lambda \geq 0.001$ and for $\lambda = 0.01$ the contribution of unimportant components goes to zero. However, as λ continues to increase an becomes 0.1, even important components (right of f_4) becomes zero. This could result in a false negative error.

For Friedman 2 dataset we do not know unimportant ANOVA components a priori. However, the application of all three penalties results in identification of third-order interactions as insignificant (see Fig. 4.9). We order the components with respect to their contributions for $\lambda = 0.001$ and, surprisingly, this order is very stable independently of the selected penalty. As 0.001 implies a relatively high regularisation, the influence of the regularisation is not insignificant and the stability of this order is an evidence that our methods of measuring contribution makes sense and that regularisations make a desirable impact.

The choice of the regularisation parameter implies the relative order of the contributions more strongly for ℓ_2^2 than for other regularisation penalties: We can see that for lasso and group lasso the contributions strongly monotonically decrease with increasing λ, at the same time the order of components does not change significantly. In contrast, for ℓ_2^2-norm the general trend is declining but without monotonicity (see, e.g. the increase in contributions for f_4 or the strong change for the contributions for $f_{1,4}$).

We also observe that group lasso is more prone to false positives for larger values of λ. For example for $\lambda > 0.01$ all components right of f_2 are basically excluded. This is not necessarily a reason for abandoning this penalty, it is rather a caution to be careful with the choice of λ.

For Friedman 3 dataset (see Fig. 4.10) we can again confirm that stable ordering of more important components for all three regularisation methods. Sur-

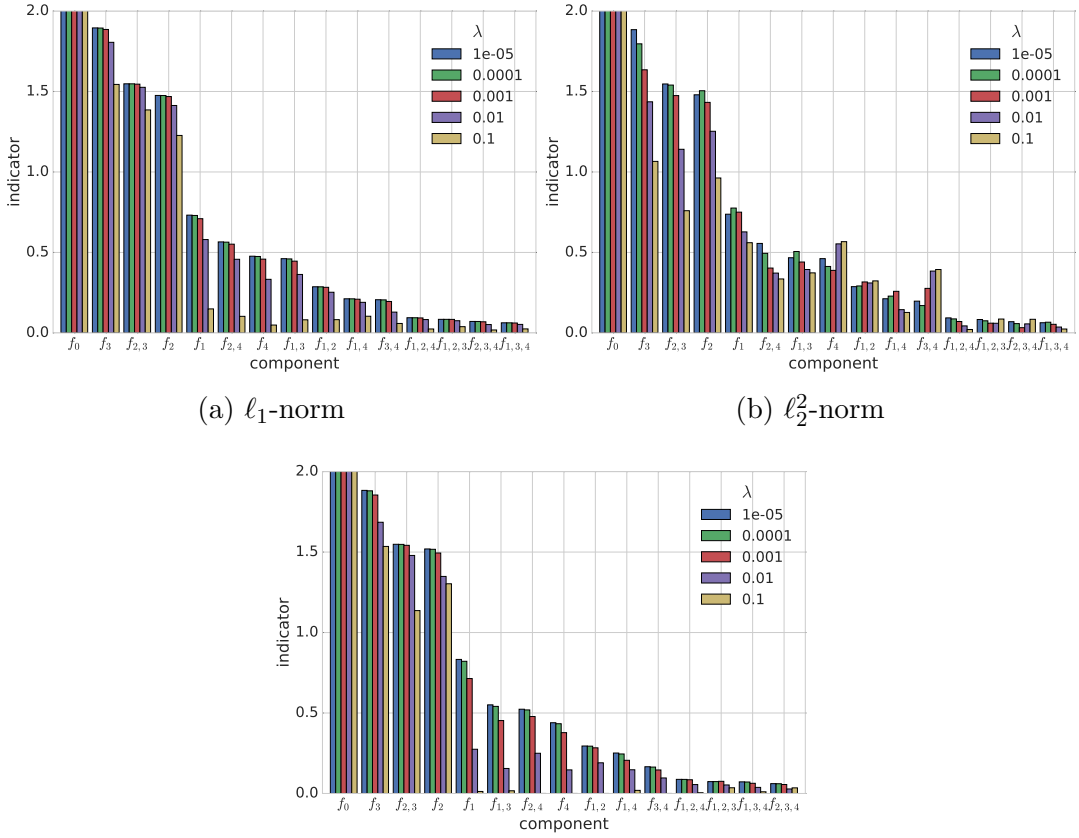


(a) ℓ_1-norm

(b) ℓ_2^2-norm

(c) ℓ_1/ℓ_2-norm

Figure 4.8.: Penalty term comparison for Friedman 1. The indicator interval is restricted to $[0, 5 \cdot 10^{-5}]$, while the maximum indicator value for f_0 is ca. $2 \cdot 10^{-4}$. The penalties produce similar errors for the corresponding values of the parameter λ. Both lasso and group lasso succeed at identifying important ANOVA subspaces (all left of f_5). In contrast, ℓ_2^2-norm regulariser tends to shrinking all contributions and misses the component f_5.

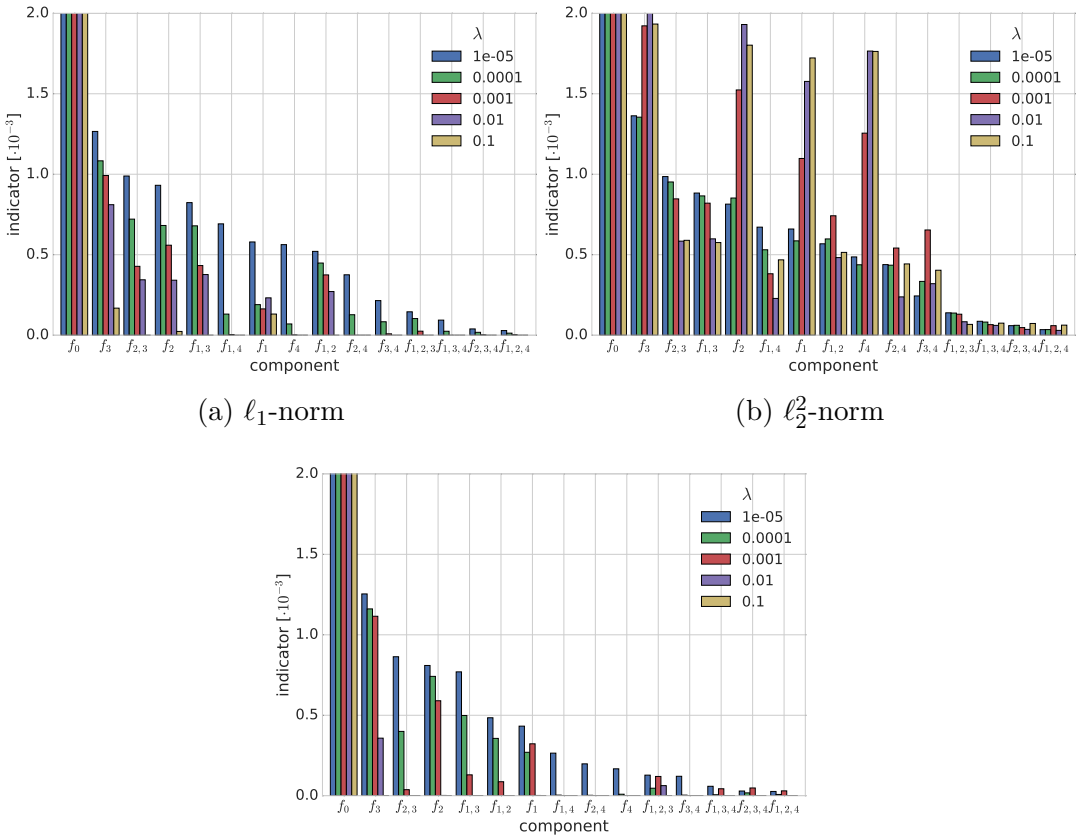


(a) ℓ_1-norm

(b) ℓ_2^2-norm

(c) ℓ_1/ℓ_2-norm

Figure 4.9.: Penalty term comparison for Friedman 2. The indicator interval is restricted to $[0, 2]$, while maximum indicator value for f_0 is ca. 6. The penalties produce similar errors for the corresponding values of the parameter λ. The order of components is similar for all penalties which implies the significance of the results. Three-dimensional ANOVA components f_{ijk} are marginal in all three cases. Lasso and group lasso produce stable monotonically declining sequences of components while results from the ℓ_2^2-norm are noisy.



(a) ℓ_1-norm

(b) ℓ_2^2-norm

(c) ℓ_1/ℓ_2-norm

Figure 4.10.: Penalty term comparison for Friedman 3. The indicator interval is restricted to $[0, 0.002]$, while the maximum indicator value for f_0 is ca. 0.02. The penalties produce similar errors for the corresponding values of the parameter λ. The order of components is similar for all penalties which implies the significance of the results. Lasso and group lasso produce stable monotonically declining sequences of components while results from the ℓ_2^2-norm are noisy.

prisingly, the diagrams shows that the contribution of the constant part is overwhelmingly high, while one-dimensional contributions and two-way interactions are significantly small and three-way interactions are basically non-existent. We cannot conclude that one-dimensional components are always more important than two-way interactions, but three-way interactions can be safely ignored! The differences in the results between ℓ_1-, ℓ_2^2-, and ℓ_1/ℓ_2-norms are identical to those from Friedman 2 dataset.

We learned that additive structure can come from properties of a particular problem, as in the case of ANOVA decomposition, or be imposed artificially to facilitate the processing of high-dimensional data, as in the case of ensembles with random projections. Depending on a particular problem, both methods can be used to solve high-dimensional supervised learning problems with sparse grids.

Prior knowledge or specific properties of the problem domain can be integrated directly into the sparse grid structure, using the generalised sparse grids technique or explicitly reducing the order of allowed interactions in the shape of a sparse grid. If the prior knowledge is unavailable, the sparsity-inducing regularisation penalties with a group structure can facilitate the identification of the ANOVA components.

Besides theoretical aspects of problem decomposition there are practical aspects of efficient fitting additive models. How can we train the football players, from the beginning of the chapter, to act as a team? We direct our attention to this problem next.

5. Parallel Fitting of Models with Additive Structure

> *How can intelligence emerge from nonintelligence? [...] you can build a mind from many little parts, each mindless by itself. I'll call "Society of Mind" this scheme in which each mind is made of many smaller processes. These we'll call agents. Each mental agent by itself can only do some simple thing that needs no mind or thought at all. Yet when we join these agents in societies—in certain very special ways—this leads to true intelligence.*
>
> — Marvin Minsky, *Society of Mind*

Nowadays, the importance of high-performance data analysis and distributed machine learning algorithms cannot be overstated. And so our study of sparse grid methods for high-dimensional data mining would be incomplete without a discussion of parallel algorithms for fitting sparse grid models.

In recent years, a variety of parallel algorithms for sparse grid learning emerged in and around our research group at the Technische Universität München. Heinecke (2014) conjured up a number of optimised and parallelised operations for spatially adaptive sparse grids for commodity hardware as well as for accelerators. Similarly, Buse (2015) developed optimised and parallelised algorithms for dimensionally adaptive sparse grids utilising the subspace scheme. In a recently published paper, Pfander, Heinecke, and Pflüger (2016) developed a new algorithm for spatially-adaptive sparse grids that exploits the subspace scheme for efficiency.

In contrast to their work, here we are not concerned with parallelising the evaluation of a sparse grid model or with the implementation of a distributed conjugate gradient solver for the normal equation (2.33). Instead, we investigate alternative problem formulations that arise from a sparse grid model decomposition. We are interested in the theoretical and empirical properties of these formulations and in the distributed algorithms to solve these problems.

In this chapter we consider two such formulations in combination with the corresponding fitting procedures. The first one, described in Sec. 5.1, comes from statistics and is used for solving a system of smoothing equations. It can be used for differentiable refularisation penalties, such as ℓ_2^2. To solve this problem we develop a new parallel Krylov solver. The second one, discussed in Sec. 5.2, appears in machine learning and optimisation of separable functions. It is called Alternating Directions Method of Multipliers (ADMM) and can be

used for the regularisaton penalties with known proximal operators. We derive two parallel implementations for ADMM and discuss the implication of using ADMM on the hierarchical structures of sparse grids.

5.1. Solving Additive Smoothing Equations

In Sec. 2.4.5 we introduced linear smoothers. The definition of a linear smoother can be generalised to additive models. In fact, many different nonparametric models can be generalised to the linear smoother formulation. It is therefore no surprise that efficient solutions for this generalisation are highly interesting and influential. In this section we suggest a new algorithm for fitting additive smoothers, analyse its potential for improvement through preconditioning, and develop a parallel version of the algorithm.

We are going to focus on the minimisation problem

$$\min_{\mathbf{w}} \|\mathbf{\Phi w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{\Gamma w}\|_2^2 \qquad \text{(2.76 revisited)}$$

introduced in Sec. 2.3, but first let us define some additional notation used throughout this chapter.

In the last chapter we considered sparse grid models with an additive structure of the form

$$f(\mathbf{x}) = f_0 + \sum_i f_i(x_i) + \sum_{i \neq j} f_{ij}(x_i, x_j) + \cdots . \qquad \text{(4.3 revisited)}$$

For convenience of the notation and more general results, in this chapter we consider models in a more general form

$$f(\mathbf{x}) = \sum_{j=1}^{m} f_j(\mathbf{x}), \tag{5.1}$$

where the sum is taken over m D-dimensional functions f_j. Obviously, (4.3) is a special case of (5.1). Furthermore, we change the order of the basis function in the sparse grid model—and, hence, the columns in the matrices $\mathbf{\Phi}$ and $\mathbf{\Gamma}$—such that the basis functions from the same function term f_j appear next to each other. This allows us to identify the block-matrices and block-vectors that correspond to the individual components: $\mathbf{\Phi} = [\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_m]$, $\mathbf{w} = [\mathbf{w}_1, \ldots, \mathbf{w}_m]$, and the matrix $\mathbf{D}$, the result of $\mathbf{\Gamma}^T\mathbf{\Gamma}$, which can be expressed as $\operatorname{diag}(\mathbf{D}_1, \ldots, \mathbf{D}_m)$. The evaluation of the complete model $f(\mathbf{x})$ and of functions $f_j(\mathbf{x})$ at all points in the dataset can then be expressed in a vector form as

$$\mathbf{f} := \mathbf{\Phi w},\ \mathbf{f}_j := \mathbf{\Phi}_j \mathbf{w}_j \text{ such that } \mathbf{f} = \sum_{j=1}^{m} \mathbf{f}_j. \tag{5.2}$$

Similar to the definition of the smoother matrix (2.89), we define the partial smoother matrices as

$$\mathbf{S}_j := \mathbf{\Phi}_j(\mathbf{\Phi}_j^T\mathbf{\Phi}_j + \lambda\mathbf{D}_j)^{-1}\mathbf{\Phi}_j^T, \text{ for } j = 1, \ldots, m. \tag{5.3}$$

With this new notation we can rewrite the normal equation (2.33) as

$$\begin{bmatrix} \mathbf{\Phi}_1^T\mathbf{\Phi}_1 + \lambda\mathbf{D}_1 & \mathbf{\Phi}_1^T\mathbf{\Phi}_2 & \cdots & \mathbf{\Phi}_1^T\mathbf{\Phi}_m \\ \mathbf{\Phi}_2^T\mathbf{\Phi}_1 & \mathbf{\Phi}_2^T\mathbf{\Phi}_2 + \lambda\mathbf{D}_2 & \cdots & \mathbf{\Phi}_2^T\mathbf{\Phi}_m \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{\Phi}_m^T\mathbf{\Phi}_1 & \mathbf{\Phi}_m^T\mathbf{\Phi}_2 & \cdots & \mathbf{\Phi}_m^T\mathbf{\Phi}_m + \lambda\mathbf{D}_m \end{bmatrix} \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_m \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}_1^T\mathbf{y} \\ \mathbf{\Phi}_2^T\mathbf{y} \\ \vdots \\ \mathbf{\Phi}_m^T\mathbf{y} \end{bmatrix}. \tag{5.4}$$

By multiplying both sides of this equation from left with the block-diagonal matrix

$$\begin{bmatrix} \mathbf{\Phi}_1\left(\mathbf{\Phi}_1^T\mathbf{\Phi}_1 + \lambda\mathbf{D}_1\right)^{-1} & & & \\ & \mathbf{\Phi}_2\left(\mathbf{\Phi}_j^T\mathbf{\Phi}_2 + \lambda\mathbf{D}_2\right)^{-1} & & \\ & & \ddots & \\ & & & \mathbf{\Phi}_m\left(\mathbf{\Phi}_m^T\mathbf{\Phi}_m + \lambda\mathbf{D}_m\right)^{-1} \end{bmatrix}$$

we obtain the transformed equation

$$\begin{bmatrix} \mathbf{\Phi}_1 & \mathbf{S}_1\mathbf{\Phi}_2 & \cdots & \mathbf{S}_1\mathbf{\Phi}_m \\ \mathbf{S}_2\mathbf{\Phi}_1 & \mathbf{\Phi}_2 & \cdots & \mathbf{S}_2\mathbf{\Phi}_m \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_m\mathbf{\Phi}_1 & \mathbf{S}_m\mathbf{\Phi}_2 & \cdots & \mathbf{\Phi}_m \end{bmatrix} \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_m \end{bmatrix} = \begin{bmatrix} \mathbf{S}_1\mathbf{y} \\ \mathbf{S}_2\mathbf{y} \\ \vdots \\ \mathbf{S}_m\mathbf{y} \end{bmatrix}.$$

Finally, we transform it to the additive smoothing equation. This means that instead of $\mathbf{w}_j$ being an unknown, our unknown is $\mathbf{f}_j = \mathbf{\Phi}_j\mathbf{w}_j$:

$$\underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{S}_1 & \cdots & \mathbf{S}_1 \\ \mathbf{S}_2 & \mathbf{I} & \cdots & \mathbf{S}_2 \\ \vdots & & & \\ \mathbf{S}_m & \mathbf{S}_m & \cdots & \mathbf{I} \end{bmatrix}}_{=:\tilde{\mathbf{S}}} \underbrace{\begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_m \end{bmatrix}}_{=:\tilde{\mathbf{f}}} = \underbrace{\begin{bmatrix} \mathbf{S}_1\mathbf{y} \\ \mathbf{S}_2\mathbf{y} \\ \vdots \\ \mathbf{S}_m\mathbf{y} \end{bmatrix}}_{=:\mathbf{c}}. \tag{5.5}$$

This transformation gives us an alternative view on the regression problem: the solution is a number of predictions $\mathbf{f}_1, \ldots, \mathbf{f}_m$, such that together they give an overall prediction $\mathbf{f} = \mathbf{f}_1 + \cdots + \mathbf{f}_m \approx \mathbf{y}$ and every individual prediction is a smoothed version of the residual $\mathbf{f}_j = \mathbf{S}_j(\mathbf{y} - \sum_{k\neq j}\mathbf{f}_k)$.

This alternative view also gives us new learning procedures described in Sections 5.1.1 and 5.1.2. Yet, we still can recover the internal model parameters without additional costs. Section 5.1.3 describes how to do it. In Sections 5.1.4

and 5.1.5 we consider in detail the ways how to parallelise and accelerate the new learning procedure.

5.1.1. Backfitting Procedure

Traditionally, problem (5.5) is solved using the *backfitting procedure* suggested by Buja et al. (1989). Backfitting is essentially a block-Gauss-Seidel method presented in Alg. 12.

Buja et al. published their paper "Linear Smoothers and Additive Models" in the Annals of Statistics. They showed the convergence of Alg. 12 for (5.5) with $\mathbf{S}_j$ defined as in (5.3). However, the definition of smoothers they considered in their work was much broader and, besides linear smoothers, included other one-dimensional scatterplot and kernel smoothers. A linear smoother is symmetric positive definite and, as Proposition 2.6 shows, has eigenvalues between 0 and 1. The other smoothers may be unsymmetric and with unrestricted spectrum, which inhibits the convergence of the backfitting algorithm. Moreover, the backfitting algorithm fails to eliminate the errors that correspond to constant, linear, and low-frequency terms, which are associated with eigenvalues close to 1. Buja et al. suggested to remedy the situation by deflating those eigenvalues explicitly, which may be cumbersome and require a deeper analysis of the smoothers matrices.

The parallelisation of the backfitting algorithm is also not simple. Hegland, McIntosh, and Turlach (1999) developed an algorithm for parallel fitting of (generalised) additive models based on the backfitting procedure where they split the existing data and merge the results. This parallelisation procedure may work well if the amount of data is large but the smoothing models are inexpensive to compute. However, it offers little remedy if we want to distribute the computation of individual smoothers. The parallel execution of the blocked Gauss-Seidel Procedure of Alg. 12 is more difficult. In fact, we could not find any successful mentions in the literature and our own attempts to parallelise it failed miserably. The parallel backfitting would simply not converge.

This motivated my work with Markus Hegland on an alternative fitting procedure described in the following section. And since the interest in smoothing

Algorithm 12: Backfitting Algorithm

Input: $\mathbf{S}, \mathbf{S}_1, \ldots, \mathbf{S}_m$ smoothing matrices, $\mathbf{y}$ target vector
Output: $\mathbf{f}^T := (\mathbf{f}_1^T, \ldots, \mathbf{f}_m^T)$ predictions
while not converged **do**
  for $j = 1$ to m **do**
    $\mathbf{f}_j = \mathbf{S}_j \left(\mathbf{y} - \sum_{k \neq j} \mathbf{f}_k \right)$
  end for
end while

methods is not restricted to the sparse grid community, we studied a wide range of different models and not only sparse grids.

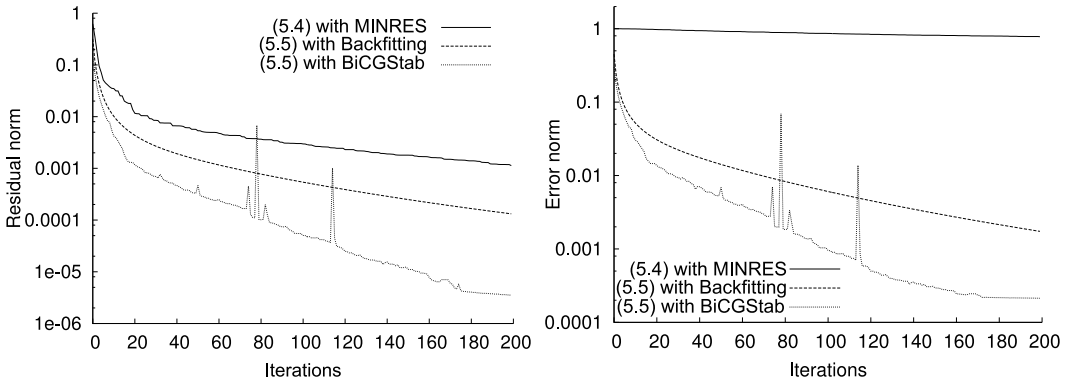
5.1.2. BiCGStab Procedure

In (Khakhutskyy & Hegland, 2014a) we suggested to solve (5.5) using a Krylov method. Experience has shown that Krylov methods can handle system matrices with clustered eigenvalues, characteristic for smoothers, better than the Gauss-Seidel procedure. Since the system matrix in (5.5) is not symmetric, we use the stabilised version of the BiCG algorithm, the Biconjugate Gradient Stabilised (BiCGStab) method, suggested by van der Vorst (1992) and shown in Alg. 13. We also adopt his notation throughout this section.

Algorithm 13: BiCGStab Algorithm without preconditioning for solving (5.5) adopted from (van der Vorst, 1992)

Input: $\tilde{\mathbf{S}}, \mathbf{S}_1, \ldots, \mathbf{S}_m$ smoothing matrices, $\mathbf{y}$ target vector
Output: $\mathbf{f}$ predictions of the additive models
$\mathbf{r}^T = (\mathbf{r}^0)^T = (\mathbf{S}_1\mathbf{y}, \mathbf{S}_2\mathbf{y}, \ldots, \mathbf{S}_m\mathbf{y})$
$\tilde{\mathbf{f}} = \mathbf{t} = \mathbf{v} = \mathbf{s} = \mathbf{0}$
$\alpha = \omega = 1$
$\rho^{\text{old}} = \rho^{\text{new}} = \beta = \mathbf{r}^T\mathbf{r}$
while not converged **do**
    $\beta = \rho^{\text{new}}/\rho^{\text{old}} \cdot \alpha/\omega$
    $\rho^{\text{old}} = \rho^{\text{new}}$
    $\mathbf{p} = \beta(\mathbf{p} - \omega\mathbf{v}) + \mathbf{r}$
    $\mathbf{v} = \tilde{\mathbf{S}}\mathbf{p}$ {synchronisation}
    $\alpha = \rho^{\text{old}}/\mathbf{v}^T\mathbf{r}^0$ {synchronisation}
    $\mathbf{s} = \mathbf{r} - \alpha\mathbf{v}$
    check convergence
    $\mathbf{t} = \tilde{\mathbf{S}}\mathbf{s}$ {synchronisation}
    $\omega = \frac{\mathbf{t}^T\mathbf{s}}{\mathbf{t}^T\mathbf{t}}$ {synchronisation}
    $\rho^{\text{new}} = -\omega\mathbf{t}^T\mathbf{r}^0$ {synchronisation}
    $\mathbf{r} = \mathbf{s} - \omega\mathbf{t}$
    $\tilde{\mathbf{f}} = \tilde{\mathbf{f}} + \omega\mathbf{s} + \alpha\mathbf{p}$
    check convergence
end while

In each iteration step the BiCGStab algorithm identifies two correction directions—$\mathbf{s}$ and $\mathbf{p}$—and step sizes—ω and α—that are used to modify the vector of unknowns $\tilde{\mathbf{f}}$. Ideally, we would like all correction directions to be orthogonal, similar to the conjugate gradients algorithm. Unfortunately, such orthogonalisation procedure is often infeasible if the system matrix is not symmetric. Instead, the pairs of the correction directions are *bi*orthogonal. The step sizes insure the



(a) Residual (b) Error

Figure 5.1.: Comparison of problem formulations (5.4) and (5.5) for minimisation of residual and error of a synthetic 100-dimensional dataset from model. Fitting of 1 000 data points was performed using regression cubic splines with $\lambda = 10^{-7}$ and dof=10. The problem formulation (5.5) is superior to (5.4). MINRES succeeds at minimising the residual this does not affect the prediction error. Source: (Khakhutskyy & Hegland, 2014a).

stable convergence of the algorithm, although they do not guarantee the minimisation of the residual norm in every step.

To illustrate the advantage of working with (5.5) instead of (5.4), we solve the same problem in both formulations and compare the convergence speed. Our benchmark is a synthetic dataset generated by a linear model with an additive noise term. The dataset has 100 dimensions, whereas only 10 of them are informative:

$$y = w_0 + w_1 x_1 + \ldots + w_{10} x_{10} + 0 x_{11} + \ldots + 0 x_{100} + \varepsilon$$

with uniformly distributed variables $x_1, \ldots, x_{100} \sim \mathcal{U}(0, 1)$ and a normally distributed noise $\varepsilon \sim \mathcal{N}(0, 0.01)$. We denote the parameter vector by $\mathbf{w}$.

We generated 1 000 samples and fitted a regression cubic spline to the data (Khakhutskyy & Hegland, 2014a). Figure 5.1 illustrates the decline relative residual norm $\|(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{D})\mathbf{w}^k - \mathbf{\Phi}^T\mathbf{y}\|/\|(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{D})\mathbf{w}^0 - \mathbf{\Phi}^T\mathbf{y}\|$ for MINRES and $\|\tilde{\mathbf{S}}\tilde{\mathbf{f}}^k - \mathbf{c}\|/\|\tilde{\mathbf{S}}\tilde{\mathbf{f}}^0 - \mathbf{c}\|$ for the backfitting and the BiCGStab; and the relative prediction error norm $\|\mathbf{f}^k - \mathbf{y}\|/\|\mathbf{f}^0 - \mathbf{y}\|$ for iteration k.

One can see the superiority of the problem formulation (5.5) both by using the classical backfitting algorithm and the BiCGStab method. While the residual in Formulation (5.4) decreases, this does not considerably affect the error of the resulting additive model. This is important since our main goal is to improve the prediction error.

If we compare between the backfitting and the BiCGStab methods, we observe that the error and residual norms of the backfitting algorithm decline more steadily but also much slower. Since BiCGStab does not directly minimise the error norm, some irregularities (spikes as seen in Fig. 5.1) are not uncommon, although the algorithm would usually continue to converge afterwards.

5.1.3. Updating Internal Parameters

Algorithm 13 describes the fitting procedure by adjusting the components $\mathbf{f}_j$ of $\tilde{\mathbf{f}}$. However, our primary goal is to learn the parameters $\mathbf{w}_j$. Fortunately, BiCGStab can be modified to update internal parameters of the model without additional costs.

It follows from (5.5) that, once the algorithm converged, we have

$$\mathbf{\Phi}_j \mathbf{w}_j = \mathbf{f}_j = \mathbf{S}_j \left(\mathbf{y} - \sum_{i \neq j} \mathbf{f}_i \right) \text{ for } j = 1, \ldots, m. \tag{5.6}$$

Since $\mathbf{S}_j = \mathbf{\Phi}_j (\mathbf{\Phi}_j^T \mathbf{\Phi}_j + \lambda \mathbf{D})^{-1} \mathbf{\Phi}_j^T$, (5.6) implies that

$$\mathbf{w}_j = \mathbf{\Phi}_j^\dagger \left(\mathbf{y} - \sum_{i \neq j} \mathbf{f}_i \right), \text{ with } \mathbf{\Phi}_j^\dagger := (\mathbf{\Phi}_j^T \mathbf{\Phi}_j + \lambda D)^{-1} \mathbf{\Phi}_j^T. \tag{5.7}$$

A naive approach would be to estimate $\mathbf{f}_j$ first and then recalculate $\mathbf{w}_j$ from (5.7). But nobody says "a naive approach" without having a better one up their sleeves. For the Master's thesis of Uphoff (2015) we derived a procedure that avoids the solution of (5.7) and instead updates $\mathbf{w}_j$ using already calculated results as the algorithm proceeds. This idea is based on two observations.

First, the multiplication of a matrix $\mathbf{S}_j$ with a vector can be seen as a two-step operation: In the first step, a vector of size N, number of data points, is multiplied by $\mathbf{\Phi}_j^\dagger$, which is a more expensive calculation. In the second step, the intermediate result is multiplied by $\mathbf{\Phi}_j$, which is cheaper.

Second, the prediction $\mathbf{f}_j$ is updated in Line 19 by adding the values $\omega \mathbf{s}$ and $\alpha \mathbf{p}$. If we denote the values $\omega, \mathbf{s}, \alpha$, and $\mathbf{p}$ at the iteration l by $\omega^l, \mathbf{s}^l, \alpha^l$, and $\mathbf{p}^l$ then after k iterations we have

$$\mathbf{f}_j^k = \sum_{l=1}^{k} (\omega^l \mathbf{s}^l + \alpha^l \mathbf{p}^l). \tag{5.8}$$

It then follows from (5.7) that

$$\mathbf{w}_j = \mathbf{\Phi}_j^\dagger \left(\mathbf{y} - \sum_{i \neq j} \sum_{l=1}^{k} \left(\omega^l \mathbf{s}^l + \alpha^l \mathbf{p}^l \right) \right) = \mathbf{\Phi}_j^\dagger \mathbf{y} - \sum_{l=1}^{k} \left(\omega^l \mathbf{\Phi}_j^\dagger \sum_{i \neq j} \mathbf{s}_i^l + \alpha^l \mathbf{\Phi}_j^\dagger \sum_{i \neq j} \mathbf{p}_j^l \right). \tag{5.9}$$

If we look at the smoothing procedure this way, we notice that the intermediate results in Lines 11 and 15 already give us $\mathbf{\Phi}_j^\dagger \left(\sum_{i\neq j} \mathbf{p}_i\right)$ and $\mathbf{\Phi}_j^\dagger \left(\sum_{i\neq j} \mathbf{s}_i\right)$.

Putting these observations together, our improved approach initiates the parameter vector $\mathbf{w}_j$ with the value $\mathbf{\Phi}_j^\dagger \mathbf{y}$ and then updates the vector once as

$$\mathbf{w}_j = \mathbf{w}_j - \alpha \mathbf{\Phi}_j^\dagger \mathbf{p}_j \tag{5.10}$$

after new α is computed in Line 12 and once as

$$\mathbf{w}_j = \mathbf{w}_j - \omega \mathbf{\Phi}_j^\dagger \mathbf{s}_j \tag{5.11}$$

after new ω is computed in Line 16.

5.1.4. Preconditioned BiCGStab

Preconditioners can often improve the convergence of Krylov solvers. Instead of solving a system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{b}$, we consider a preconditioned system

$$\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}} \tag{5.12}$$

with

$$\tilde{\mathbf{A}} = \mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-2} \text{ and } \tilde{\mathbf{b}} = \mathbf{M}_1^{-1}\mathbf{b}$$

for some nonsingular matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ such that $\mathbf{A} \approx \mathbf{M}_1\mathbf{M}_2$. Once (5.12) is solved, we retrieve the original solution as $\mathbf{x} = \mathbf{M}_2^{-1}\tilde{\mathbf{x}}$.

Together with Carsten Uphoff (2015) we studied the applicability and impact of several preconditioners for the solution of (5.5) with the BiCGStab method. The study showed that the benefit of most common preconditioners, i.e. SSOR and sparse approximations (ILU, SPAI, AINV), is very limited. The most promising results were achieved with a preconditioner Carsten called *LRW*: a **L**ow-**R**ank approximation of the matrices $\mathbf{S}_j$ and application of the **W**oodbury matrix identity.

The idea of this preconditioner is very elegant. It considers the matrix $\tilde{\mathbf{S}}$ in the form

$$\tilde{\mathbf{S}} = \begin{bmatrix} \mathbf{I}-\mathbf{S}_1 & & & \\ & \mathbf{I}-\mathbf{S}_2 & & \\ & & \ddots & \\ & & & \mathbf{I}-\mathbf{S}_m \end{bmatrix} + \begin{bmatrix} \mathbf{S}_1 \\ \mathbf{S}_2 \\ \vdots \\ \mathbf{S}_m \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} \end{bmatrix}, \tag{5.13}$$

computes the rank-k approximate SVD of the matrices $\mathbf{S}_j$ using the fast algorithm by Halko et al. (2011), and then applies the Sherman-Morrison-Woodbury formula (A.3) to obtain the approximate inverse of $\mathbf{S}$.

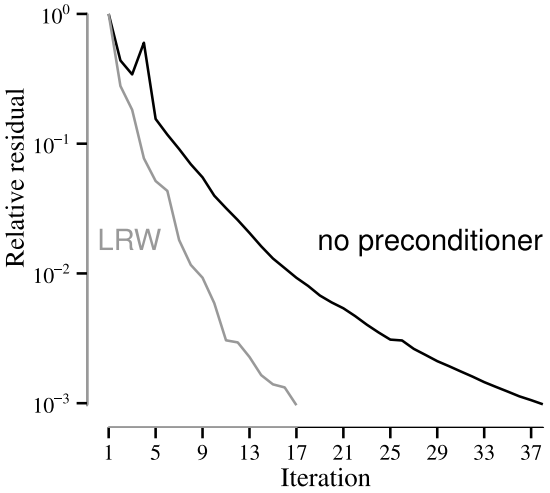


Figure 5.2.: Comparison of the relative residual for BiCGStab without preconditioner and with the LRW preconditioner. LRW facilitates the convergence significantly. Source: (Uphoff, 2015, Fig. 8.1.).

The LRW preconditioner allows to significantly accelerate the convergence of our BiCGStab method as shown in Fig. 5.2. However, the use of the the preconditioner requires an additional synchronisation step of the distributed algorithm (see the next section) and computation of the approximate SVD requires an additional setup time. Hence, the application of LRW makes sense only when every smoothing step is costly (e.g. for Nadaraya-Watson kernel smoothers described in the Master's thesis).

5.1.5. Parallel BiCGStab

The original BiCGStab method, as shown in Alg. 13, is not well suited for parallelisation since the operations in Lines 11, 12, 15, 16, and 17 require synchronisation. Therefore, we perform a series of modifications to the original algorithm exploiting the special structure of the matrix $\tilde{\mathbf{S}}$.

Let us now consider a parallel BiCGStab method shown in Alg. 14. The general idea behind the parallelisation is following: The large vectors $\mathbf{r}, \mathbf{r}^0, \mathbf{v}, \mathbf{s}, \mathbf{t}$, and $\mathbf{f}$ have mN entries for m smoothers and N data points. To parallelise the algorithm, we place different smoothers on different processing units. Therefore, we consider the subvectors of N elements $\mathbf{r}_j^0$, $\mathbf{v}_j$, etc. computed and stored on the same processing unit as $\mathbf{S}_j$. Now, to compute the block j of the vector $\mathbf{v}$ in Line 11, we can use the structure of the j-th block of the matrix $\tilde{\mathbf{S}}$ to obtain

$$\mathbf{v}_j = \begin{bmatrix}\mathbf{S}_j & \cdots & \mathbf{I} & \cdots \mathbf{S}_j\end{bmatrix} \mathbf{p} = \sum_{k \neq j} \mathbf{S}_j \mathbf{p}_k + \mathbf{p}_j = \mathbf{S}_j \left(\sum_{k=1}^{m} \mathbf{p}_k - \mathbf{p}_j \right) + \mathbf{p}_j. \quad (5.14)$$

We modify the algorithm to avoid the exchange of all mN elements of these vectors. Instead, we aggregate sums of subvectors or scalar products. For example, in the equation above we would aggregate the sum $\mathbf{p}_\Sigma = \sum_{k=1}^{m} \mathbf{p}_k$. Summation of vectors from different processing units with subsequent redistribution of the result is a common operation in parallel computing. There are efficient implementations of this operation in standard communication libraries. For instance, MPI libraries implement it as the function `Allreduce` with the SUM operand. We adopt this notation in Lines 4, 10, and 20 of Alg. 14 where we sum up the variables from the first argument across all processes and store the results in the variables of the second argument.

Now let us eliminate the synchronisation step in Line 12 of Alg. 13. Since all smoothing matrices $\mathbf{S}_j$ are symmetric, we can rewrite the dot product $\mathbf{v}^T\mathbf{r}^0$ as follows:

$$\mathbf{v}^T\mathbf{r}^0 = \sum_{j=1}^{m} \mathbf{v}_j^T\mathbf{r}_j^0 = \sum_{j=1}^{m} \left(\mathbf{p}_j + \mathbf{S}_j(\mathbf{p}_\Sigma - \mathbf{p}_j)\right)^T \mathbf{r}_j^0 \tag{5.15}$$

$$= \sum_{j}^{m} \mathbf{p}_j^T\mathbf{r}_j^0 + \mathbf{p}_\Sigma^T \sum_{j}^{m} \mathbf{S}_j\mathbf{r}_j^0 - \sum_{j}^{m} \mathbf{p}_j^T\mathbf{S}_j r_j^0. \tag{5.16}$$

Here the first and the last summands can be accumulated using `Allreduce`, while the middle summand can be precomputed at the beginning.

The computation of the scalar products in Lines 16 and 17 of Alg. 13 requires an exchange of scalar values between processing units. To avoid this additional communication, we reorder the algorithmic steps in a way that allows us to compute the dot products and matrix-vector products simultaneously by storing the variables in one memory segment and then performing an `Allreduce` summation on the complete segment.[1] The scalar products in the second argument of the `Allreduce` functions in Lines 10 and 20 of Alg. 14 stand for variables containing the corresponding scalar products. Note that due to the reordering, it is more convenient to aggregate the vectors $\mathbf{t}$ and $\mathbf{v}$ instead of $\mathbf{p}$ and $\mathbf{s}$.

Altogether, we reduced five synchronisation steps per iteration in Alg. 13 to only two in Alg. 14.

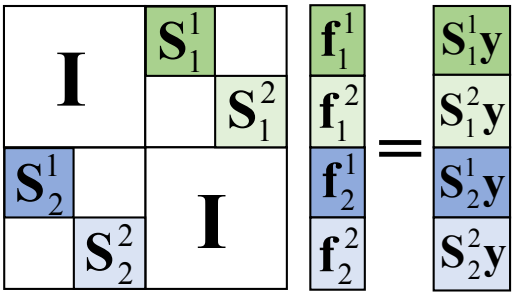
Besides this model-based parallelism, we can add a layer of data-parallelism (Khakhutskyy & Hegland, 2014b). The data is partitioned into subdomains using kd-trees. This allows us to decouple the individual problems as illustrated in Fig. 5.3. The smoothness of the solution across the partition borders may get lost. However, it is often not required in practice. Additionally, the decomposition complemented with orthogonalisation is known to adapt to the internal data manifolds (Guangliang & Maggioni, 2010).

If domain decomposition is used, the communication in Lines 10 and 20 of Alg. 14 can be performed in three steps to minimise the amount of sent and

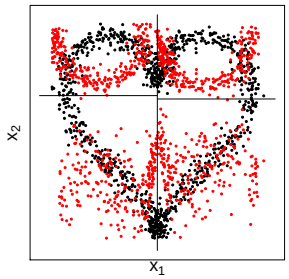
[1] A similar idea was proposed by L. Yang and Brent (2002).

Algorithm 14: Parallel BiCGStab Algorithm: code for processor $j, 1 \leq j \leq m$ (Khakhutskyy & Hegland, 2014a)

Input: $\mathbf{S}_j$ smoothing matrix, $\mathbf{y}$ target vector
Output: $\mathbf{f}_j$ predictions of the function $f_j(\mathbf{x})$ at the data points
$\mathbf{r}_j^0 = \mathbf{S}_j\mathbf{y}; \mathbf{r}_j = \mathbf{r}_j^0$
$\texttt{Allreduce}([\mathbf{r}_j, \mathbf{r}_j^T\mathbf{r}_j], [\mathbf{r}_\Sigma, \rho^{\text{new}}], \text{SUM})$
$\alpha = \omega = 1$; $\rho^{\text{old}} = \beta = \rho^{\text{new}}$
$\mathbf{f}_j = \mathbf{t}_j = \mathbf{v}_j = \mathbf{v}_\Sigma = \mathbf{p}_j = \mathbf{p}_\Sigma = \mathbf{s}_j = \mathbf{s}_\Sigma = \mathbf{0}$
while not converged **do**
  if iteration > 0 **then**
    $\rho^{\text{old}} = \rho^{\text{new}}$
    $\texttt{Allreduce}([\mathbf{s}_j^T\mathbf{r}_j^0, \mathbf{t}_j^T\mathbf{s}_j, \mathbf{t}_j^T\mathbf{t}_j, \mathbf{t}_j^T\mathbf{r}_j^0, \mathbf{t}_j], [\mathbf{s}^T\mathbf{r}^0, \mathbf{t}^T\mathbf{s}, \mathbf{t}^T\mathbf{t}, \mathbf{t}^T\mathbf{r}^0, \mathbf{t}_\Sigma], \text{SUM})$
    $\omega = \frac{\mathbf{t}^T\mathbf{s}}{\mathbf{t}^T\mathbf{t}}; \rho^{\text{new}} = \mathbf{s}^T\mathbf{r}^0 - \omega\mathbf{t}^T\mathbf{r}^0; \beta = \frac{\rho^{\text{new}}}{\rho^{\text{old}}} \cdot \frac{\alpha}{\omega}$
    $\rho^{\text{old}} = \rho^{\text{new}}$
    $\mathbf{r}_j = \mathbf{s}_j - \omega\mathbf{t}_j; \mathbf{r}_\Sigma = \mathbf{s}_\Sigma - \omega\mathbf{t}_\Sigma$
    $\mathbf{f}_j = \mathbf{f}_j + \omega\mathbf{s}_j$
    check convergence on $\mathbf{r}$
  end if
  $\mathbf{p}_j = \beta(\mathbf{p}_j - \omega\mathbf{v}_j) + \mathbf{r}_j$
  $\mathbf{p}_\Sigma = \beta(\mathbf{p}_\Sigma - \omega\mathbf{v}_\Sigma) + \mathbf{r}_\Sigma$
  $\mathbf{v}_j = \mathbf{p}_j + \mathbf{S}_j(\mathbf{p}_\Sigma - \mathbf{p}_j)$
  $\texttt{Allreduce}\big([\mathbf{v}_j^T\mathbf{r}_j^0, \mathbf{v}_j], [\mathbf{v}^T\mathbf{r}^0, \mathbf{v}_\Sigma], \text{SUM}\big)$
  $\alpha = \rho^{\text{old}}/\mathbf{v}^T\mathbf{r}^0$
  $\mathbf{s}_j = \mathbf{r}_j - \alpha\mathbf{v}_j; \mathbf{s}_\Sigma = \mathbf{r}_\Sigma - \alpha\mathbf{v}_\Sigma$
  $\mathbf{f}_j = \mathbf{f}_j + \alpha\mathbf{p}_j$
  check convergence on $\mathbf{s}$
  $\mathbf{t}_j = \mathbf{s}_j + \mathbf{S}_j(\mathbf{s}_\Sigma - \mathbf{s}_j)$
end while



(a) Task and data decomposition of Equation (5.5)



(b) Data partitioning and transformation with kd-trees

Figure 5.3.: Parallelisation of the fitting algorithm using task and data decomposition with kd-trees. On the left: Transformation of Equation (5.5). The coloured blocks are computed simultaneously. On the right: Kd-tree based partitioning (solid lines) of data with a low-dimensional manifold (black dots) and its orthogonalisation in individual partitions (red dots). Source: (Khakhutskyy & Hegland, 2014b).

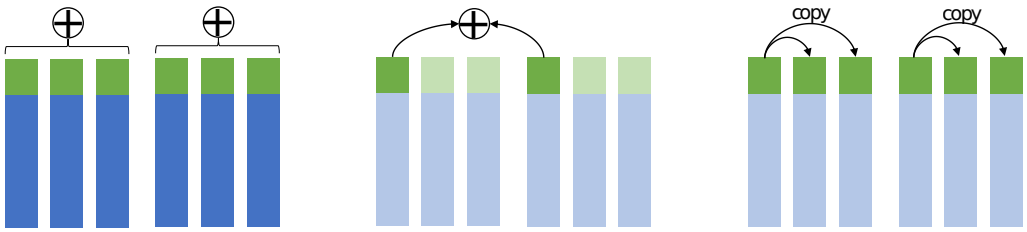
received data (see Fig. 5.4). In the first step, we calculate the sum of large vectors $[\mathbf{s}_j^T\mathbf{r}_j^0, \mathbf{t}_j^T\mathbf{s}_j, \mathbf{t}_j^T\mathbf{t}_j, \mathbf{t}_j^T\mathbf{r}_j^0, \mathbf{t}_j]$ and $[\mathbf{v}_j^T\mathbf{r}_j^0, \mathbf{v}_j]$ among the processors that share the same data partition (Fig. 5.4a). At this stage the size of the communicated vectors is basically the number of points in the partition. In the second step, only the scalar product results $[\mathbf{s}_j^T\mathbf{r}_j^0, \mathbf{t}_j^T\mathbf{s}_j, \mathbf{t}_j^T\mathbf{t}_j, \mathbf{t}_j^T\mathbf{r}_j^0]$ and $[\mathbf{v}_j^T\mathbf{r}_j^0]$ are aggregated between the root processes of individual partitions (Fig. 5.4b). While the communication between partitions is more expensive in distributed clusters, we need to communicate only a small constant number of values, which is efficiently implemented in MPI libraries. In the last step, the updated values are broadcasted from the roots to all processes in the partitions (Fig. 5.4c).

We illustrate the performance of our parallelised BiCGStab algorithm using the MSD benchmark introduced in Sec. 3.5.3. To remind you, the dataset consists of 463 715 training examples with 88 numerical attributes describing the song's acoustic properties.[2] We use linear models as smoothing operators $\mathbf{S}_j$, although an extension to other smoothers is straightforwards.

As mentioned above, data partitioning and orthogonalisation can improve the performance and accelerate the convergence. Figure 5.5 compares the relative residual norm and the mean prediction error for the same model with and without orthogonalisation. The orthogonalised version converges after the first step!

Figure 5.6a illustrates the strong scaling of our algorithm. To make the plot, we conducted 45 iterations using different number of processors: between 2 and

[2] Actually, there are 90 features but 88 is easier to split evenly among different number of processors.



(a) Step 1 (b) Step 2 (c) Step 3

Figure 5.4.: Three-steps communication model for Lines 10 and 20 of Alg. 14 with data partitioning. In the first step, we calculate the sum of large vectors among the processors that share the same data partition (a). In the second step, only the scalar product results are aggregated between the root processes of individual partitions (b). In the last step, the updated values are broadcasted from the roots to all processes in the partitions (c). Source: (Khakhutskyy & Hegland, 2014b).

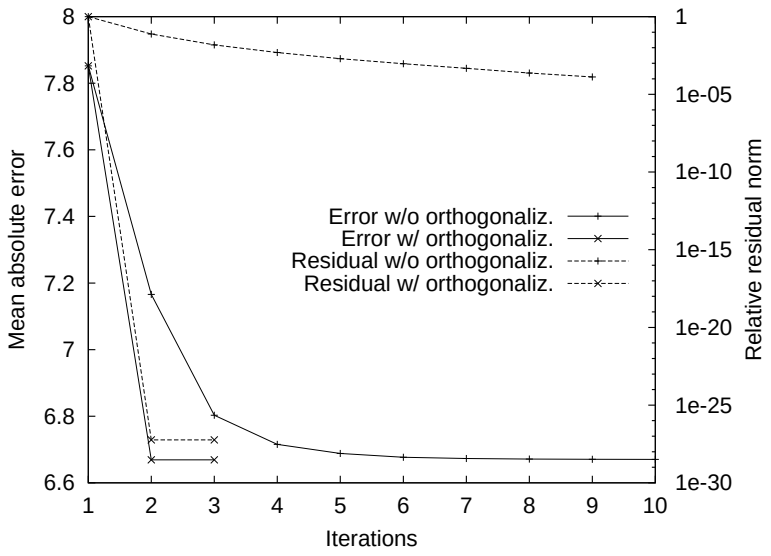
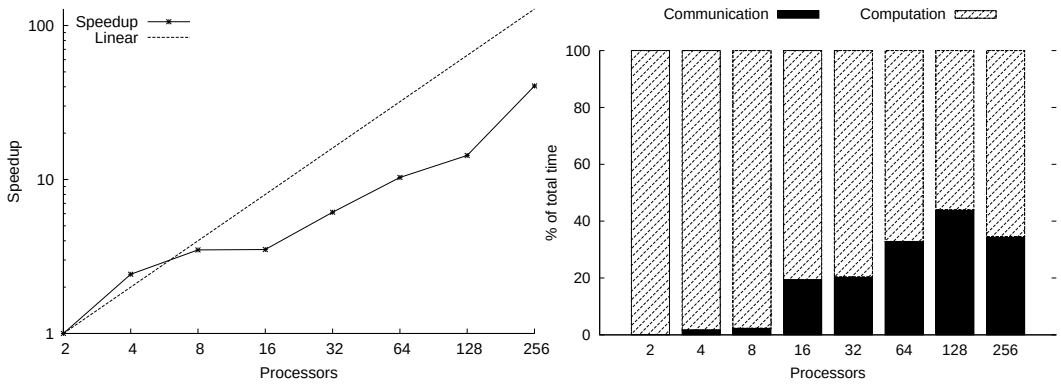


Figure 5.5.: Comparison of the relative residual norm and the mean prediction error for the same model with and without orthogonalisation on the MSD. Source: (Khakhutskyy & Hegland, 2014b).



(a) Strong scaling (b) Computation/communication

Figure 5.6.: Scaling properties of Alg. 14. On the left: strong scaling results for the Million Songs dataset with 88 features and 463 715 examples. On the right: the computation-to-communication relationship for different number of processors. Due to the limited size of the dataset we show the strong scaling results of up to 256 processors. Source: (Khakhutskyy & Hegland, 2014b).

256. There is a trade-off between the reduction of computation time and the increase of communication overhead (Fig. 5.6b). For up to 8 processors, every process receives all examples. Starting with 16 processors, we use data partitioning. This is responsible for a jump in communication overhead. An application of more computationally intensive basis functions or smoothing kernels would improve the computation/communication ratio.

Due to the limited size of the dataset we show the strong scaling results for up to 256 processors. To use more processors we would need to split the dataset further, which does not make sense in this particular case. However, these diagrams are characteristic and similar behaviour is expected when the total number of processors grows proportionally to the dataset size.

While originally motivated in statistics, fitting a system of smoothers can find a much broader applicability as a framework for handling problems with additive structure. We derived a new Krylov fitting method that exhibits a faster convergence than the state of the art per se. Additionally, if necessary, this method can be parallelised or further accelerated using a preconditioner. As always, the usefulness of such measures is governed by the trade-off between the gain and the additional overhead. As we demonstrated, this method can be applied to a broad range of smoothers including sparse grid models.

The application of the method discussed in this section is limited to problems with ℓ_2^2 regularisation. This is a broadly used regularisation penalty, but, as we showed earlier, not the only one of interest. In the next section we consider an alternative problem formulation in combination with an optimisation algorithm that can be used with a wide range of regularisation functions.

5.2. Alternating Direction Method of Multipliers

The Alternating Directions Method of Multipliers (ADMM) is a further development of the dual ascent algorithm by Arrow, Hurwicz, and Uzawa (1958). The method gained recent popularity largely due to the seminal review by Boyd et al. (2010) whose exposition we follow in the introduction of the method in Sec. 5.2.1. Afterwards, we show in Sec. 5.2.2 the connection between ADMM and backfitting, uncovering the relationships unknown before. Sections 5.2.3 and 5.2.4 discuss the distributed version of the ADMM algorithm for sparse grid models with synchronous and asynchronous communication patterns.

5.2.1. Introduction to ADMM

ADMM solves constrained optimisation problems of the form

$$\begin{aligned} \min_{\mathbf{x},\mathbf{z}} \quad & h(\mathbf{x}) + g(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c} \end{aligned} \tag{5.17}$$

with variables $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{z} \in \mathbb{R}^m$, linear constraints given by a vector $\mathbf{c} \in \mathbb{R}^p$ and matrices $\mathbf{A} \in \mathbb{R}^{p\times n}$, $\mathbf{B} \in \mathbb{R}^{p\times m}$, and convex cost functions $h : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^m \to \mathbb{R}$. Note that for now this notation is general and has nothing to do with the notation, for the input variable $\mathbf{x}$, used in the previous chapters. Later on we adopt the specific notation used elsewhere in the thesis.

The problem (5.17) is solved iteratively using the augmented Lagrangian function

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = h(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2, \tag{5.18}$$

with the Lagrangian multiplier $\mathbf{y} \in \mathbb{R}^p$ and a positive penalisation coefficient $\rho \in \mathbb{R}^+$.

In each iteration of the ADMM algorithm, the $\mathbf{x}$ and $\mathbf{z}$ variables are calculated as minimisers of the augmented Lagrangian

$$\mathbf{x}^{k+1} := \arg\min_{\mathbf{x}} L(\mathbf{x}, \mathbf{z}^k, \mathbf{y}^k), \tag{5.19}$$

$$\mathbf{z}^{k+1} := \arg\min_{\mathbf{z}} L(\mathbf{x}^{k+1}, \mathbf{z}, \mathbf{y}^k) \tag{5.20}$$

followed by the update of the Lagrangian multiplier variable

$$\mathbf{y}^{k+1} := \mathbf{y}^k + \rho(\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}). \tag{5.21}$$

Often, it is more convenient to use a so-called *scaled form* of the equations (5.19)–(5.21). Let $\mathbf{r} := \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}$ denote the residual and let $\mathbf{u} := (1/\rho)\mathbf{y}$

denote the scaled dual variable. Then

$$\begin{aligned}\mathbf{y}^T\mathbf{r} + (\rho/2)\|\mathbf{r}\|_2^2 &= (\rho/2)\|\mathbf{r} + (1/\rho)\mathbf{y}\|_2^2 - (1/2\rho)\|\mathbf{y}\|_2^2 \\ &= (\rho/2)\|\mathbf{r} + \mathbf{u}\|_2^2 - (\rho/2)\|\mathbf{u}\|_2^2.\end{aligned}$$

Thus, the ADMM updates (5.19)–(5.21) can be expressed as

$$\mathbf{x}^{k+1} := \arg\min_{\mathbf{x}} \left(h(\mathbf{x}) + (\rho/2)\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}^k - \mathbf{c} + \mathbf{u}^k\|_2^2\right), \tag{5.22}$$

$$\mathbf{z}^{k+1} := \arg\min_{\mathbf{z}} \left(g(\mathbf{z}) + (\rho/2)\|\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}^k\|_2^2\right), \tag{5.23}$$

$$\mathbf{u}^{k+1} := \mathbf{u}^k + \mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}. \tag{5.24}$$

Note that, since $\mathbf{A}\mathbf{x}^t + \mathbf{B}\mathbf{z}^t - \mathbf{c}$ can be interpreted as the residual at iteration t, the scaled dual variable $\mathbf{u}^k = \mathbf{u}^0 + \sum_{t=1}^k \mathbf{A}\mathbf{x}^t + \mathbf{B}\mathbf{z}^t - \mathbf{c}$ is the running sum of residuals.

The usage of the penalisation term $(\rho/2)\|\cdot\|_2^2$ in the augmented Lagrangian function improves the convergence of the dual ascent algorithm (Fortin & Glowinski, 1983). It can be easily seen that for feasible solutions $(\mathbf{x}^*, \mathbf{z}^*)$ the penalty becomes 0. If the saddle point of the Lagrangian function exists, Gabay and Mercier (1976) have shown that the saddle point of the Lagrangian function without the penalisation term is a saddle point of L_ρ and vice versa.

The convergence of the ADMM for closed, proper convex functions $h : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ and $\rho \geq 0$ was shown by Gabay and Mercier in 1976. More recently, He, Yang, and Wang (2000) were able to give the convergence rate using a variational inequality formulation of Problem (5.17). The authors showed that after T iterations of the ADMM algorithm the averaged aggregated solution vector $\overline{(\mathbf{x}^k, \mathbf{z}^k, \mathbf{y}^k)^T} = 1/T \sum_{k=1}^T (\mathbf{x}^k, \mathbf{z}^k, \mathbf{y}^k)^T$ would approximate the solution of the variational inequality with accuracy $\mathcal{O}(1/T)$.

To show the relationship between ADMM and proximal operators, let $\mathbf{v}^k := -\mathbf{B}\mathbf{z}^k + \mathbf{c} - \mathbf{u}^k$ and assume $\mathbf{A} = \mathbf{I}$. Then we can express the $\mathbf{x}$-update step (5.22) as

$$\mathbf{x}^{k+1} := \arg\min_{\mathbf{x}} \left(h(\mathbf{x}) + (\rho/2)\|\mathbf{x} - \mathbf{v}\|_2^2\right). \tag{5.25}$$

In this update step we can recognise the proximal operator (3.27):

$$\mathbf{x}^{k+1} := \text{prox}_{(1/\rho)h}(-\mathbf{B}\mathbf{z}^k + \mathbf{c} - \mathbf{u}^k). \tag{5.26}$$

Analogously, we can define the update of $\mathbf{z}$ using the proximal operator as

$$\mathbf{z}^{k+1} := \text{prox}_{(1/\rho)g}(-\mathbf{A}\mathbf{x}^{k+1} + \mathbf{c} - \mathbf{u}^k). \tag{5.27}$$

If $\mathbf{A} \neq \mathbf{I}$, then the updates (5.22) and (5.23) are, strictly speaking, not proximal operators in itself. However, they are strongly related to these operators as we will see later.

ADMM is often used when the properties of the functions h and g—i.e. convexity, differentiability, or separability—are different. Then the particular proximal operators (5.26) and (5.27) can benefit from exploiting these properties.

To derive a concrete ADMM algorithm for empirical risk minimisation with sparse grid models, we observe that for the models with additive structure the regularisation term described in Sec. 4.3.2 can be divided in a sum of penalty terms on subvectors of parameters, like in (4.30). At the same time, the residual sum of squares $\sum_{i=1}^{N}(\sum_{j=1}^{m} f_j(\mathbf{x}_i) - y_i)^2$ cannot be divided. This yields the empirical risk minimisation problem of the form

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_m} \frac{1}{2}\sum_{i=1}^{N}\left(\sum_{j=1}^{m}\mathbf{w}_j^T\boldsymbol{\phi}_j(\mathbf{x}_i) - y_i\right)^2 + \sum_{j=1}^{m}\lambda\Omega_j(\mathbf{w}_j), \tag{5.28}$$

where $\boldsymbol{\phi}_j(\mathbf{x}_i)$ denotes a vector with the basis functions from the split j evaluated at the point $\mathbf{x}_i$.

To apply the ADMM, we bring this minimisation problem in form (5.17). In order to do this, we introduce an auxiliary variable $\mathbf{f}_j$ to signify the result of the matrix vector product $\boldsymbol{\Phi}_j\mathbf{w}_j$ and connect it to $\mathbf{w}_j$ using the constraint. We then obtain the minimisation problem

$$\begin{aligned}\min_{\substack{\mathbf{f}_1,\ldots,\mathbf{f}_m\\ \mathbf{w}_1,\ldots,\mathbf{w}_m}} \frac{1}{2}\left\|\sum_{j=1}^{m}\mathbf{f}_j - \mathbf{y}\right\|_2^2 + \sum_{j=1}^{m}\lambda\Omega_j(\mathbf{w}_j)\\ \text{subject to } \boldsymbol{\Phi}_j\mathbf{w}_j = \mathbf{f}_j, \quad j = 1,\ldots,m.\end{aligned}$$

Let $g(\mathbf{f}) := 1/2\|\mathbf{f} - \mathbf{y}\|^2$ and $h_j(\mathbf{w}_j) := \lambda\Omega_j(\mathbf{w}_j)$. By splitting the problem into a sum of two functions, we obtain an optimisation problem in the form required for ADMM:

$$\begin{aligned}\min \quad & g\left(\sum_{j=1}^{m}\mathbf{f}_j\right) + \sum_{j=1}^{m}h_j(\mathbf{w}_j)\\ \text{subject to} \quad & \boldsymbol{\Phi}_j\mathbf{w}_j = \mathbf{f}_j \quad j = 1,\ldots,m.\end{aligned} \tag{5.29}$$

Application of the ADMM steps (5.22)– (5.24) leads to the updates

$$\mathbf{w}_j^{k+1} := \arg\min_{\mathbf{w}_j}\left\{\lambda\Omega_j(\mathbf{w}_j) + \frac{\rho}{2}\|\boldsymbol{\Phi}_j\mathbf{w}_j - \mathbf{f}_j^k + \mathbf{u}_j^k\|_2^2\right\}, \tag{5.30}$$

$$\tilde{\mathbf{f}}^{k+1} := \arg\min_{\tilde{\mathbf{f}}=(\mathbf{f}_1^T,\ldots,\mathbf{f}_m^T)^T} \frac{1}{2}\left\{\left\|\sum_{j=1}^{m}\mathbf{f}_j - \mathbf{y}\right\|_2^2 + \sum_{j=1}^{m}\frac{\rho}{2}\left\|\boldsymbol{\Phi}_j\mathbf{w}_j^{k+1} - \mathbf{f}_i + \mathbf{u}_j^k\right\|_2^2\right\}, \tag{5.31}$$

$$\mathbf{u}_j^{k+1} := \mathbf{u}_j^k + \boldsymbol{\Phi}_j\mathbf{w}_j^{k+1} - \mathbf{f}_j^{k+1}. \tag{5.32}$$

The update (5.31) involves optimisation over a mN-dimensional space, which would be infeasible for most applications. Fortunately, it is not required since it can be proven (see, e.g., (Boyd et al., 2010)) that solving the minimisation step (5.31) is equivalent to solving

$$\bar{\mathbf{f}}^{k+1} = \arg\min_{\bar{\mathbf{f}}} \tfrac{1}{2}\|m\bar{\mathbf{f}} - \mathbf{y}\|_2^2 + \frac{m\rho}{2}\|\bar{\mathbf{f}} - \frac{1}{m}\sum_{j=1}^{m}\boldsymbol{\Phi}_j\mathbf{w}_j^{k+1} - \tfrac{1}{m}\sum_{j=1}^{m}\mathbf{u}_j^{k+1}\|_2^2. \quad (5.33)$$

Individual variables $\mathbf{f}_j$ can then be recovered as

$$\mathbf{f}_j^{k+1} = \boldsymbol{\Phi}_j\mathbf{x}_j^{k+1} + \mathbf{u}_j^k + \bar{\mathbf{f}}^{k+1} - \frac{1}{m}\sum_{j=1}^{m}\boldsymbol{\Phi}_j\mathbf{w}_j^{k+1} - \frac{1}{m}\sum_{j=1}^{m}\mathbf{u}_j^k\ . \quad (5.34)$$

Finally, by substituting (5.33) and (5.34) into (5.30), (5.31), and (5.32), one can see that the different Laplacian multiplier vectors $\mathbf{u}_j$ become equal. Hence, we can use a single vector $\mathbf{u}$. We then obtain the equations

$$\mathbf{w}_j^{k+1} := \arg\min_{\mathbf{w}_j\in\mathbb{R}^{M_j}}\left\{\lambda\Omega_j(\mathbf{w}_j) + \frac{\rho}{2}\left\|\boldsymbol{\Phi}_j\mathbf{w}_j - \boldsymbol{\Phi}_j\mathbf{w}_j^k - \bar{\mathbf{f}}^k + \frac{1}{m}\sum_{j=1}^{m}\boldsymbol{\Phi}_j\mathbf{w}_j^k + \mathbf{u}^k\right\|_2^2\right\}, \quad (5.35)$$

$$\bar{\mathbf{f}}^{k+1} := \arg\min_{\mathbf{f}\in\mathbb{R}^N}\left\{\frac{1}{2}\|m\cdot\mathbf{f} - \mathbf{y}\|_2^2 + \frac{m\cdot\rho}{2}\left\|\mathbf{f} - \frac{1}{m}\sum_{j=1}^{m}\boldsymbol{\Phi}_j\mathbf{w}_j^{k+1} - \mathbf{u}^k\right\|_2^2\right\}, \quad (5.36)$$

$$\mathbf{u}^{k+1} := \mathbf{u}^k + \frac{1}{m}\sum_{i=1}^{m}\boldsymbol{\Phi}_j\mathbf{w}_j^{k+1} - \bar{\mathbf{f}}^{k+1}\ . \quad (5.37)$$

Let us comment on the technical details of the steps (5.35) and (5.36). The minimisation problem (5.36) has an analytical solution and its minimiser is inexpensive to compute:

$$\bar{\mathbf{f}}^{k+1} = \frac{1}{m+\rho}\left(\mathbf{y} + \rho\tfrac{1}{m}\sum_{j=1}^{m}\boldsymbol{\Phi}_j\mathbf{w}_j^{k+1} + \rho\mathbf{u}^k\right).$$

The update step (5.35) is, strictly speaking, not a proximal operator since it has the form $\arg\min_{\mathbf{w}_j}\{\lambda\Omega_j(\mathbf{w}_j) + (\rho/2)\|\boldsymbol{\Phi}_j\mathbf{w}_j - \mathbf{v}\|_2^2\}$, with appropriately defined $\mathbf{v}$, instead of $\arg\min_{\mathbf{w}_j}\{\lambda\Omega_j(\mathbf{w}_j) + (\rho/2)\|\mathbf{w}_j - \mathbf{v}\|_2^2\}$ (notice the missing $\boldsymbol{\Phi}_j$). Naturally, the solution of the modified operators can be computed similarly to the original proximal operators summarised in Table 3.2. The minimisation step for common forms of Ω_j were described by Boyd et al. (2010, Sec. 8.3). For

completeness, we summarise them here and give details on efficient implementation.

ℓ_2^2-norm If $\Omega_j(\mathbf{w}_j) = \|\mathbf{w}_j\|_2^2$, the step (5.35) becomes a ridge regression problem

$$\begin{aligned}\mathbf{w}_j^{k+1} &:= \arg\min_{\mathbf{w}_j} \tfrac{\rho}{2}\|\mathbf{\Phi}_j\mathbf{w}_j - \mathbf{v}\|_2^2 + \lambda\|\mathbf{w}_j\|_2^2 \\ &= (\mathbf{\Phi}_j^T\mathbf{\Phi}_j + \frac{2\lambda}{\rho}\mathbf{I})^{-1}\mathbf{\Phi}_j^T\mathbf{v}.\end{aligned}$$

Since this step requires a repeated solution of linear systems with the same system matrix and different right-hand-side vectors, precomputing the Cholesky factorisation of the matrix $(\mathbf{\Phi}_j^T\mathbf{\Phi}_j + (2\lambda/\rho)\mathbf{I})$ at the initialisation of the algorithm could improve the overall performance. The Cholesky factorisation often becomes feasible due to this splitting and parallelisation (see Sec. 5.2.3). While the total matrix does not fit in the main memory of a single node, the small matrices of the splits do.

ℓ_1-norm If $\Omega_j(\mathbf{w}_j) = \|\mathbf{w}_j\|_1$, the step (5.35) becomes a lasso problem

$$\mathbf{w}_j^{k+1} := \arg\min_{\mathbf{w}_j} \tfrac{\rho}{2}\|\mathbf{\Phi}_j\mathbf{w}_j - \mathbf{v}\|_2^2 + \lambda\|\mathbf{w}_j\|_1.$$

We discussed the methods for solution of lasso problems in Sec. 3.4.

To accelerate the computation, we observe that

$$\|\mathbf{\Phi}_j^T\mathbf{v}\|_2 \leq \tfrac{\lambda}{\rho} \Leftrightarrow \mathbf{w}_j = 0.$$

Hence, we need to solve the lasso problem only if $\|\mathbf{\Phi}_j^T\mathbf{v}\|_2 > \lambda/\rho$.

ℓ_1/ℓ_2-norm If $\Omega_j(\mathbf{w}_j) = \|\mathbf{w}_j\|_2$, the step (5.35) becomes

$$\mathbf{w}_j^{k+1} = \arg\min_{\mathbf{w}_j} \tfrac{\rho}{2}\|\mathbf{\Phi}_j\mathbf{w}_j - \mathbf{v}\|_2^2 + \lambda\|\mathbf{w}_j\|_2.$$

Similar to lasso, we have

$$\|\mathbf{\Phi}_j^T\mathbf{v}\|_2 \leq \tfrac{\lambda}{\rho} \Leftrightarrow \mathbf{w}_j = 0.$$

If $\|\mathbf{\Phi}_j^T\mathbf{v}\|_2 > \lambda/\rho$ then the solution has the form

$$\mathbf{w}_j = (\mathbf{\Phi}_j^T\mathbf{\Phi}_j + \nu\mathbf{I})^{-1}\mathbf{\Phi}_j^T\mathbf{v}$$

for some positive $\nu > 0$ such that $\nu\|\mathbf{w}_j\|_2 = \lambda/\rho$. Such ν can be found using a line-search method.

Again, $(\mathbf{\Phi}_j^T\mathbf{\Phi}_j + \nu\mathbf{I})$ does not change between iterations and can be decomposed using Cholesky factorisation. Alternatively, we can use the eigenvalue decomposition of $\mathbf{\Phi}_j^T\mathbf{\Phi}_j$ to accelerate the search for the optimal parameters λ and ν. If $\mathbf{\Phi}_j^T\mathbf{\Phi}_j = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$, then $\|\mathbf{w}_j\|_2 = \|(\mathbf{\Lambda} + \nu\mathbf{I})^{-1}\|_2 \cdot \|\mathbf{Q}^T\mathbf{\Phi}_j^T\mathbf{v}\|_2$. It only remains to compute $\mathbf{Q}^T\mathbf{\Phi}_j^T\mathbf{v}$, which is relatively cheap, and inversion of the diagonal matrix $\mathbf{\Lambda} + \nu\mathbf{I}$ is just linear in the size of $\mathbf{w}_j$.

5.2.2. On the Connection between ADMM and Backfitting

Historically, the backfitting algorithm for solving (5.5) and the ADMM were considered independently: one rises from statistics and linear smoothers while the other from partial differential equations and constrained optimisation with proximal operators. While the convergence analysis of ADMM splitting is still out of reach, we arrive at a formulation that gives us a good intuition about the convergence properties of the algorithm.

To summarise the results from the previous section, for $\Omega(\mathbf{w}) = \|\mathbf{w}\|_2^2$ ADMM iterates between the following sequence of updates:

$$(\rho\mathbf{\Phi}_j^T\mathbf{\Phi}_j + \lambda\mathbf{I})\ \mathbf{w}_j^{k+1} = \rho\mathbf{\Phi}_j^T(\mathbf{\Phi}_j\mathbf{w}_j^k + \bar{\mathbf{f}}^k - \frac{1}{m}\sum_{i=1}^m \mathbf{\Phi}_i\mathbf{w}_i^k - \mathbf{u}^k) \tag{5.38}$$

$$\bar{\mathbf{f}}^{k+1} = \frac{1}{m+\rho}\left(\mathbf{y} + \frac{\rho}{m}\sum_{i=1}^m \mathbf{\Phi}_i\mathbf{w}_i^{k+1} + \rho\mathbf{u}^k\right) \tag{5.39}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \frac{1}{m}\sum_{i=1}^m \mathbf{\Phi}_i\mathbf{w}_i^{k+1} - \bar{\mathbf{f}}^{k+1}. \tag{5.40}$$

To establish the relationship between ADMM and smoothing equations, we need to transform them into the same form. Therefore, we rewrite the ADMM updates as a stationary method for solving a system of linear equations. Then we deduce what this system looks like.

Proposition 5.1. *Let*

$$\mathbf{x} = \begin{pmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_m \\ \bar{\mathbf{f}} \end{pmatrix} \quad \textit{and} \quad \mathbf{b} = \frac{1}{\rho}\begin{pmatrix} \mathbf{\Phi}_1^T\mathbf{y} \\ \mathbf{\Phi}_2^T\mathbf{y} \\ \vdots \\ \mathbf{\Phi}_m^T\mathbf{y} \\ \mathbf{0} \end{pmatrix}.$$

The ADMM algorithm (5.38)– (5.40) *corresponds to a stationary method of the form*

$$\mathbf{M}\mathbf{x}^{k+1} = \mathbf{N}\mathbf{x}^k + \mathbf{b}$$

with

$$\mathbf{M} = \begin{bmatrix} (\frac{\lambda}{\rho}\mathbf{I} + \mathbf{\Phi}_1^T\mathbf{\Phi}_1) & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\frac{\lambda}{\rho}\mathbf{I} + \mathbf{\Phi}_2^T\mathbf{\Phi}_2) & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & (\frac{\lambda}{\rho}\mathbf{I} + \mathbf{\Phi}_m^T\mathbf{\Phi}_m) & \mathbf{0} \\ -\frac{\rho}{m}\mathbf{\Phi}_1 & -\frac{\rho}{m}\mathbf{\Phi}_2 & \dots & -\frac{\rho}{m}\mathbf{\Phi}_m & (m+\rho)\mathbf{I} \end{bmatrix} \tag{5.41}$$

$$\mathbf{N} = \begin{bmatrix} (1-\frac{1}{m})\mathbf{\Phi}_1^T\mathbf{\Phi}_1 & -\frac{1}{m}\mathbf{\Phi}_1^T\mathbf{\Phi}_2 & \dots & -\frac{1}{m}\mathbf{\Phi}_1^T\mathbf{\Phi}_m & (1-\frac{m}{\rho})\mathbf{\Phi}_1^T \\ -\frac{1}{m}\mathbf{\Phi}_2^T\mathbf{\Phi}_1 & (1-\frac{1}{m})\mathbf{\Phi}_2^T\mathbf{\Phi}_2 & \dots & -\frac{1}{m}\mathbf{\Phi}_2^T\mathbf{\Phi}_2 & (1-\frac{m}{\rho})\mathbf{\Phi}_2^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\frac{1}{m}\mathbf{\Phi}_m^T\mathbf{\Phi}_1 & -\frac{1}{m}\mathbf{\Phi}_m^T\mathbf{\Phi}_2 & \dots & (1-\frac{1}{m})\mathbf{\Phi}_m^T\mathbf{\Phi}_m & (1-\frac{m}{\rho})\mathbf{\Phi}_m^T \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & m\mathbf{I} \end{bmatrix}. \tag{5.42}$$

If $\mathbf{M}$ *is nonsingular and the spectral radius* $\rho(\mathbf{M}^{-1}\mathbf{N}) < 1$ *then this method converges to the solution of the system of linear equations*

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{5.43}$$

with

$$\begin{aligned} \mathbf{A} &= \mathbf{M} - \mathbf{N} \\ &= \begin{bmatrix} \left(\frac{\lambda}{\rho}\mathbf{I} + \frac{1}{m}\mathbf{\Phi}_1^T\mathbf{\Phi}_1\right) & \frac{1}{m}\mathbf{\Phi}_1^T\mathbf{\Phi}_2 & \dots & \frac{1}{m}\mathbf{\Phi}_1^T\mathbf{\Phi}_m & (\frac{m}{\rho}-1)\mathbf{\Phi}_1^T \\ \frac{1}{m}\mathbf{\Phi}_2^T\mathbf{\Phi}_1 & \left(\frac{\lambda}{\rho}\mathbf{I} + \frac{1}{m}\mathbf{\Phi}_2^T\mathbf{\Phi}_2\right) & \dots & \frac{1}{m}\mathbf{\Phi}_2^T\mathbf{\Phi}_p & (\frac{m}{\rho}-1)\mathbf{\Phi}_2^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{m}\mathbf{\Phi}_m^T\mathbf{\Phi}_1 & \frac{1}{m}\mathbf{\Phi}_p^T\mathbf{\Phi}_2 & \dots & \left(\frac{\lambda}{\rho}\mathbf{I} + \frac{1}{m}\mathbf{\Phi}_m^T\mathbf{\Phi}_m\right) & (\frac{m}{\rho}-1)\mathbf{\Phi}_m^T \\ -\frac{\rho}{m}\mathbf{\Phi}_1 & -\frac{\rho}{m}\mathbf{\Phi}_2 & \dots & -\frac{\rho}{m}\mathbf{\Phi}_m & \rho\mathbf{I} \end{bmatrix} \end{aligned}$$

Proof. An interested reader can find the proof of the proposition in Appendix B.2. □

This proposition moves ADMM from the realm of optimisation into the realm of iterative solvers, which may be much more familiar to some readers. In particular, we can recognise in the entries of $\mathbf{M}$ a combination of Jacobi and Gauss-Seidel methods. We can also formulate the similarity between ADMM and the linear smoother formulation describe in Sec. 5.1 as described in the following corollary.

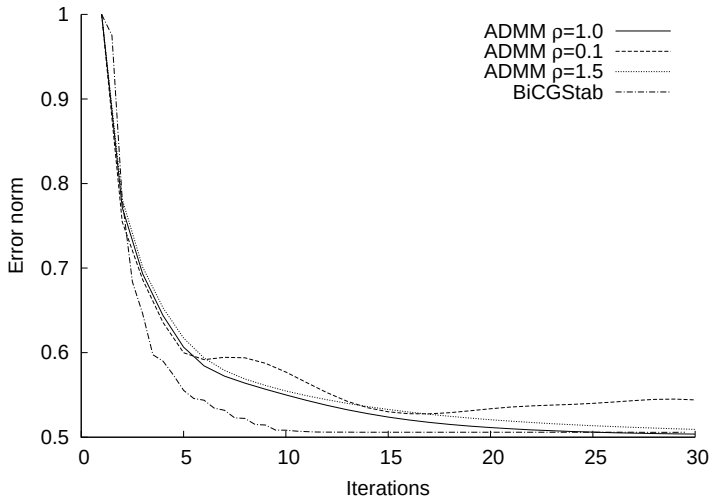


Figure 5.7.: Training error using ADMM and BiCGStab for regression on SDSS DR5 with 60 000 entries and random subspace projection ensemble with five 3-dimensional sparse grids with level 3 and $\lambda = 10^{-6}$. Source: (Khakhutskyy & Hegland, 2014b).

Corollary 5.1. *The system of linear equations* (5.43) *is equivalent to the augmented system of linear smoother equations*

$$\begin{bmatrix} \mathbf{I} & \mathbf{S}_1 & \cdots & \mathbf{S}_1 & (\frac{m}{\rho}-1)\mathbf{S}_1 \\ \mathbf{S}_2 & \mathbf{I} & \cdots & \mathbf{S}_2 & (\frac{m}{\rho}-1)\mathbf{S}_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{S}_m & \mathbf{S}_m & \cdots & \mathbf{S}_m & (\frac{m}{\rho}-1)\mathbf{S}_m \\ \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} & -\mathbf{I} \end{bmatrix} \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_m \\ \mathbf{f} \end{pmatrix} = \frac{m}{\rho} \begin{pmatrix} \mathbf{S}_1\mathbf{y} \\ \mathbf{S}_2\mathbf{y} \\ \vdots \\ \mathbf{S}_m\mathbf{y} \\ \mathbf{0} \end{pmatrix} \tag{5.44}$$

with $\mathbf{f} = m\bar{\mathbf{f}}$, $\mathbf{f}_j = \mathbf{\Phi}_j\mathbf{x}_j$, *and* $\mathbf{S}_j = \mathbf{\Phi}_j(\mathbf{\Phi}_j^T\mathbf{\Phi}_j + \lambda\frac{m}{\rho}\mathbf{I})^{-1}\mathbf{\Phi}_j^T$ *for* $j = 1, \ldots, m$.

Proof. An interested reader can find the proof in Appendix B.3. □

Note, Corollary 5.1 implies that, since in (5.44) the regularisation parameter is $\lambda \cdot m/\rho$ instead of only λ in (5.5), ADMM and linear smoother converge to the same solution only if $\rho = m$.

To compare ADMM and BiCGStab empirically, we use the photometric redshift prediction problem of the SDSS DR5 dataset introduced in Sec. 3.5.2. To create an additive model of the form (5.1) we randomly select three attributes out of available six and use them to fix a three-dimensional sparse grid model with level 3. We add the predictions from five such models together, which gives us an additive model. Figure 5.7 compares the results. Both fitting methods are comparable, although the BiCGStab-based backfitting method exhibits a faster convergence at the beginning.

5.2.3. Distributed Implementation

The idea of parallelising the ADMM algorithm for (generalised) additive models was initially suggested by Chu et al. (2013). Here, we address the parallelisation of the ADMM algorithm for hierarchical models such as sparse grids. The procedure is summarised in Alg. 15. Individual processors can compute the updates (5.35) independently. Once finished, they communicate the results of the products $\mathbf{\Phi}_j \mathbf{w}_j^{k+1}$ in order to compute the sum $\sum_{j=1}^{m} \mathbf{\Phi}_j \mathbf{w}_j^{k+1}$ in a distributed way. This distributed computation can be done by the MPI operation `Allreduce` already used in the parallel BiCGStab algorithm in Sec. 5.1.5. Since steps (5.36) and (5.37) are not computationally expensive and can be performed independently by each process, no further communication is required during individual iterations. Once ADMM converges, the sub-vectors $\mathbf{w}_j$ are collected using the MPI `Gatherv` method.

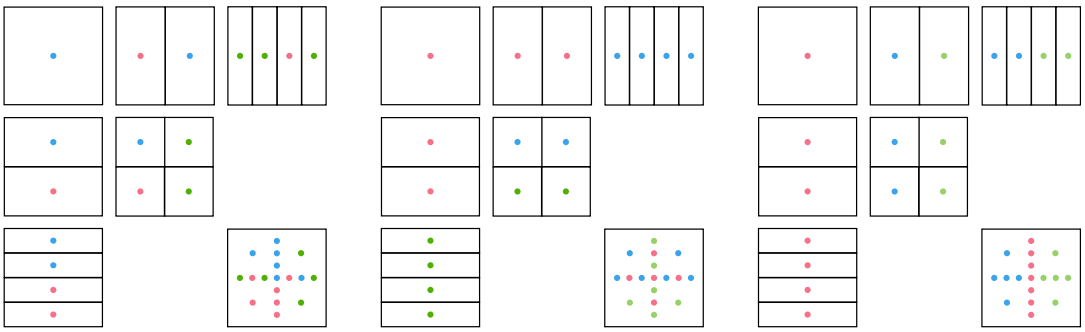
Algorithm 15: ADMM Algorithm for Sparse Grids (Khakhutskyy & Pflüger, 2014)

{Initialise variables:}
$\mathbf{w}_j \leftarrow \mathbf{0}$;
$\bar{\mathbf{f}} \leftarrow \frac{1}{m}\mathbf{y}$;
$\mathbf{u} \leftarrow \mathbf{0}$;
repeat
    update $\mathbf{x}_j$ by solving (5.35) simultaneously by all processors;
    `Allreduce`$(\mathbf{\Phi}_j \mathbf{w}_j, \mathbf{\Phi}\mathbf{w}, \text{SUM})$; {sum up the vectors $\mathbf{\Phi}_j \mathbf{w}_i$ from all processes}
    update $\bar{\mathbf{f}}$ using (5.36);
    update $\mathbf{u}$ using (5.37);
until convergence;
`Gatherv`$(\mathbf{w}_j, \mathbf{w})$; {collect partial results}
return $\mathbf{w}$;

Let us consider ADMM as a generic parallelisation algorithm for sparse grids. We can do this since the matrix $\mathbf{\Phi}$ can be divided into matrices $\mathbf{\Phi}_j$ arbitrary and in many different ways. Proposition 5.1 states that ADMM convergences if $\rho(\mathbf{M}^{-1}\mathbf{N}) < 1$. The spectral radius, however, is the subject to the division strategy. Therefore, it is fruitful to observe the numerical properties of the different grid splitting strategies. Let us consider the alternatives.

The simplest way to split basis functions into subsets is to do it *randomly* (see Fig. 5.8a). In general, random splitting offers the best load balancing since it does not constrain the structure of the subsets and allows to partition the grid function equally.

Unfortunately, random splitting of hierarchical structure of sparse grids may have undesired effects and slow convergence (Khakhutskyy & Pflüger, 2014). Instead, we can split the basis functions *level-wise* (see Fig. 5.8b). In this case



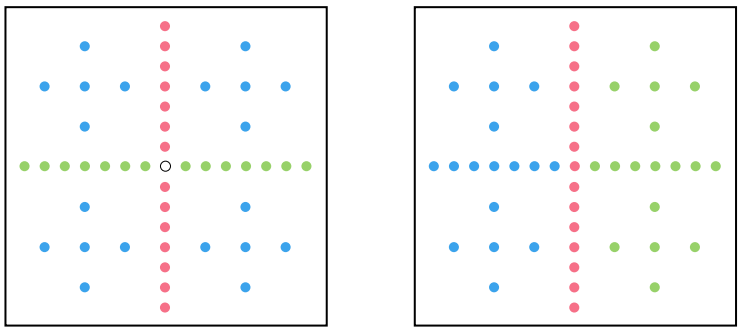
(a) Random (b) Levelwise (c) Recursive

Figure 5.8.: Sparse grid splitting schemes. Different colours identify the sparse grid functions assigned to the same processor.

the sparse grid functions that belong to the same or to adjacent diagonals in the subspace tableau are assigned to the same processor. This splitting strategy can lead to a significant reduction in the number of ADMM iterations, if the first update step (5.35) is performed serially and later update steps are performed concurrently (Khakhutskyy & Pflüger, 2014).

A more elaborate way to split a sparse grid is using the *recursive* splitting scheme (see Fig. 5.8c). This splitting approach roots in the idea that every sparse grid of level l and dimension d can be divided into two sparse grids of level $l-1$ and dimension d and one sparse grid of level l and dimension $d-1$ (Bungartz,1998). The recursive splitting retains the hierarchical structure inside the individual splits, ensuring that every step (5.35) by itself solves a small sparse grid problem. Alas, the size of such subgrids may vary leading to an imbalance. To minimise this imbalance we developed an algorithm that maintains the queue of subsets of the sparse grid basis functions, consequently splitting the largest subset in the queue until the required number of splits (the number of processors) is achieved (Khakhutskyy et al., 2014).

In fact, the recursive splitting scheme bears a similarity to splitting a sparse grid into ANOVA components f_0, f_i, f_{ij}, etc. In Fig. 5.9 we compare the results of the both splitting schemes for a two-dimensional sparse grid with level 4. Recursive splitting keeps one one-dimensional component (shown in red) and splits the other (horizontal) between two different processes (shown in blue and green). It also assigns the parts of the two-dimensional interaction to these processes. If these subtask would need to be divided further, the one-dimensional component will be kept and the two-dimensional interactions would be distributed between other two processors. We split the grid points among 3 processors and in recursive splitting scheme the processors get either 15 or 17 grid points (unknown parameters) to estimate, while in ANOVA scheme 14 or 20. The problem sizes for ANOVA splitting schemes are less balanced. This imbalance becomes more severe as the dimensionality or level grows. Figure 4.6 shows that for larger sparse grids the number of grid points in different interaction orders can differ



(a) ANOVA scheme (b) Recursive scheme

Figure 5.9.: Comparison of the recursive and additive splitting schemes for a sparse grid. Grid points of the same colours are assigned to the same processor. In recursive scheme the processors get either 15 or 17 grid points, in additive scheme either 14 or 20. The imbalance in the ANOVA scheme is higher and grows with the level.

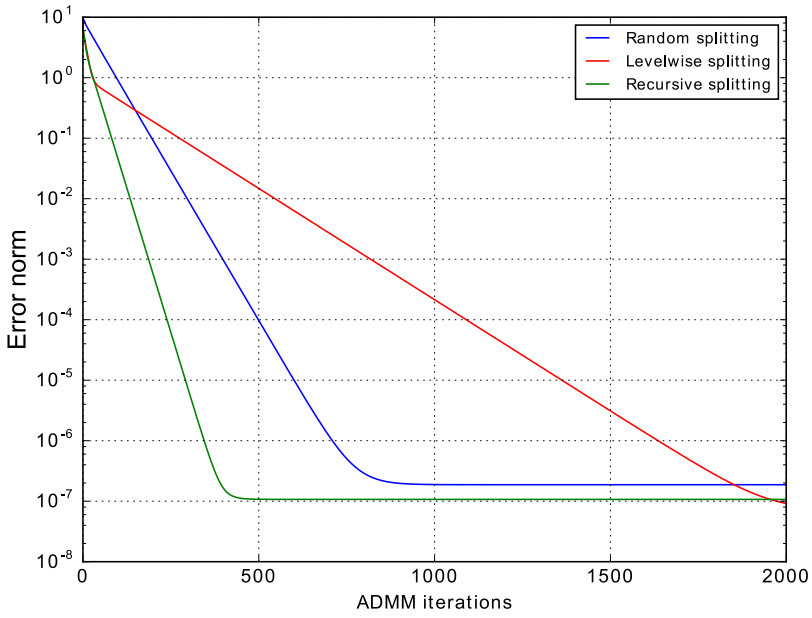
by orders of magnitudes. Hence, recursive scheme offers a good middle ground between retaining the structure of the subproblems of a sparse grid model and load-balancing of parallel algorithms.

We observed the superior properties of the recursive splitting (Khakhutskyy et al., 2014). Motivated by these results, together with Strauß (2016) we studied the connection between the convergence of ADMM and the different kinds of splitting. This study confirmed the advantage of the recursive splitting.

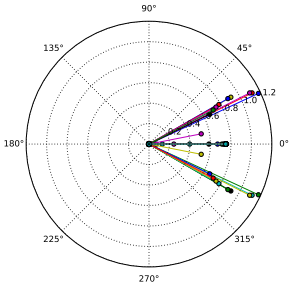
To illustrate this advantage we consider the problem of reconstructing the parameter vector $\mathbf{w}^*$ of a two-dimensional sparse grid with level 3 from 400 observations. The resulting matrix $\mathbf{\Phi} \in \mathbb{R}^{400\times 17}$ is distributed between 3 processors and for the purpose of comparison we use random, level-wise, and recursive splitting strategies as illustrated in Fig. 5.8. We use Alg. 15 with $\lambda = 10^{-7}$ and $\rho = 1$ to solve the underlying regression problem.

In Fig. 5.10a we see how the norm of the difference between $\mathbf{w}^*$ and the estimated parameter $\mathbf{w}^k$ decreases with proceeding ADMM iterations. The recursive splitting clearly outperforms the alternatives. Figures 5.10b– 5.10d compare the distribution of the eigenvalues of $\mathbf{M}^{-1}\mathbf{N}$. This comparison suggests the connection between these distributions and the convergence speed. For the recursive splitting we have $\rho(\mathbf{M}^{-1}\mathbf{N}) < 1$ (see Figure 5.10d) and therefore convergence is guaranteed. For two other splitting methods we have $\rho(\mathbf{M}^{-1}\mathbf{N}) \approx 1.2$. Note that, while $\rho(\mathbf{M}^{-1}\mathbf{N}) > 1$, ADMM with these splittings still converges, although the convergence is much solver.

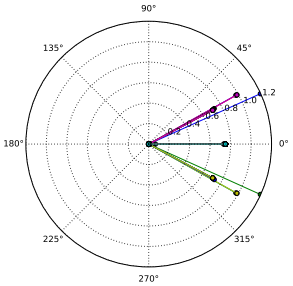
The ability to train models on a distributed computer architecture enables ADMM to train large sparse grid models. This, combined with a large number of supported regularisation functions, makes sparse grids ready for large-scale data analysis. We postpone the scalability tests of this (synchronous) distributed



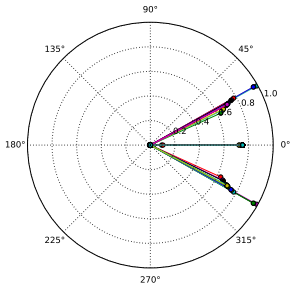
(a) Errors for different splitting strategies



(b) Eigenvalues of $\mathbf{M}^{-1}\mathbf{N}$ for random splitting



(c) Eigenvalues of $\mathbf{M}^{-1}\mathbf{N}$ for levelwise splitting



(d) Eigenvalues of $\mathbf{M}^{-1}\mathbf{N}$ for recursive splitting

Figure 5.10.: The declining error norm $\|\mathbf{w}^k - \mathbf{w}^*\|_2$ for Alg. 15 and the distribution of eigenvalues for different splitting strategies. Recursive splitting is clearly superior to others and has $\rho(\mathbf{M}^{-1}\mathbf{N}) < 1$. Source: (Strauß, 2016).

ADMM implementation until the next section, where it is compared with the asynchronous implementation.

5.2.4. Asynchronous Updates and Delayed Synchronisation

We can further accelerate the distributed ADMM algorithm by letting the processes to communicate asynchronously. This mitigates the need of one processor to wait for others before computing the step (5.36). On the other side, this means that different processes may work on different data, which is dangerous and may lead to instability. However, with a proper problem decomposition scheme this danger becomes negligible (Khakhutskyy et al., 2014).

Algorithm 16 shows the asynchronous version of the ADMM. The main difference appears in the Line 7 in Alg. 15 and Lines 8 to 11 in Alg. 16. In the first case, we can use `Allreduce` with summation implemented in MPI. In the second case, we update the sum vector $\mathbf{\Phi w}_{\text{shared}}$ with increments using the one-sided `Accumulate` operation. As the name suggests, this function adds the value of the increment vector $\Delta\mathbf{\Phi}_j\mathbf{w}_j$ to all instances of $\mathbf{\Phi w}_{\text{shared}}$ using direct memory access. This method is described by the MPI-2 standard and implemented in distributed shared memory libraries like GlobalArrays (Nieplocha et al., 1996).

Algorithm 16: ADMM Algorithm with Asynchronous Communication

1: {Initialise variables:}
2: define $\mathbf{\Phi w}_{\text{shared}}$; {shared over all processors}
3: $\mathbf{w}_j \leftarrow \mathbf{0}$;
4: $\bar{\mathbf{f}} \leftarrow \frac{1}{m}\mathbf{y}$;
5: $\mathbf{u} \leftarrow \mathbf{0}$;
6: **repeat**
7: $\quad\Delta\mathbf{\Phi}_j\mathbf{w}_j \leftarrow \Delta\mathbf{\Phi}_j\mathbf{w}_j - \mathbf{\Phi}_j\mathbf{w}_j$;
8: $\quad$update $\mathbf{w}_j$ by solving (5.35) simultaneously in all processes;
9: $\quad\Delta\mathbf{\Phi}_j\mathbf{w}_i \leftarrow \Delta\mathbf{\Phi}_j\mathbf{w}_j + \mathbf{\Phi}_j\mathbf{w}_j$;
10: $\quad$`Accumulate` $(\Delta\mathbf{\Phi}_j\mathbf{w}_j, \mathbf{\Phi w}_{\text{shared}})$; {sum up the vectors $\Delta\mathbf{\Phi}_j\mathbf{w}_j$ from all processes}
11: $\quad$update $\bar{\mathbf{f}}$ using (5.36);
12: $\quad$update $\mathbf{u}$ using (5.37);
13: **until** $\|\mathbf{\Phi w} - \mathbf{u}\|_2^2 \leq$ convergence threshold; {training error is one possible convergence criterion}
14: `Gatherv`$(\mathbf{w}_j, \mathbf{w})$; {collect partial results}
15: **return** $\mathbf{w}$;

In (Strauß, 2016) we studied the sufficient conditions for the asynchronous convergence of Alg. 16. Let $|\mathbf{M}^{-1}\mathbf{N}|$ be a matrix with absolute values of the entries of $\mathbf{M}^{-1}\mathbf{N}$. Summarising the results from (Strauß, 2016), we can say that, for Alg. 16 to converge in a totally asynchronous setting, it is sufficient to

have $\rho(|\mathbf{M}^{-1}\mathbf{N}|) < 1$. The prove of conditions for asynchronous convergence of a parallel algorithm is not straightforward. Since the delays in the communication between individual processors can be arbitrary small, the required theoretical conditions are often very strong. So in our case $\rho(|\mathbf{M}^{-1}\mathbf{N}|) < 1$ is rarely given. This by no means implies that Alg. 16 would necessarily diverge otherwise but rather that there can be an example setting where it could diverge.

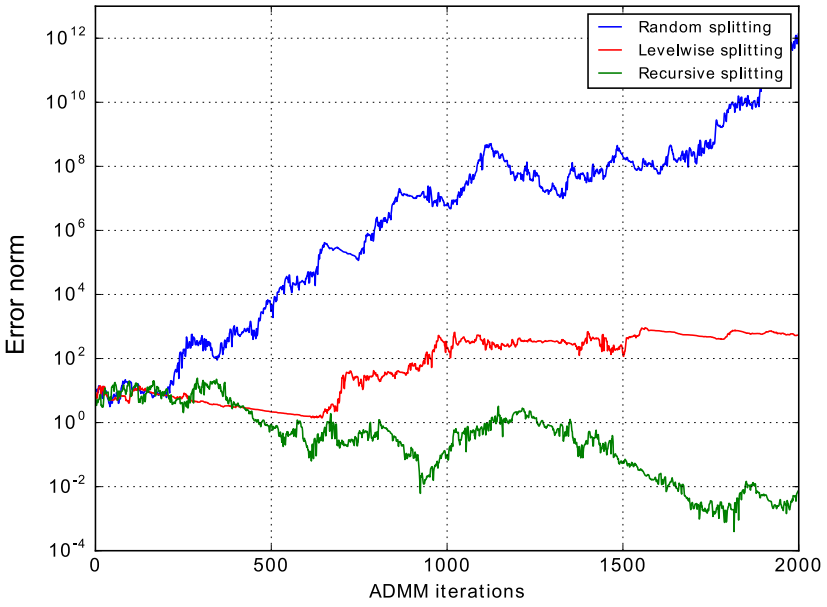
We repeat the reconstruction of the parameters of a two-dimensional sparse grid from 400 samples presented in the previous section but this time we simulate a totally asynchronous random communication. Figure 5.11a illustrates the change of the error norm $\|\mathbf{w}^k - \mathbf{w}^*\|_2$ with proceeding ADMM iterations. The error in the recursive splitting scenario slowly declines while the two adversary methods diverge. Figures 5.11b, 5.11c, and 5.11d illustrate the distribution of the absolute values of eigenvalues of $|\mathbf{M}^{-1}\mathbf{N}|$, which is not to be confused with the distributions in Figures 5.10b, 5.10c, and 5.10d where the matrix is $\mathbf{M}^{-1}\mathbf{N}$ can have positive or negative entries. The spectral radius in all three cases in Figures 5.11b, 5.11c, and 5.11d is larger than 1. For recursive splitting scheme it is still comparably smaller and this may cause the convergence.

In practice, the extreme cases of totally asynchronous communication are very rare. Apart from the cases where the hardware fails completely, the results from every processor are usually distributed within some realistic time frame. If we can assume that such time frame exists and its finite, we speak of *partially asynchronous* algorithms.[3] For partially asynchronous algorithms the theory becomes more complex while the practice more simple.

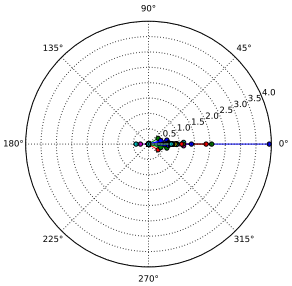
Once again we consider the reconstruction of the parameters of a two-dimensional sparse grid from 400 examples. Again, we use Alg. 16 but this time we assume (and enforce) that the processors exchange information within 5 update steps. Figure 5.12 illustrates the change of the error norm $\|\mathbf{w}^k - \mathbf{w}^*\|_2$ with proceeding ADMM iterations. The results for this partially asynchronous experiment stand between the results for the synchronous and the totally asynchronous experiments: Random and recursive splittings converge, although slower than in the synchronous case. Levelwise splitting, which exhibits the slowest convergence in the synchronous case, diverges in the partially asynchronous setting. The recursive splitting converges slower than in the synchronous setting, faster than in the totally asynchronous, and faster than the adversaries.

To compare the synchronous and asynchronous ADMM algorithms in a realistic scenario, we use the SDSS DR5 photometric redshift prediction problem introduced in Sec. 3.5.2. The performance of the different splitting schemes is similar to that shown in examples above and therefore we focus only on the recursive splitting scheme in synchronous (Alg. 15) and asynchronous (Alg. 16) settings.

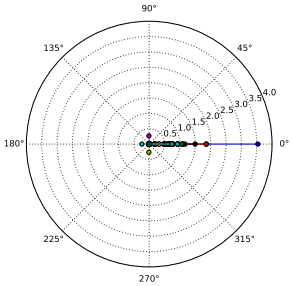
[3]Obviously, partial asynchronicity changes only assumptions about properties of the computational environment and not the algorithm itself.



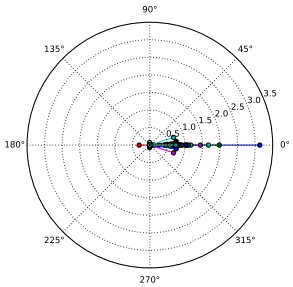
(a) Errors for different splitting strategies



(b) Eigenvalues of $|\mathbf{M}^{-1}\mathbf{N}|$ for random splitting



(c) Eigenvalues of $|\mathbf{M}^{-1}\mathbf{N}|$ for levelwise splitting



(d) Eigenvalues of $|\mathbf{M}^{-1}\mathbf{N}|$ for recursive splitting

Figure 5.11.: The declining error norm $\|\mathbf{w}^k - \mathbf{w}^*\|_2$ for Alg. 16 in totally asynchronous setting and the distribution of eigenvalues for different splitting strategies. Recursive splitting has the smaller spectral radius and is the only method to with declining error. Source: (Strauß, 2016).

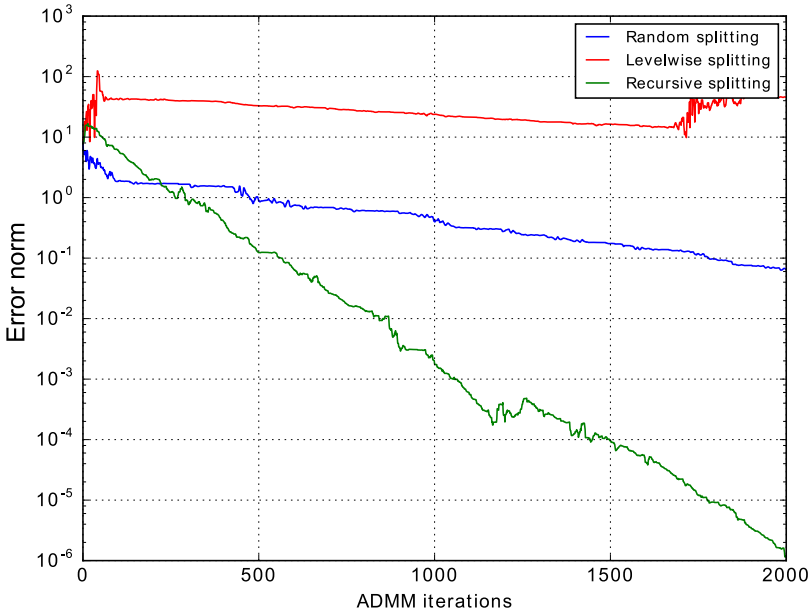
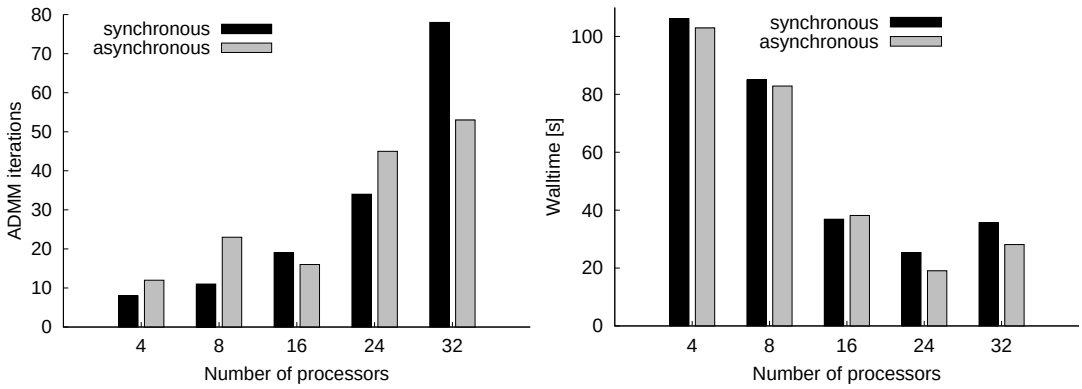


Figure 5.12.: The declining error norm $\|\mathbf{w}^k - \mathbf{w}^*\|_2$ for Alg. 16 in partially asynchronous setting. The results lie between the synchronous setting in Fig. 5.10a and totally asynchronous setting in Fig. 5.11a. Adopted from (Strauß, 2016).



(a) ADMM iterations (b) Time

Figure 5.13.: Comparison of the number of ADMM iterations and elapsed time for the synchronous and asynchronous distributed settings. Asynchronous version is superior for more than 24 processors. Source: (Khakhutskyy et al., 2014).

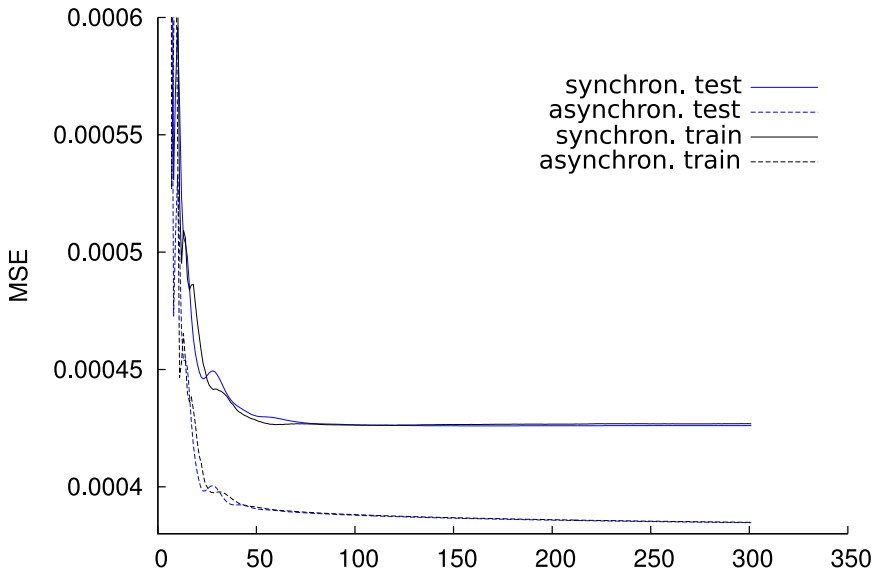


Figure 5.14.: Convergence of ADMM algorithm based on grid splitting for the synchronous and asynchronous settings. The results are indistinguishable. Adopted from (Khakhutskyy et al., 2014).

Figure 5.13 compares the scalability of the synchronous and asynchronous ADMM methods. It is characteristic for ADMM that while the number of iterations increases with the number of processors due to the need for agreement between more parties (partitions), the time for matrix assembly and factorisation as well as the time of individual iterations decreases. As the number of processors grows, the advantage of the asynchronous implementation becomes apparent: between 4 and 16 processors both methods finish at a comparable time, for more than 16 processors the asynchronous algorithm is faster even if it requires to perform more ADMM iterations (as for 24 processors). Figure 5.14 demonstrates the decline of the mean squared error on the training and test datasets for a run with 32 processors. The error curves are virtually identical.

As the number of processors grows, the efficiency of the distributed ADMM becomes a subject to a trade-off between decreasing computation time and increasing communication overhead. This overhead can be reduced using asynchronous communication patterns. Unfortunately, we observe that asynchronous algorithms need not to converge in every case and the existing theory is fuzzy on this matter. Our experience and numerical experiments showed, however, that recursive splitting performs remarkably robust in an asynchronous setting. We hypothesise that recursive sparse grid splitting maintains a better internal hierarchical structure, which improves the performance. The numerical evaluation of the spectrum of the contraction operator $\mathbf{M}^{-1}\mathbf{N}$ supports this hypothesis.

In this chapter we discussed two methods for training models that can be decomposed into a sum of smaller models. While the developed methods had different motivations, under certain circumstances, these methods are connected. This connection was unknown before. Both methods can be parallelised and hence can be used for large-scale learning problems. Both methods have their advantages. BiCGStab can exhibit a faster convergence, especially at the beginning, it can also be accelerated with preconditioning. ADMM supports a large number of different regularisation functions and asynchronous communication. Hence, the final decision, which method to use for a particular problem, should be made by weighing these advantages against each other.

Sparse grids is a prominent example of models that can be decomposed into a sum of smaller models. Among many different ways to decompose a sparse grid, we advise to use either the ANOVA decomposition discussed in the previous chapter or the recursive decomposition discussed in this chapter. These splitting schemes maintain a meaningful structure inside of individual subproblems. The use of the ANOVA decomposition reduces the size of the dataset, as only a subset of input dimensions is used by an ANOVA component. Recursive decomposition, on the other hand, offers a better load-balancing. Additionally, the number of subproblems in recursive splitting is independent of the number of ANOVA component, which is a big advantage for distributed computing.

6. Conclusions and Future Work

> *I would rather have questions that can't be answered than answers which can't be questioned.*
>
> — attributed to Richard Feynman

We presented new techniques for supervised learning with sparse grids that make them better suitable for large-scale learning and handling the volume of Big Data. This aptitude is achieved by improving the model's adaptability to a low-dimensional manifold in data and by discounting unimportant interactions between input variables. To this end, we extended the notion of parsimony for hierarchical sparse grid models and developed a learning algorithm that simultaneously optimises the model parameters and the model structure to befit the problem at hand.

With seven original propositions, four lemmas, one theorem, and one corollary, a significant portion of this dissertation is dedicated to the theoretical treatment of the presented ideas. This attention to theory was motivated by an aspiration to provide the models and algorithms derived in this dissertation with a solid theoretical foundation and to make the results generalisable.

The main contributions of this thesis can be summarised as follows:

- The interpretation of sparse grids as Bayesian linear basis models, presented in Chapter 2, makes the sparse grid techniques better accessible to machine learning experts and statisticians. It also provides the probabilistic meaning to various regularisation functions, making them easier to interpret and to adjust to different model assumptions.

- The kernel trick algorithm developed and analysed in Chapter 2 integrates the sparse grid models into the kernel machines. As the complexity of our kernel trick is only quadratic in the number of attributes, it effectively breaks the curse of dimensionality making the combination of sparse grids and kernel machines especially attractive for high-dimensional problems with moderately many data points.

- In Chapters 2 and 3 we posed the refinement of sparse grids as an optimisation problem, which gave us a theoretical framework for development and analysis of new problem-specific refinement strategies. Using this framework, we derived a new spatially-dimension-adaptive procedure that fits to a low-dimensional manifold in data more effectively than the previous methods.

- In Chapter 4 we derived a new ANOVA-like notion for effective dimensionality. It provides new criteria for adaptive sparse grids to fit the additive structure of a problem at hand. This new notion bridges the gap in the existing theory between the classical functional ANOVA decomposition and the anchored ANOVA. It provides error bounds for supervised learning problems (like classical ANOVA decomposition) while at the same time it can be efficiently estimated for standard sparse grid basis functions, e.g. linear or modified linear basis functions, and is not limited to prewavelet basis intractable in high dimensions (like classical ANOVA).

- Finally, in Chapter 5 we developed two algorithms for parallel training of sparse grid models. The choice of a particular training method depends on the regularisation function that has to be used. We implemented and analysed these algorithms for synchronous and asynchronous communication patterns and derived a sparse grid partitioning scheme for better convergence.

Sparse grids should not be considered in isolation but rather embedded in the family of Bayesian linear basis models with a specific feature transformation. The benefits of this paradigm shift are numerous: On the one side, we can apply the knowledge from other Bayesian linear basis models to sparse grid models improving interpretability (as in Chapter 2) and adopting the rich theoretical and algorithmic groundwork (as in Chapter 3). On the other side, with this perspective in mind, the new algorithms developed for sparse grids can be readily applied to other linear regression models and linear smoothers (as in Chapter 5). We can only speculate what further improvements this synergy can produce.

The principal advantage of the sparse grid models lies undoubtedly in their adaptability. This, however, depends critically on the quality of the refinement strategies used for different problems. In turn, the number of possible refinement strategies is limited only by the number of PhD students dedicated to invent a new one. By posing the refinement process as an active-set optimisation problem with specific sparsity-inducing regularisation functions, I intended to formalise the frame of thinking and to streamline the invention process.

However, as every optimisation algorithm considers only the training data, a zealous optimisation unavoidably leads to deterioration of the model's generalisation performance. Right now, the premature overfitting is prevented by refining a larger number of grid points at once and by including all hierarchical ancestors of a grid point. The trade-off between the adaptivity and generality of the sparse grid models—or between exploration and exploitation of the hypothesis space—is still poorly understood. The challenge of developing a regularisation procedure that effectively manages this trade-off is waiting for its champion.

Acknowledgements

What seems to us as bitter trials are often blessings in disguise for which we are later, in the fullness of time and understanding, very grateful for!

— attributed to Oscar Wilde

It is now time for me to give credit where credit is due. First and foremost, I am privileged and grateful to be working with Hans-Joachim Bungartz and Markus Hegland in the past several years. Much of what I know about scientific research and writing I learned from them. I am indebted to my supervisor, Hans-Joachim Bungartz, for giving me the opportunity to pursue my doctoral degree and for the great atmosphere at his chair. I am thankful to Markus Hegland for taking me as a doctoral candidate in his IAS Hans Fischer Fellowship, for sharing his ideas, his knowledge, and his time with me as well as for his hospitality during my research stays at the Australian National University.

Over the course of the past years I was happy to work in a great group with talented and supportive young researchers. I am grateful to Dirk Pflüger for introducing me into data mining with sparse grids back in 2008 and for his support and guidance since the beginning of my doctoral program; to my colleagues and collaborators Benjamin Peherstorfer, Benjamin Uekermann, Christoph Kowitz, Gerrit Buse, Kilian Röhner, and Steffen Seckler from TUM and Matthias Wong from ANU for lending me a listening ear, a helping hand, and for sharing their valuable ideas and opinions with me.

I was honoured to advise a great number of young and talented students: Carsten Uphoff, Florian Klemt, Florian Zipperle, Gunnar Koenig, Johann Maier, Karthikeya Sampa Subbarao, Kevin Strauß, Kilian Batzner, Maxim Schmidt, are Michael Lettrich. They were testing, extending, and challenging my ideas, most of which have failed but some did not. I hope these students were able to learn from me, I certainly learned from them a lot.

I am thankful to the Institute for Advanced Study of Technische Universität München for the financial support of my doctoral program and for providing the first-rate infrastructure and facilities. I am thankful personally to Stefanie Merz and Anna Fischer for their organisational support and workflow management at the IAS.

Finally, I want to thank my parents for their continuous support and patience.

A. Results from Linear Algebra and Probability Theory

This appendix briefly summarises the standard theoretical results from linear algebra and probability theory used throughout the dissertation.

Lemma A.1 (Blockwise inversion)**.** *Let* $\mathbf{A}$ *and* $\mathbf{D}$ *be square invertible matrices. Let* $\mathbf{B}$ *and* $\mathbf{C}$ *be matrices such that* $\mathbf{A}$ *and* $\mathbf{BD}^{-1}\mathbf{C}$ *as well as* $\mathbf{D}$ *and* $\mathbf{CA}^{-1}\mathbf{B}$ *have the same dimensions. Then following equalities hold:*

$$\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}^{-1} = \begin{bmatrix}\mathbf{A}^{-1}{+}\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}{-}\mathbf{CA}^{-1}\mathbf{B})^{-1}\mathbf{CA}^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}{-}\mathbf{CA}^{-1}\mathbf{B})^{-1}\\ -(\mathbf{D}{-}\mathbf{CA}^{-1}\mathbf{B})^{-1}\mathbf{CA}^{-1} & (\mathbf{D}{-}\mathbf{CA}^{-1}\mathbf{B})^{-1}\end{bmatrix} \tag{A.1}$$

$$= \begin{bmatrix}(\mathbf{A}{-}\mathbf{BD}^{-1}\mathbf{C})^{-1} & -(\mathbf{A}{-}\mathbf{BD}^{-1}\mathbf{C})^{-1}\mathbf{BD}^{-1}\\ -\mathbf{D}^{-1}\mathbf{C}(\mathbf{A}{-}\mathbf{BD}^{-1}\mathbf{C})^{-1} & \mathbf{D}^{-1}{+}\mathbf{D}^{-1}\mathbf{C}(\mathbf{A}{-}\mathbf{BD}^{-1}\mathbf{C})^{-1}\mathbf{BD}^{-1}\end{bmatrix}. \tag{A.2}$$

Corollary A.1 (Sherman–Morrison–Woodbury formula)**.** *Let* $\mathbf{A}$ *and* $\mathbf{D}$ *be square invertible matrices. Let* $\mathbf{B}$ *and* $\mathbf{C}$ *be matrices such that* $\mathbf{A}$ *and* $\mathbf{BD}^{-1}\mathbf{C}$ *as well as* $\mathbf{D}$ *and* $\mathbf{CA}^{-1}\mathbf{B}$ *have the same dimensions. Then following equalities hold:*

$$(\mathbf{A}-\mathbf{BD}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}-\mathbf{CA}^{-1}\mathbf{B})^{-1}\mathbf{CA}^{-1}, \tag{A.3}$$

$$(\mathbf{D}-\mathbf{CA}^{-1}\mathbf{B})^{-1}\mathbf{CA}^{-1} = \mathbf{D}^{-1}\mathbf{C}(\mathbf{A}-\mathbf{BD}^{-1}\mathbf{C})^{-1}. \tag{A.4}$$

Lemma A.2 (Marginal and Conditional Gaussians (Bishop, 2006, p. 93))**.** *Given a marginal Gaussian distribution for* $\mathbf{x}$ *and a conditional Gaussian distribution for* $\mathbf{y}$ *given* $\mathbf{x}$ *in the form*

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{D}^{-2}), \tag{A.5}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{Ax}+\mathbf{w}, \mathbf{B}^{-1}), \tag{A.6}$$

the marginal distirbtuion of $\mathbf{y}$ *and the conditional distribution of* $\mathbf{x}$ *given* $\mathbf{y}$ *are given by*

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu}+\mathbf{w}, \mathbf{B}^{-1}+\mathbf{AD}^{-1}\mathbf{A}^{T}), \tag{A.7}$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\mathbf{A}^{T}\mathbf{B}(\mathbf{y}-\mathbf{w})+\mathbf{D}\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{A.8}$$

where

$$\boldsymbol{\Sigma} = (\mathbf{D} + \mathbf{A}^T\mathbf{B}\mathbf{A})^{-1}. \tag{A.9}$$

B. Proofs

This appendix presents the proofs for some original theoretical results in the main part of the dissertation.

B.1. Theorem 3.1

To prove Theorem 3.1 we use two auxiliary propositions. First, we establish a relationship between projections onto nested subspaces in Proposition B.1. Then in Proposition B.2 we show that J^- can be defined as a negative residual norm using such a projection.

Proposition B.1. *Let V_1, V_2, and W be Euclidean vector spaces, $V_1 \subset V_2$, and let P_{V_1}, P_{V_2}, and P_W be orthogonal projections onto the corresponding spaces. Then*

$$\|(P_{V_1+W} - P_{V_1})\mathbf{y}\| \geq \|(P_{V_2+W} - P_{V_2})\mathbf{y}\|. \tag{B.1}$$

Proof. The vector space W can be either a part of V_2 or a part of its orthogonal complement $V_2^\perp$ or consist of parts in V_2 and parts in its orthogonal complement. We consider these three cases separately.

Case 1. *If $W \subseteq V_2$ then we have*

$$\|(P_{V_1+W} - P_{V_1})\mathbf{y}\| \geq \|(P_{V_2+W} - P_{V_2})\mathbf{y}\| = 0.$$

Case 2. *If $W \perp V_2$ then*

$$\|(P_{V_1+W} - P_{V_1})\mathbf{y}\| = \|P_W\mathbf{y}\| = \|(P_{V_2+W} - P_{V_2})\mathbf{y}\|.$$

Case 3. *Finally, if $W = W_1 + W_2$ such that $W_1 \subseteq V_2^\perp$ and $W_2 \subseteq V_2$ then*

$$\begin{aligned}
\|(P_{V_1+W_1+W_2} - P_{V_1})\mathbf{y}\| &= \|(P_{V_1+W_2}+P_{W_1}-P_{V_1})\mathbf{y}\| = \|(P_{V_1+W_2}-P_{V_1})\mathbf{y}+P_{W_1}\mathbf{y}\| \\
&= \sqrt{\|(P_{V_1+W_2} - P_{V_1})\mathbf{y}\|^2 + \|P_{W_1}\mathbf{y}\|^2} \geq \|P_{W_1}\mathbf{y}\|. \\
\|(P_{V_2+W_1+W_2} - P_{V_2})\mathbf{y}\| &= \|(P_{V_2+W_2} + P_{W_1} - P_{V_2})\mathbf{y}\| = \|P_{W_1}\mathbf{y}\|
\end{aligned}$$

and together we have

$$\|(P_{V_1+W_1+W_2} - P_{V_1})\mathbf{y}\| \geq \|(P_{V_2+W_1+W_2} - P_{V_2})\mathbf{y}\|.$$

□

Let $\mathcal{I} : \{1, \ldots, |\Gamma|\} \to \Gamma$ be a bijective mapping that enumerates the elements of Γ. For every sparse grid index g we define a vector in $\mathbb{R}^{N+|\Gamma|}$ as

$$\mathbf{v}_g := \left(\phi_g(\mathbf{x}_1), \ldots, \phi_g(\mathbf{x}_N), \sqrt{\lambda N}\delta_{\mathcal{I}(g),1}, \sqrt{\lambda N}\delta_{\mathcal{I}(g),2}, \ldots, \sqrt{\lambda N}\delta_{\mathcal{I}(g),|\Gamma|}\right)^T, \quad \text{(B.2)}$$

with $\delta_{i,j}$ being the Kronecker delta. Then

$$V_X := \operatorname{span}\{\mathbf{v}_g\}_{g \in X} \quad \text{(B.3)}$$

is a vector space embedding of X. We appropriately extend the target vector $\mathbf{y}$ to the dimensionality $N + |\Gamma|$ by appending $|\Gamma|$ zeros:

$$\tilde{\mathbf{y}} := \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}. \quad \text{(B.4)}$$

Proposition B.2. *Assume that V_X is defined as in* (B.3) *and $\tilde{\mathbf{y}}$ is defined as in* (B.4)*. Then for the orthogonal projection P_X that satisfies*

$$\begin{aligned} P_X : \mathbb{R}^{N+|\Gamma|} &\to V_X \\ \tilde{\mathbf{y}} &\mapsto \operatorname*{arg\,min}_{\mathbf{f} \in V_X} \|\mathbf{f} - \tilde{\mathbf{y}}\|^2 \end{aligned} \quad \text{(B.5)}$$

we have

$$J^-(X) = -\|P_X\tilde{\mathbf{y}} - \tilde{\mathbf{y}}\|^2. \quad \text{(B.6)}$$

Proof. For any $\mathbf{f} \in V_X$ there is a set of linear combination coefficients $\mathbf{w} := \{w_g\}_{g \in X}$ such that

$$\mathbf{f} = \sum_{g \in X} w_g \mathbf{v}_g.$$

The squared norm of the difference between $\mathbf{f}$ and $\tilde{\mathbf{y}}$ is equal to the value of the cost function $J(\mathbf{w}; X)$ defined in (2.30):

$$\|\mathbf{f} - \tilde{\mathbf{y}}\|^2 = \sum_{i=1}^{N} \left(\sum_{g \in X} w_g \phi_g(\mathbf{x}_i) - y_i \right)^2 + \lambda N \sum_{g \in X} (w_g - 0)^2.$$

Hence, the projection $P_X\tilde{\mathbf{y}}$ that minimises (B.5) corresponds to $\mathbf{w}^\star$ that solves the minimisation problem in (3.3). This leads directly to (B.6). □

Now everything is ready for the actual proof. Let us start by restating the theorem formulation.

Theorem 3.1. *The function J^- defined in* (3.3) *is a monotone non-decreasing submodular function. The submodular optimisation problem with cardinality constraint*

$$\max_{G \subset \Gamma, |G| \leq m} J^-(G) \quad \text{(3.5 revisited)}$$

is NP-hard. It can be solved with a greedy algorithm such that, if $\widehat{G}$ is the greedy solutions and G^* is the optimal solution, we have

$$\frac{J^-(\widehat{G})}{J^-(G^*)} \geq 1 - \left(\frac{m-1}{m}\right)^m \geq \left(1 - \frac{1}{e}\right) \approx 0.632. \qquad \text{(3.6 revisited)}$$

This boundary is tight unless P=NP.

Proof. Suppose that $X \subset \Gamma$, V_X is defined as in (B.3), $\tilde{\mathbf{y}}$ is defined as in (B.4), and P_X is defined as in (B.5). For simplicity of the notation we also introduce the residual vector

$$R_X := P_X\tilde{\mathbf{y}} - \tilde{\mathbf{y}}. \tag{B.7}$$

Let Y be another sparse grid index-set such that $X \subseteq Y \subset \Gamma$, and let $s \in \Gamma \setminus Y$ be a new sparse grid index. Analogously, we define the projections P_Y, P_s, $P_{X\cup\{s\}}$, $P_{Y\cup\{s\}}$ as well as the residual vectors R_Y, R_s, $R_{X\cup\{s\}}$, $R_{Y\cup\{s\}}$.

We are going to show the property of diminishing returns

$$J^-(s \mid X) \geq J^-(s \mid Y),$$

where $J^-(\cdot \mid \cdot)$ is a marginal gain function of J^- as defined in (2.37).

We begin by writing the relation (B.1) from Lemma B.1 as

$$\|R_{X\cup\{s\}} - R_X\|^2 = \|(P_{X\cup\{s\}} - P_X)\tilde{\mathbf{y}}\|^2 \geq \|(P_{Y\cup\{s\}} - P_Y)\tilde{\mathbf{y}}\|^2 = \|R_{Y\cup\{s\}} - R_Y\|^2. \tag{B.8}$$

We now consider the norms $\|R_{X\cup\{s\}} - R_X\|^2$ and $\|R_{Y\cup\{s\}} - R_Y\|^2$ in (B.8). We can rewrite the inequality as

$$\|R_{X\cup\{s\}}\|^2 - 2\langle R_{X\cup\{s\}}, R_X\rangle + \|R_X\|^2 \geq \|R_{Y\cup\{s\}}\|^2 - 2\langle R_{Y\cup\{s\}}, R_Y\rangle + \|R_Y\|^2.$$

Since P_X and $P_{X\cup\{s\}}$ are orthogonal projections, we have $P_{X\cup\{s\}}(P_X\mathbf{a}) = P_X\mathbf{a}$ and $\langle R_{X\cup\{s\}}, P_{X\cup\{s\}}\mathbf{a}\rangle = 0$ for any $\mathbf{a} \in \mathbb{R}^{N+|\Gamma|}$. With this in mind, we can rewrite the inner product as

$$\begin{aligned}
\langle R_{X\cup\{s\}}, R_X\rangle &= \langle (P_{X\cup\{s\}} - I)\tilde{\mathbf{y}}, (P_X - I)\tilde{\mathbf{y}}\rangle \\
&= \langle P_{X\cup\{s\}}\tilde{\mathbf{y}}, P_X\tilde{\mathbf{y}}\rangle - \langle P_{X\cup\{s\}}\tilde{\mathbf{y}}, \tilde{\mathbf{y}}\rangle - \langle \tilde{\mathbf{y}}, P_X\tilde{\mathbf{y}}\rangle + \langle \tilde{\mathbf{y}}, \tilde{\mathbf{y}}\rangle \\
&= \langle \tilde{\mathbf{y}}, \tilde{\mathbf{y}}\rangle - \langle P_{X\cup\{s\}}\tilde{\mathbf{y}}, \tilde{\mathbf{y}}\rangle + \langle R_{X\cup\{s\}}, P_X\tilde{\mathbf{y}}\rangle \\
&= \langle \tilde{\mathbf{y}}, \tilde{\mathbf{y}}\rangle - \langle P_{X\cup\{s\}}\tilde{\mathbf{y}}, \tilde{\mathbf{y}}\rangle + \langle R_{X\cup\{s\}}, P_{X\cup\{s\}}P_X\tilde{\mathbf{y}}\rangle \\
&= \langle \tilde{\mathbf{y}}, \tilde{\mathbf{y}}\rangle - \langle P_{X\cup\{s\}}\tilde{\mathbf{y}}, \tilde{\mathbf{y}}\rangle.
\end{aligned} \tag{B.9}$$

Similarly, we obtain

$$\|R_{X\cup\{s\}}\|^2 = \langle R_{X\cup\{s\}}, R_{X\cup\{s\}}\rangle = \langle \tilde{\mathbf{y}}, \tilde{\mathbf{y}}\rangle - \langle P_{X\cup\{s\}}\tilde{\mathbf{y}}, \tilde{\mathbf{y}}\rangle. \tag{B.10}$$

Using identities (B.9) and (B.10), we can rewrite the expression $\|R_{X\cup\{s\}} - R_X\|^2$ as

$$\begin{aligned}\|R_{X\cup\{s\}} - R_X\|^2 &= \|R_{X\cup\{s\}}\|^2 - 2\langle R_{X\cup\{s\}}, R_X\rangle + \|R_X\|^2 \\ &= \langle\tilde{\mathbf{y}},\tilde{\mathbf{y}}\rangle - \langle P_{X\cup\{s\}}\tilde{\mathbf{y}},\tilde{\mathbf{y}}\rangle - 2\langle\tilde{\mathbf{y}},\tilde{\mathbf{y}}\rangle + 2\langle P_{X\cup\{s\}}\tilde{\mathbf{y}},\tilde{\mathbf{y}}\rangle + \|R_X\|^2 \\ &= \|R_X\|^2 - \big(\langle\tilde{\mathbf{y}},\tilde{\mathbf{y}}\rangle - \langle P_{X\cup\{s\}}\tilde{\mathbf{y}},\tilde{\mathbf{y}}\rangle\big) = \|R_X\|^2 - \|R_{X\cup\{s\}}\|^2.\end{aligned} \tag{B.11}$$

The expression $\|R_{Y\cup\{s\}} - R_Y\|^2$ can be rewritten analogously. Hence, plugging it back into (B.8), we get

$$-\|R_{X\cup\{s\}}\|^2 + \|R_X\|^2 \geq -\|R_{Y\cup\{s\}}\|^2 + \|R_Y\|^2$$

and then with (B.6):

$$J^-(X\cup\{s\}) - J^-(X) \geq J^-(Y\cup\{s\}) - J^-(Y).$$

Hence, $J^-(X)$ satisfies the diminishing returns property from Definition 3.1.

Now we show that $J^-(X)$ is monotonically non-decreasing. Suppose that $J^-(X)$ is not monotonically non-decreasing and there is $s \in \Gamma \setminus Y$ such that

$$J^-(X\cup\{s\}) - J^-(X) < 0.$$

From (B.6) and (B.7) this would be equivalent to $\|R_X\|^2 < \|R_{X\cup\{s\}}\|^2$. In this case, there would be a vector $\mathbf{v} \in V_X$ such that $\|\mathbf{v} - \tilde{\mathbf{y}}\|^2 < \|\mathbf{v}' - \tilde{\mathbf{y}}\|^2$ for every $\mathbf{v}' \in V_{X\cup\{s\}}$. This is a contradiction, since $V_X \subset V_{X\cup\{s\}}$.

Finally, the relationship (3.6) was shown in (Nemhauser, Wolsey, & Fisher, 1978) for all submodular functions and, hence, can be applied to J^- as well. The tightness of the boundary follows from (Feige, 1998). □

B.2. Proposition 5.1

Let

$$\mathbf{x} = \begin{pmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_m \\ \bar{\mathbf{f}} \end{pmatrix} \quad \textit{and} \quad \mathbf{b} = \frac{1}{\rho}\begin{pmatrix} \boldsymbol{\Phi}_1^T\mathbf{y} \\ \boldsymbol{\Phi}_2^T\mathbf{y} \\ \vdots \\ \boldsymbol{\Phi}_m^T\mathbf{y} \\ \mathbf{0} \end{pmatrix}.$$

The ADMM algorithm (5.38)– (5.40) *corresponds to a stationary method of the form*

$$\mathbf{M}\mathbf{x}^{k+1} = \mathbf{N}\mathbf{x}^k + \mathbf{b}$$

with

$$\mathbf{M} = \begin{bmatrix} (\frac{\lambda}{\rho}\mathbf{I} + \mathbf{\Phi}_1^T\mathbf{\Phi}_1) & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\frac{\lambda}{\rho}\mathbf{I} + \mathbf{\Phi}_2^T\mathbf{\Phi}_2) & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & (\frac{\lambda}{\rho}\mathbf{I} + \mathbf{\Phi}_m^T\mathbf{\Phi}_m) & \mathbf{0} \\ -\frac{\rho}{m}\mathbf{\Phi}_1 & -\frac{\rho}{m}\mathbf{\Phi}_2 & \dots & -\frac{\rho}{m}\mathbf{\Phi}_m & (m+\rho)\mathbf{I} \end{bmatrix} \tag{5.41 revisited}$$

$$\mathbf{N} = \begin{bmatrix} (1-\frac{1}{m})\mathbf{\Phi}_1^T\mathbf{\Phi}_1 & -\frac{1}{m}\mathbf{\Phi}_1^T\mathbf{\Phi}_2 & \dots & -\frac{1}{m}\mathbf{\Phi}_1^T\mathbf{\Phi}_m & (1-\frac{m}{\rho})\mathbf{\Phi}_1^T \\ -\frac{1}{m}\mathbf{\Phi}_2^T\mathbf{\Phi}_1 & (1-\frac{1}{m})\mathbf{\Phi}_2^T\mathbf{\Phi}_2 & \dots & -\frac{1}{m}\mathbf{\Phi}_2^T\mathbf{\Phi}_2 & (1-\frac{m}{\rho})\mathbf{\Phi}_2^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\frac{1}{m}\mathbf{\Phi}_m^T\mathbf{\Phi}_1 & -\frac{1}{m}\mathbf{\Phi}_m^T\mathbf{\Phi}_2 & \dots & (1-\frac{1}{m})\mathbf{\Phi}_m^T\mathbf{\Phi}_m & (1-\frac{m}{\rho})\mathbf{\Phi}_m^T \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & m\mathbf{I} \end{bmatrix}. \tag{5.42 revisited}$$

If $\mathbf{M}$ *is nonsingular and the spectral radius* $\rho(\mathbf{M}^{-1}\mathbf{N}) < 1$ *then this method converges to the solution a system of linear equations*

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{5.43 revisited}$$

with

$$\begin{aligned} \mathbf{A} &= \mathbf{M} - \mathbf{N} \\ &= \begin{bmatrix} \left(\frac{\lambda}{\rho}\mathbf{I} + \frac{1}{m}\mathbf{\Phi}_1^T\mathbf{\Phi}_1\right) & \frac{1}{m}\mathbf{\Phi}_1^T\mathbf{\Phi}_2 & \dots & \frac{1}{m}\mathbf{\Phi}_1^T\mathbf{\Phi}_m & (\frac{m}{\rho}-1)\mathbf{\Phi}_1^T \\ \frac{1}{m}\mathbf{\Phi}_2^T\mathbf{\Phi}_1 & \left(\frac{\lambda}{\rho}\mathbf{I} + \frac{1}{m}\mathbf{\Phi}_2^T\mathbf{\Phi}_2\right) & \dots & \frac{1}{m}\mathbf{\Phi}_2^T\mathbf{\Phi}_p & (\frac{m}{\rho}-1)\mathbf{\Phi}_2^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{m}\mathbf{\Phi}_m^T\mathbf{\Phi}_1 & \frac{1}{m}\mathbf{\Phi}_p^T\mathbf{\Phi}_2 & \dots & \left(\frac{\lambda}{\rho}\mathbf{I} + \frac{1}{m}\mathbf{\Phi}_m^T\mathbf{\Phi}_m\right) & (\frac{m}{\rho}-1)\mathbf{\Phi}_m^T \\ -\frac{\rho}{m}\mathbf{\Phi}_1 & -\frac{\rho}{m}\mathbf{\Phi}_2 & \dots & -\frac{\rho}{m}\mathbf{\Phi}_m & \rho\mathbf{I} \end{bmatrix} \end{aligned}$$

Proof. If we solve (5.39) with respect to $\mathbf{u}^k$ and substitute the result into (5.40), we can remove the recurrent relationship and obtian simple update formula

$$\mathbf{u}^k = \frac{1}{\rho}(m \cdot \bar{\mathbf{f}}^k - \mathbf{y}). \tag{B.12}$$

Even more, we can completely eliminate the variable $\mathbf{u}^k$ in updates steps (5.38) and (5.39). To do this, we replace $\mathbf{u}^k$ in (5.38) and (5.39) according to (B.12). This leads to the two ADMM update steps

$$(\rho\mathbf{\Phi}_j^T\mathbf{\Phi}_j + \lambda\mathbf{I})\ \mathbf{w}_j^{k+1} = \rho\mathbf{\Phi}_j^T\left(\mathbf{\Phi}_j\mathbf{w}_j^k + \left(1 - \frac{m}{\rho}\right)\bar{\mathbf{f}}^k - \frac{1}{m}\sum_{i=1}^{m}\mathbf{\Phi}_i\mathbf{w}_i^k + \frac{1}{\rho}\mathbf{y}\right) \tag{B.13}$$

$$\overline{\mathbf{f}}^{k+1} = \frac{1}{m+\rho}\left(\frac{\rho}{m}\sum_{i=1}^{m}\mathbf{\Phi}_i\mathbf{w}_i^{k+1} + m\overline{\mathbf{f}}^{k}\right). \tag{B.14}$$

These equations signify that the linear relationship $(\mathbf{w}^{k+1}, \overline{\mathbf{f}}^{k+1})$ and $(\mathbf{w}^{k}, \overline{\mathbf{f}}^{k})$. Hence, (B.13) and (B.14) can be written in a matrix-vector form

$$\underbrace{\begin{bmatrix} (\frac{\lambda}{\rho}\mathbf{I}+\mathbf{\Phi}_1^T\mathbf{\Phi}_1) & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\frac{\lambda}{\rho}\mathbf{I}+\mathbf{\Phi}_2^T\mathbf{\Phi}_2) & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & (\frac{\lambda}{\rho}\mathbf{I}+\mathbf{\Phi}_m^T\mathbf{\Phi}_m) & \mathbf{0} \\ -\frac{\rho}{m}\mathbf{\Phi}_1 & -\frac{\rho}{m}\mathbf{\Phi}_2 & \dots & -\frac{\rho}{m}\mathbf{\Phi}_m & (m+\rho)\mathbf{I} \end{bmatrix}}_{=:\mathbf{M}} \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_m \\ \overline{\mathbf{f}} \end{bmatrix}^{k+1} =$$

$$\underbrace{\begin{bmatrix} (1-\frac{1}{m})\mathbf{\Phi}_1^T\mathbf{\Phi}_1 & -\frac{1}{m}\mathbf{\Phi}_1^T\mathbf{\Phi}_2 & \dots & -\frac{1}{m}\mathbf{\Phi}_1^T\mathbf{\Phi}_m & (1-\frac{m}{\rho})\mathbf{\Phi}_1^T \\ -\frac{1}{m}\mathbf{\Phi}_2^T\mathbf{\Phi}_1 & (1-\frac{1}{m})\mathbf{\Phi}_2^T\mathbf{\Phi}_2 & \dots & -\frac{1}{m}\mathbf{\Phi}_2^T\mathbf{\Phi}_2 & (1-\frac{m}{\rho})\mathbf{\Phi}_2^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\frac{1}{m}\mathbf{\Phi}_m^T\mathbf{\Phi}_1 & -\frac{1}{m}\mathbf{\Phi}_m^T\mathbf{\Phi}_2 & \dots & (1-\frac{1}{m})\mathbf{\Phi}_m^T\mathbf{\Phi}_m & (1-\frac{m}{\rho})\mathbf{\Phi}_m^T \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & m\mathbf{I} \end{bmatrix}}_{=:\mathbf{N}} \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_m \\ \overline{\mathbf{f}} \end{bmatrix}^{k} + \underbrace{\frac{1}{\rho}\begin{pmatrix} \mathbf{\Phi}_1^T\mathbf{y} \\ \mathbf{\Phi}_2^T\mathbf{y} \\ \vdots \\ \mathbf{\Phi}_m^T\mathbf{y} \\ \mathbf{0} \end{pmatrix}}_{=:\mathbf{b}}. \tag{B.15}$$

We recognise in a fixpoint iteration. Since the general form of fixpoint iterations is

$$\mathbf{M}\mathbf{x}^{k+1} = (\mathbf{M}-\mathbf{A})\mathbf{x}^{k} + \mathbf{b},$$

we can deduce the form the matrix $\mathbf{A}$. □

B.3. Corollary 5.1

The system of linear equations (5.43) *is equivalent to the augmented system of linear smoother equations*

$$\begin{bmatrix} \mathbf{I} & \mathbf{S}_1 & \cdots & \mathbf{S}_1 & (\frac{m}{\rho}-1)\mathbf{S}_1 \\ \mathbf{S}_2 & \mathbf{I} & \cdots & \mathbf{S}_2 & (\frac{m}{\rho}-1)\mathbf{S}_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{S}_m & \mathbf{S}_m & \cdots & \mathbf{S}_m & (\frac{m}{\rho}-1)\mathbf{S}_m \\ \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} & -\mathbf{I} \end{bmatrix} \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_m \\ \mathbf{f} \end{pmatrix} = \frac{m}{\rho}\begin{pmatrix} \mathbf{S}_1\mathbf{y} \\ \mathbf{S}_2\mathbf{y} \\ \vdots \\ \mathbf{S}_m\mathbf{y} \\ \mathbf{0} \end{pmatrix} \qquad \text{(5.44 revisited)}$$

with $\mathbf{f} = m\overline{\mathbf{f}}$, $\mathbf{f}_j = \mathbf{\Phi}_j\mathbf{x}_j$, *and* $\mathbf{S}_j = \mathbf{\Phi}_j(\mathbf{\Phi}_j^T\mathbf{\Phi}_j + \lambda\frac{m}{\rho}\mathbf{I})^{-1}\mathbf{\Phi}_j^T$ *for* $j = 1, \dots, m$.

Proof. The system of linear equation (5.43) has the form

$$\begin{bmatrix} \left(\frac{\lambda}{\rho}\mathbf{I}+\frac{1}{m}\mathbf{\Phi}_1^T\mathbf{\Phi}_1\right) & \frac{1}{m}\mathbf{\Phi}_1^T\mathbf{\Phi}_2 & \dots & \frac{1}{m}\mathbf{\Phi}_1^T\mathbf{\Phi}_m & (\frac{m}{\rho}-1)\mathbf{\Phi}_1^T \\ \frac{1}{m}\mathbf{\Phi}_2^T\mathbf{\Phi}_1 & \left(\frac{\lambda}{\rho}\mathbf{I}+\frac{1}{m}\mathbf{\Phi}_2^T\mathbf{\Phi}_2\right) & \dots & \frac{1}{m}\mathbf{\Phi}_2^T\mathbf{\Phi}_p & (\frac{m}{\rho}-1)\mathbf{\Phi}_2^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{m}\mathbf{\Phi}_m^T\mathbf{\Phi}_1 & \frac{1}{m}\mathbf{\Phi}_p^T\mathbf{\Phi}_2 & \dots & \left(\frac{\lambda}{\rho}\mathbf{I}+\frac{1}{m}\mathbf{\Phi}_m^T\mathbf{\Phi}_m\right) & (\frac{m}{\rho}-1)\mathbf{\Phi}_m^T \\ -\frac{\rho}{m}\mathbf{\Phi}_1 & -\frac{\rho}{m}\mathbf{\Phi}_2 & \dots & -\frac{\rho}{m}\mathbf{\Phi}_m & \rho\mathbf{I} \end{bmatrix} \cdot \begin{pmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_m \\ \bar{\mathbf{f}} \end{pmatrix} = \frac{1}{\rho} \begin{pmatrix} \mathbf{\Phi}_1^T\mathbf{y} \\ \mathbf{\Phi}_2^T\mathbf{y} \\ \vdots \\ \mathbf{\Phi}_m^T\mathbf{y} \\ \mathbf{0} \end{pmatrix}.$$

If we this system of equation by

$$m\cdot\begin{bmatrix} \mathbf{\Phi}_1\left(\frac{\lambda}{\rho}m\mathbf{I}+\mathbf{\Phi}_1^T\mathbf{\Phi}_1\right)^{-1} & & & \\ & \mathbf{\Phi}_2\left(\frac{\lambda}{\rho}m\mathbf{I}+\mathbf{\Phi}_2^T\mathbf{\Phi}_2\right)^{-1} & & \\ & & \ddots & \\ & & & \mathbf{\Phi}_m\left(\frac{\lambda}{\rho}m\mathbf{I}+\mathbf{\Phi}_m^T\mathbf{\Phi}_m\right)^{-1} \\ & & & \mathbf{I} \end{bmatrix}$$

from the right and substitute $\mathbf{\Phi}_j\left(\frac{\lambda}{\rho}m\mathbf{I}+\mathbf{\Phi}_j^T\mathbf{\Phi}_j\right)^{-1}\mathbf{\Phi}_j^T$ by $\mathbf{S}_j$, we obtain

$$\begin{bmatrix} \mathbf{I} & \mathbf{S}_1 & \cdots & \mathbf{S}_1 & \left(\frac{m}{\rho}-1\right)\mathbf{S}_1 \\ \mathbf{S}_2 & \mathbf{I} & \cdots & \mathbf{S}_2 & \left(\frac{m}{\rho}-1\right)\mathbf{S}_2 \\ & & \ddots & & \\ \mathbf{S}_m & \mathbf{S}_m & \cdots & \mathbf{I} & \left(\frac{m}{\rho}-1\right)\mathbf{S}_m \\ -\rho\mathbf{I} & -\rho\mathbf{I} & \cdots & -\rho\mathbf{I} & \rho\mathbf{I} \end{bmatrix} \cdot \begin{pmatrix} \mathbf{\Phi}_1\mathbf{w}_1 \\ \mathbf{\Phi}_2\mathbf{w}_2 \\ \vdots \\ \mathbf{\Phi}_m\mathbf{w}_m \\ m\bar{\mathbf{f}} \end{pmatrix} = \frac{m}{\rho} \begin{pmatrix} \mathbf{S}_1\mathbf{y} \\ \mathbf{S}_2\mathbf{y} \\ \vdots \\ \mathbf{S}_m\mathbf{y} \\ \mathbf{0} \end{pmatrix}.$$

If we replace $\mathbf{\Phi}_j\mathbf{w}_j$ by $\mathbf{f}_j$ and $m\bar{\mathbf{f}}$ by $\mathbf{f}$ we obtain (5.44). □

Glossary

ADMM Alternating Directions Method of Multipliers. 117, 118, 131–133, 136, 138–141, 143, 144, 147, 148, 159

BCD block coordinate descent algorithm. 76

BiCGStab Biconjugate Gradient Stabilised. 121–125, 128, 138, 139, 148

CAP composite absolute penalty. 67, 68

CD coordinate descent algorithm. 76

CG conjugate gradients. 78

DAG directed acyclic graph. 16, 67–70

FISTA fast iterative shrinkage-thresholding algorithm. 73, 74, 76

ISTA iterative shrinkage-thresholding algorithm. 73, 76

LARS least angle regression. 68, 75, 76

MSD Million Song Dataset. 84, 128

RKHS reproducing kernel Hilbert space. 32

References

Abramson, M. (1976). Restricted combinations and compositions. *Fibonacci Quart*, *14*(5), 439–452.

Achlioptas, D. (2001). Database-friendly random projections. In *Proceedings of the twentieth acm sigmod-sigact-sigart symposium on principles of database systems (PODS'01).* Association for Computing Machinery (ACM).

Adelman-McCarthy et al., J. K. (2007, October). The fifth data release of the sloan digital sky survey. *The Astrophysical Journal Supplement Series*, *172*(2), 634–644.

Ailon, N., & Chazelle, B. (2009, January). The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, *39*(1), 302–322.

Arnaldo, I., O'Reilly, U.-M., & Veeramachaneni, K. (2015). Building predictive models via feature synthesis. In *Proceedings of the 2015 annual conference on genetic and evolutionary computation (GECCO'15)* (pp. 983–990). New York, NY, USA: ACM.

Arrow, K. J., Hurwicz, L., & Uzawa, H. (1958). *Studies in linear and non-linear programming.* Stanford University Press.

Ashenfelter, O. (2010, March). Predicting the quality and prices of bordeaux wine. *Journal of Wine Economics*, *5*, 40–52.

Bach, F. (2013). Learning with submodular functions: A convex optimization perspective. *arXiv preprint arXiv:1111.6453*, *6*(2-3), 145-373.

Bach, F., Jenatton, R., Mairal, J., & Obozinski, G. (2011). Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, *4*(1), 1-106.

Bach, F., & Moulines, E. (2013). Non-strongly-convex smooth stochastic approximation with convergence rate O(1/n). In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Weinberger (Eds.), *Advances in neural information processing systems 26* (pp. 773–781). Curran Associates, Inc.

Balder, R. (1994). *Adaptive Verfahren für elliptische und parabolische Differential- gleichungen auf dünnen Gittern* (Doctoral dissertation). Technische Universität München.

Balder, R., & Zenger, C. (1996). The solution of multidimensional rea helmholtz equations on sparse grids. *SIAM Journal on Scientific Computing*, *17*(3), 631–646.

Batzner, K. (2015). *Subspace projection methods for high-dimensional supervised learning with sparse grids* (Bachelor's thesis). Institut für Informatik, Technische Universität München.

Beck, A., & Teboulle, M. (2009, January). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, *2*(1), 183–202.

Bellman, R. E. (1961). *Adaptive control processes - A guided tour*. Princeton, New Jersey, U.S.A.: Princeton University Press.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., & Lamere, P. (2011). The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.

Bertsekas, D., & Rheinboldt, W. (1982). *Constrained optimization and lagrange multiplier methods*. Academic Press.

Binev, P., Cohen, A., Dahmen, W., & DeVore, R. (2007). Universal algorithms for learning theory. part ii: Piecewise polynomial functions. *Constructive Approximation*, *26*(2), 127-152.

Binev, P., Cohen, A., Dahmen, W., DeVore, R., & Temlyakov, V. (2005). Universal algorithms for learning theory part i: Piecewise constant functions. *Journal of Machine Learning Research*, *6*, 1297–1321.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blumensath, T., & Davies, M. E. (2009). Stagewise weak gradient pursuits. *IEEE Transactions on Signal Processing*, *57*(11), 4333–4346.

Bohn, B., Garcke, J., & Griebel, M. (2016). A sparse grid based method for generative dimensionality reduction of high-dimensional data. *Journal of Computational Physics*, *309*, 1–17.

Bohn, B., & Griebel, M. (2013). An adaptive sparse grid approach for time series prediction. In J. Garcke & M. Griebel (Eds.), *Sparse grids and applications* (pp. 1–30). Berlin, Heidelberg: Springer Berlin Heidelberg.

Borwein, J. M., & Lewis, A. S. (2006). *Convex analysis and nonlinear optimization: Theory and examples*. Springer-Verlag New York.

Bottou, L. (1998). Online algorithms and stochastic approximations. In D. Saad (Ed.), *Online learning and neural networks* (pp. 9–42). Cambridge, UK: Cambridge University Press.

Bottou, L. (2012). Stochastic gradient tricks. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural networks, tricks of the trade, reloaded* (pp. 430–445). Springer.

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, *3*(1), 1-122.

Breiman, L. (1996, July). Stacked regressions. *Machine Learning*, *24*(1), 49–64.

Buja, A., Hastie, T., & Tibshirani, R. (1989, June). Linear Smoothers and Additive Models. *The Annals of Statistics*, *17*(2), 453–510.

Bungartz, H.-J. (1998). *Finite elements of higher order on sparse grids* (Habilitation thesis). Institut für Informatik, Technische Universität München.

Bungartz, H.-J., & Griebel, M. (2004, May). Sparse grids. *Acta Numerica*, *13*, 147-269.

Bungartz, H.-J., Pflüger, D., & Zimmer, S. (2008). Adaptive sparse grid techniques for data mining. In H. Bock, E. Kostina, H. Phu, & R. Rannacher (Eds.), *Modeling, simulation and optimization of complex processes* (p. 121-130). Springer Berlin Heidelberg.

Burges, C. J. C. (2009). Dimension reduction: A guided tour. *Foundations and Trends® in Machine Learning*, *2*(4), 275–365.

Buse, G. (2015). *Exploiting many-core architectures for dimensionally adaptive sparse grids* (Doctoral dissertation, Institut für Informatik, Technische Universität München, München).

Chen, S. S., Donoho, D. L., & Saunders, M. A. (1998). Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, *20*(1), 33–61.

Chu, E., Keshavarz, A., & Boyd, S. (2013, March). A distributed algorithm for fitting generalized additive models. *Optimization and Engineering*, *14*(2), 213–224.

Davenport, T., & Patil, D. (2012). Data scientist: the sexiest job of the 21st century. *Harvard business review*, *90*(10), 70–76.

Dick, J., Sloan, I. H., Wang, X., & Woźniakowski, H. (2004, October). Liberating the weights. *Journal of Complexity*, *20*(5), 593–623.

Donoho, D. L., Tsaig, Y., Drori, I., & Starck, J.-L. (2012, February). Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit. *IEEE Transactions on Information Theory*, *58*(2), 1094–1121.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004, April). Least angle regression. *Ann. Statist.*, *32*(2), 407–499.

Efron, B., & Stein, C. (1981). The jackknife estimate of variance. *The Annals of Statistics*, *9*(3), 586–596.

Feige, U. (1998, July). A threshold of ln n for approximating set cover. *Journal of the ACM*, *45*(4), 634–652.

Feuersänger, C. (2010). *Sparse Grid Methods for Higher Dimensional Approximation* (Doctoral dissertation, Institut für Numerische Simulation, Rheinische Friedrich-Wilhelms-Universität Bonn).

Feuersänger, C., & Griebel, M. (2009, July). Principal manifold learning by sparse grids. *Computing*, *85*(4), 267–299.

Fortin, M., & Glowinski, R. (1983). Augmented Lagrangian methods in quadratic programming. *Studies in Mathematics and its Applications*, *15*, 1–46.

Fradkin, D., & Madigan, D. (2003). Experiments with random projections for machine learning. In L. Getoor, T. E. Senator, P. M. Domingos, & C. Faloutsos (Eds.), *Proceedings of the ninth ACM SIGKDD interna-*

tional conference on knowledge discovery and data mining (pp. 517–522). ACM.

Friedman, J. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, *19*(1), 1–67.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*(1), 1–22.

Furnival, G. M., & Wilson, J., Robert W. (1974). Regressions by leaps and bounds. *Technometrics*, *16*(4), 499–511.

Gabay, D., & Mercier, B. (1976, January). A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, *2*(1), 17–40.

Garcke, J. (2004). *Maschinelles Lernen durch Funktionsrekonstruktion mit verallgemeinerten dünnen Gittern* (Doktorarbeit, Institut für Numerische Simulation, Universität Bonn).

Garcke, J., Griebel, M., & Thess, M. (2001). Data mining with sparse grids. *Computing*, *67*(3), 225–253.

Garcke, J., Hegland, M., & Nielsen, O. (2003). Parallelisation of sparse grids for large scale data analysis. *Forschung*, 699.

Gerstner, T., & Griebel, M. (2003, August). Dimension-Adaptive Tensor-Product Quadrature. *Computing*, *71*(1), 65–87.

Griebel, M. (2006). Sparse grids and related approximation schemes for higher dimensional problems. In L. Pardo, A. Pinkus, E. Suli, & M. Todd (Eds.), *Foundations of computational mathematics (FoCM05), santander* (pp. 106–161). Cambridge University Press.

Griebel, M., & Hullmann, A. (2014a). Dimensionality reduction of high-dimensional data with a nonlinear principal component aligned generative topographic mappin. *SIAM Journal on Scientific Computing*, *36*(3), A1027-A1047.

Griebel, M., & Hullmann, A. (2014b). A sparse grid based generative topographic mapping for the dimensionality reduction of high-dimensional data. *Modeling, Simulation and Optimization of Complex Processes - HPSC 2012*, 51–62.

Griebel, M., & Knapek, S. (2000). Optimized tensor-product approximation spaces. *Constructive Approximation*, *16*(4), 525–540.

Gu, C. (2013). *Smoothing spline ANOVA models*. Springer New York.

Guangliang, C., & Maggioni, M. (2010, March). Multiscale geometric wavelets for the analysis of point clouds. In *44th annual conference on information sciences and systems (CISS '10)* (pp. 1–6).

Halko, N., Martinsson, P. G., & Tropp, J. A. (2011, January). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, *53*(2), 217–288.

Hallatschek, K. (1992). Fouriertransformation auf dünnen gittern mit hierarchischen basen. *Numerische Mathematik*, *63*(1), 83-97.

Hastie, T., Tibshirani, R., & Friedman, J. (2011). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer.

He, B., Yang, H., & Wang, S. (2000). Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and Applications*, *106*(2), 337–356.

Hegland, M. (2003, April). Adaptive sparse grids. In K. Burrage & R. B. Sidje (Eds.), *ANZIAM Journal* (Vol. 44, pp. C335–C353).

Hegland, M., McIntosh, I., & Turlach, B. (1999, October). A parallel solver for generalised additive models. *Computational Statistics & Data Analysis*, *31*(4), 377–396.

Heinecke, A. (2014). *Boosting scientific computing applications through leveraging data parallel architectures* (Dissertation). Institut für Informatik, Technische Universität München, München.

Heinecke, A., Peherstorfer, B., Pflüger, D., & Song, Z. (2012, July). Sparse grid classifiers as base learners for adaboost. In *International Conference on High Performance Computing and Simulation (HPCS)* (p. 161-166).

Ho, C.-H., & Lin, C.-J. (2012, November). Large-scale linear support vector regression. *J. Mach. Learn. Res.*, *13*(1), 3323–3348.

Holtz, M. (2011). *Sparse Grid Quadrature in High Dimensions with Applications in Finance and Insurance* (Vol. 77). Berlin, Heidelberg: Springer Berlin Heidelberg.

Huang, J., & Zhang, T. (2010). The benefit of group sparsity. *Annals of Statistics*, *38*(4), 1978–2004.

Jacob, L., Obozinski, G., & Vert, J.-P. (2009). Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning* (pp. 433–440). New York, NY, USA: ACM.

Jakeman, J., & Roberts, S. (2011). Local and dimension adaptive sparse grid interpolation and quadrature. *arXiv preprint arXiv:1110.0010*, 1–21.

Johnson, W. B., & Lindenstrauss, J. (1984). Extensions of lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 189–206.

Kambadur, P., & Lozano, A. C. (2013). A parallel, block greedy method for sparse inverse covariance estimation for ultra-high dimensions. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics in Scottsdale, AZ, USA* (pp. 351–359).

Khakhutskyy, V., & Hegland, M. (2014a). Parallel fitting of additive models for regression. *KI 2014: Advances in Artificial Intelligence*, 243–254.

Khakhutskyy, V., & Hegland, M. (2014b, September). Parallel learning algorithm for large-scale regression with additive models. In *ECML PKDD 2014, PhD Session* (pp. 101–110).

Khakhutskyy, V., & Hegland, M. (2016). Spatially-dimension-adaptive sparse grids for online learning. In J. Garcke & D. Pflüger (Eds.), *Sparse Grids and Applications - Stuttgart 2014* (Vol. 109, pp. 133–162). Cham: Springer International Publishing.

Khakhutskyy, V., & Pflüger, D. (2014). Alternating direction method of multipliers for hierarchical basis approximators. In J. Garcke & D. Pflüger (Eds.), *Sparse Grids and Applications - Munich 2012* (Vol. 97, pp. 221–238). Springer International Publishing.

Khakhutskyy, V., Pflüger, D., & Hegland, M. (2014, March). Scalability and fault tolerance of the alternating direction method of multipliers for sparse grids. In M. Bader, H.-J. Bungartz, A. Bode, M. Gerndt, & G. Joubert (Eds.), *Parallel Computing: Accelerating Computational Science and Engineering (CSE)* (Vol. 25, pp. 603–612). Amsterdam: IOS Press.

Kimeldorf, G., & Wahba, G. (1971). Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, *33*, 82–95.

Kpotufe, S. (2009). Escaping the curse of dimensionality with a tree-based regressor. In *Conference on computational learning theory.*

Kpotufe, S., & Dasgupta, S. (2012). A tree-based regressor that adapts to intrinsic dimension. *Journal of Computer and System Sciences*, *78*(5), 1496-1515.

Kpotufe, S., & Garg, V. (2013). Adaptivity to local smoothness and dimension in kernel regression. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Weinberger (Eds.), *Advances in neural information processing systems 26* (pp. 3075–3083). Curran Associates, Inc.

Kushmerick, N. (1999). Learning to remove internet advertisements. In *Proceedings of the third annual conference on autonomous agents* (pp. 175–181). New York, NY, USA: ACM.

Lee, J. A., & Verleysen, M. (2007). *Nonlinear dimensionality reduction*. Springer New York.

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., & Glance, N. (2007). Cost-effective Outbreak Detection in Networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 420–429). San Jose, California, USA: ACM.

Lim, M., & Hastie, T. (2015). Learning interactions via hierarchical group-lasso regularization. *Journal of Computational and Graphical Statistics*, *24*(3), 627-654.

Liu, J., Fujimaki, R., & Ye, J. (2014). Forward-backward greedy algorithms for general convex smooth functions over a cardinality constraint. In *31st International Conference on Machine Learning, ICML 2014* (Vol. 1, p. 765-773).

Mairal, J., & Yu, B. (2013). Supervised feature selection in graphs with path coding penalties and network flows. *Journal of Machine Learning Research*, *14*, 2449-2485.

Martinez-Cantin, R. (2014). Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits. *Journal of Machine Learning Research*, *15*, 3915-3919.

Merkel, A. (2015, November). *Rede von Bundeskanzlerin Merkel zum Tag der Deutschen Industrie 2015 des Bundesverbands der Deutschen Industrie e. V. am 3. November 2015.* Talk.

Minoux, M. (1978). Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques, Lecture Notes in Control and Information Sciences*, *7*, 234–243.

M. R. Osborne, B. A. T., Brett Presnell. (2000). On the lasso and its dual. *Journal of Computational and Graphical Statistics*, *9*(2), 319-337.

Natarajan, B. K. (1995). Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, *24*(2), 227-234.

Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions-I. *Mathematical Programming*, *14*, 265–294.

Nesterov, Y. (2004). *Introductory lectures on convex optimization* (Vol. 87). Springer US.

Nesterov, Y. (2007, June). Primal-dual subgradient methods for convex problems. *Mathematical Programming*, *120*(1), 221–259.

Nielsen, O., Shen, Z., & Hegland, M. (2004). Multidimensional smoothing using hyperbolic interpolatory wavelets. *Electronic Transactions on Numerical Analysis*, *17*, 168–180.

Nielsen, O. M. (2000). High dimensional wavelet smoothing. *ANZIAM Journal*, *42*, 1034–1057.

Nieplocha, J., Harrison, R., & Littlefield, R. (1996). Global arrays: A nonuniform memory access programming model for high-performance computers. *The Journal of Supercomputing*, *10*(2), 1–17.

Nocedal, J., & Wright, S. J. (2006). *Numerical optimization* (2nd ed.). Springer New York.

Peherstorfer, B. (2013). *Model order reduction of parametrized systems with sparse grid learning techniques* (Doctoral dissertation). Department of Informatics, Technische Universität München.

Peherstorfer, B., Pflüger, D., & Bungartz, H.-J. (2011). A sparse-grid-based out-of-sample extension for dimensionality reduction and clustering with laplacian eigenmaps. In D. Wang & M. Reynolds (Eds.), *AI 2011: Advances in Artificial Intelligence* (Vol. 7106, p. 112-121). Springer Berlin Heidelberg.

Pfander, D., Heinecke, A., & Pflüger, D. (2016). Sparse grids and applications - stuttgart 2014. In J. Garcke & D. Pflüger (Eds.), (pp. 221–246). Cham: Springer International Publishing.

Pflüger, D. (2010). *Spatially adaptive sparse grids for high-dimensional problems* (Doctoral dissertation, Institut für Informatik, Technische Universität München, München).

Pflüger, D. (2012, October). Spatially adaptive refinement. In J. Garcke & M. Griebel (Eds.), *Sparse grids and applications* (pp. 243–262). Berlin Heidelberg: Springer.

Polyak, B. T., & Juditsky, a. B. (1992). Acceleration of Stochastic Approximation by Averaging. *SIAM Journal on Control and Optimization*, *30*(4), 838–855.

Rabitz, H., & Alis, O. (1999). General foundations of high-dimensional model representations. *Journal of Mathematical Chemistry*, *25*.

Rooney, N., Patterson, D., Anand, S., & Tsymbal, A. (2004). Random Subspacing for Regression Ensembles. *FLAIRS Conference*.

Rosenblatt, M. (1952). Remarks on a multivariate transformation. *The Annals of Mathematical Statistics*, *23*(3), 470–472.

Roth, V., & Fischer, B. (2008). The group-lasso for generalized linear models: Uniqueness of solutions and efficient algorithms. In *Proceedings of the 25th international conference on machine learning* (pp. 848–855). New York, NY, USA: ACM.

Schaul, T., Zhang, S., & LeCun, Y. (2013). No More Pesky Learning Rates. *Journal of Machine Learning Research*, *28*, 343–351.

Scholkopf, B., & Smola, A. J. (2001). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. Cambridge, MA, USA: MIT Press.

Scott, C., & Nowak, R. (2005). On the adaptive properties of decision trees. In *Advances in neural information processing systems.*

Sloan, I. H., Wang, X., & Woźniakowski, H. (2004, February). Finite-order weights imply tractability of multivariate integration. *Journal of Complexity*, *20*(1), 46–74.

Smolyak, S. A. (1963). Quadrature and interpolation formulas for tensor products of certain classes of functions. *Dokl. Akad. Nauk SSSR.*

Stone, C. J. (1982, December). Optimal global rates of convergence for nonparametric regression. *The Annals of Statistics*, *10*(4), 1040–1053.

Strauß, K. (2016). *Convergence of the asynchronous and partially-asynchronous admm methods for sparse grid model splitting* (Bachelor's thesis). Fakultät für Mathematik, Technische Universität München.

Szlam, A., Maggioni, M., Coifman, R., & Bremer Jr., J. (2005). Diffusion-driven multiscale analysis on manifolds and graphs: Top-down and bottom-up constructions. In *Proceedings of SPIE - The International Society for Optical Engineering* (Vol. 5914, p. 1-11).

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, *58*(1), pp. 267-288.

Tipping, M. E. (2001, September). Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, *1*, 211–244.

Tipping, M. E., & Faul, A. C. (2003). Fast marginal likelihood maximisation for sparse bayesian models. In *Proceedings of the ninth international workshop on artificial intelligence* (pp. 3–6).

Tropp, J. A., & Gilbert, A. C. (2007, December). Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on*

Information Theory, *53*(12), 4655–4666.

Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, *109*(3), 475–494.

Turlach, B., Venables, W., & Wright, S. (2005). Simultaneous variable selection. *Technometrics*, *47*(3), 349-363.

Uphoff, C. (2015). *Parallel fitting of additive models* (Master's thesis). Fakultät für Elektrotechnik und Informationstechnik, Technische Universität München.

van der Vorst, H. (1992). Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, *13*(2), 631-644.

Wahba, G. (1990, January). Spline models for observational data. In *Regional Conference Series in Applied Mathematics of CBMS-NSF* (Vol. 59, pp. 127–133). 3600 Market Street, 6th Floor Philadelphia, PA 19104-2688: SIAM.

Wahba, G., Wang, Y., Gu, C., Klein, R., & Klein, B. (1995, Dec). Smoothing spline ANOVA for exponential families, with application to the wisconsin epidemiological study of diabetic retinopathy: the 1994 Neyman memorial lecture. *The Annals of Statistics*, *23*(6), 1865–1895.

Wang, H., Vieira, J., Ferreira, P., Jesus, B., & Duarte, I. (2009). Batch algorithms of matching pursuit and orthogonal matching pursuit with applications to compressed sensing. In *IEEE International Conference on Information and Automation (ICIA 2009)* (pp. 824–829).

Wei, K., Bilmes, J., Edu, R. U. W., & Edu, B. U. W. (2014). Fast Multi-Stage Submodular Maximization. In *International conference on machine learning*. Beijing, China.

Weinberger, K., Dasgupta, A., Langford, J., Smola, A., & Attenberg, J. (2009). Feature hashing for large scale multitask learning. In *Proceedings of the 26th annual international conference on machine learning* (pp. 1113–1120). New York, NY, USA: ACM.

Wipf, D. P., & Nagarajan, S. S. (2008). A new view of automatic relevance determination. In J. Platt, D. Koller, Y. Singer, & S. Roweis (Eds.), *Advances in neural information processing systems 20* (pp. 1625–1632). Curran Associates, Inc.

Wolpert, D. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241–259.

Wolpert, D. H. (1996, October). The lack of a priori distinctions between learning algorithms. *Neural Computing*, *8*(7), 1341–1390.

Yang, L., & Brent, R. P. (2002, October). The improved bicgstab method for large and sparse unsymmetric linear systems on parallel distributed memory architectures. In *Proceedings of the fifth International Conference on Algorithms and Architectures for Parallel Processing, 2002* (p. 324-328).

Yang, Y., & Dunson, D. B. (2016, April). Bayesian manifold regression. *The Annuals of Statistics*, *44*(2), 876–905.

Yuan, G.-X., Chang, K., Hsieh, C.-J., & Lin, C.-J. (2010). A comparison of optimization methods and software for large-scale l1-regularized linear classification. *Journal of Machine Learning Research*, *11*, 3183-3234.

Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *68*(1), 49–67.

Zenger, C. (1990). Sparse grids. In W. Hackbusch (Ed.), *Parallel algorithms for partial differential equations* (pp. 241–251). Vieweg.

Zhang, H. H., Wahba, G., Lin, Y., Voelker, M., Ferris, M., Klein, R., & Klein, B. (2004, September). Variable selection and model building via likelihood basis pursuit. *Journal of the American Statistical Association*, *99*(467), 659–672.

Zhang, T. (2009). Adaptive forward-backward greedy algorithm for sparse learning with linear models. In *Advances in neural information processing systems 21* (p. 1921-1928).

Zhang, T. (2011). Adaptive forward-backward greedy algorithm for learning sparse representations. *IEEE Transactions on Information Theory*, *57*(7), 4689-4708.

Zhao, P., Rocha, G., & Yu, B. (2009). The composite absolute penalties family for grouped and hierarchical variable selection. *Annals of Statistics*, *37*(6 A), 3468-3497.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, *67*(2), 301-320.