Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 236

Sparse Grid Datamining with Huge Datasets

Max Franke

Course of Study:	Computer Science
Examiner:	Jun. Prof. Dr. rer. nat. Dirk Pflüger
Supervisor:	Dipl. Inf. David Pfander
Commenced:	2015-05-25
Completed:	2015-11-25
CR-Classification:	H.2.8

Acknowledgements

This thesis has benefited greatly from the support of various people, whom I'd like to thank sincerely for everything they have done for me.

First of all, I'd like to thank the Institute for Parallel and Distributed Systems (IPVS) at the University of Stuttgart in general, for generously providing me with a very powerful server to do the computations for my thesis on. The approximately 15 years of user and system time I got to consume are highly appreciated, and, hopefully, put to a good use.

I'd also like to thank, in particular, the department for Simulation of Large Systems (SGS) at the IPVS for their general friendly atmosphere. I greatly enjoyed bike tours and Christmas parties with the department, and regret not having been able to partake in the Back to the Future movies night. The occasional free coffee and friendly words in the office of Stefan Zimmer are also highly appreciated.

I'd like to offer special thanks to my supervisor, David Pfander, for his extensive and accommodating assistance during my thesis. He left no questions unanswered, could always offer me helpful pointers when confronted with an error message, and generally took his position as my supervisor very serious.

I'd like to thank Dr.-Ing. Markus Schubert and Prof. Dr. rer. nat. habil. Jürgen Werner from the Institute for Photovoltaics for providing the photovoltaic data and, on request, agreeing to a meeting and suggesting scenarios for data mining and a portion of knowledge about photovoltaic cells. Furthermore, my thanks go to Thomas Müller, Michael Grotz and Micha Eisele from the Institute for Water and Environment Modeling for providing a second weather dataset on short notice. The data were very helpful.

Last, but not least, I'd like to thank my friends and family for their support during the thesis. Thanks go to Gregor Daiß, who wrote his thesis simultaneously with me, for sharing his knowledge of C++ and data mining algorithms, and for the good times we shared in our free time. Special thanks go to my girlfriend Fiona for always being there, especially when I despaired of segmentation faults or the amount of work simply overwhelmed me, and for accepting the time pressure I was under.

Abstract

Due to the inflated costs of disk space and the prevalence of sensor equipment everywhere, the scientific world is flooded by huge amounts of data. The intention being to somehow benefit from that data, data mining algorithms are used to evaluate those data. As conventional data mining methods scale at least linear with problem size and exponentially with input problem dimension, this poses a great problem as to the computing power required to mine these data. For the testing of data mining algorithms, very few real world reference datasets exist. Using an already in-place toolkit for data mining on sparse grids, the goal of this thesis is to generate one or more real world reference datasets for data mining purposes. For this purpose, multiple weather and photovoltaic datasets were used. It was possible to learn 6-dimensional datasets with 1.2 million data points and obtain a very good prediction of photovoltaic power. Thus, a dataset was obtained to test regression on. For classification, a 9-dimensional dataset with 200 000 data points was generated, which however didn't have overly good results, with a 41% hit rate over 4 classes. Here, further processing of the data will be necessary.

Contents

List of Figures

List of Tables

Listings

List of Algorithms

1 Introduction

The ultimate goal of this work is to produce one or more huge real world datasets, which then can be used to test the efficiency of data mining algorithms. There is a lack of real world datasets to work on, and in particular there is a lack of real world datasets of a certain size. Most real world datasets have a very limited size. For instance, the well-known **Old Faithful**[Was13] dataset, which consists of a data column for the time between eruptions for the geyser with the same name and a data row for the duration of the eruption, only consists of 272 data points. Moreover, it is only two-dimensional. This makes it easy to apply data mining algorithms to it. While there exist larger and higher-dimensional collected data from Old Faithful, applying data mining methods to it still remains a very small problem.

To test new data mining methods, bigger datasets are often needed to test the stability and scalability of the method. To achieve thorough testing, in many cases, synthetic datasets are generated. Synthetic datasets have a number of advantages: They are easily scalable, both in dimension and size. They are easily conveyable, as they are generated by a program with specific parameters, so in practice, one can simply send the source code and parameters. Synthetic datasets are typically generated by evaluating a function at random points throughout the attribute space, and then adding white noise to the evaluated function[TSN99][DR11].

However, data mining methods are designed to be used on real world data. Learning synthetic data has no other use than to test whether the method is working as wanted. Sooner or later, every data mining algorithm has to be used on real world data, for example to detect traffic patterns, predict weather, or group customers by their shopping habits. These applications are all based on real world data, and real world data has some properties that are very hard to replicate in synthetic datasets, and which make data mining way harder.

Real world datasets nearly always contain some sort of noise or measurement inaccuracy which is very difficult to replicate by simply applying random white noise by normal distribution or other techniques. This is because most measured values in some way dependent on some variable which might not be part of the dataset. For instance, the wind speed at a certain location might be dependent on the solar radiation on a large forest a few kilometers away. However, for a data mining scenario to predict the wind speed at that location, the solar radiation above the forest is most likely not measured, and thus the wind speed will seem to have a random component.

Moreover, in synthetic datasets, the data points are usually evenly distributed throughout the attribute space. This makes it a lot easier to do regression on the data, as there are no

undefined areas to consider. In real world datasets, a large part of the attribute space is often sparsely populated with data points, if there are points there at all. This is not a problem for clustering (see section 2.5 on page 33) or density estimation based classification (see section 2.4.2 on page 30), but it is for regression (see section 2.3 on page 26).

Another point is that synthetic datasets can be generated tailor-made for the algorithm or method they are supposed to test. This leads to the possibility that datasets are generated in a way that favors the algorithm, and thus testing it becomes a charade. Using real world datasets to test algorithms assures that the algorithm is working correctly under harder conditions as well.

Finding real world datasets of a higher dimensionality and especially a greater dataset size is as of now very difficult. However, for many methods, testing requires large datasets to test algorithmic attributes like scalability and stability. The goal of this work was to experiment with a very large database of collected weather and power data from a photovoltaic experimental plant in Stuttgart and to generate a big dataset from those data which could be used to test data mining algorithms.

In chapter 2, I will define the necessary symbols and notations for my thesis. I will define the metrics necessary to assess the accuracy of data mining methods. I will then explain in detail what interpolation of a function on a finite function space means. The term of the full grid will be explained and then used to explain the hierarchical grid and, last, the sparse grid.

The data mining methods used in this thesis will be listed. Their functionality will be explained, together with the mathematical definition of what they do.

In chapter 3, the tools and data used for this work will be listed. The spatially adaptive sparse grid toolkit SG++will be introduced. Other tools used to generate this work will be shortly listed. I will also explain the source of the used data and how they were preprocessed.

In chapter 4, I will explain the generation of the datasets used for experiments in the thesis. First, I will describe how the data were preprocessed and merged into one database. Then, I will explain the concept of delay embedding, which was used to generate additional and useful attributes for the datasets. Last, I will briefly show the pipeline from the database to the dataset.

In chapter 5, I will list the scenarios I based my datasets on and their reasoning. I will also list all experiments I did with different datasets and discuss their outcome. Lastly, I will share some findings about dataset sizes.

In chapter 6, I will propose some starting points on which further work can be done based on this thesis. This is only a Bachelor's thesis, and it is easily possible to start additional research on the status quo of the data I collected and maintained. With more knowledge into photovoltaics or weather theory, far better scenarios could be designed. Also, with some specialization into data engineering, the data could be refined to only the most relevant data, possibly leading to better results.

2 Data Mining on Sparse Grids

2.1 Notations

In this section, the notations and definitions which are used throughout the thesis are listed.

2.1.1 General notations

Let a measured value with be called an **attribute**, with a specific domain. An attribute can for example be the air temperature or relative humidity.

For a set of attributes $A_i, 1 \leq i \leq d$ with $A_i = \left\{a_j \in \mathbb{R}\right\}_{j=1}^{N \in \urcorner}$ let the general attribute space Ω be the space of value tuples of those attributes

$$\Omega := \mathbb{R}^d \tag{2.1}$$

In order to be able to work with the attribute space in a discrete environment, e.g. in computers, we scale the attribute space down to the $[0,1]^d$ hypercube and define the normalized attribute space

$$\Omega_n := [0,1]^d \tag{2.2}$$

for an arbitrary dimension d.

For a grid of dimension d, let the **grid level** be denoted as l such that the number of grid points in each dimension is $2^l - 1$. Let the per-dimension index i of a grid point be its sequential number. Not including the boundary grid points, the indexation starts with 1 and ends with $2^l - 1$. In Fig. 2.1, a full grid of level 2 is shown, with the boundary grid points in red. For a grid of level l, let the resolution h_l of the grid be denoted as

$$h_l := 2^{-l} \tag{2.3}$$

Let the Euclidean norm or 2-norm be denoted by

$$\|\vec{v}\|_2 := \sqrt{\sum_{j=1}^{n} v_j^2} \tag{2.4}$$

as the length of the vector $\vec{v}$, and let the discrete ℓ_2 norm be the 2-norm squared.

Let the ℓ_1 norm or Manhattan norm of $\vec{v}$ be defined as

$$|\vec{v}|_1 := \sum_{j=1}^{n} \left| v_j \right| \tag{2.5}$$

Let the maximum norm of $\vec{v}$ be

$$|\vec{v}|_\infty := \max_{1 \leq j \leq d} \left| v_j \right| \tag{2.6}$$

Let the i-th component of a vector $\vec{v} \in \mathbb{R}^k$ be denoted as

$$v_i \in \mathbb{R} := \langle \vec{v}, \vec{e}_i \rangle \tag{2.7}$$

For two vectors of same size $\vec{u}, \vec{v}$, let

$$\vec{u} < \vec{v} \Leftrightarrow u_i < v_i \forall i \tag{2.8}$$

and analog for $\leq, \geq, >$.

Let $\vec{1}$ denote the vector $\vec{v}$ of matching length with $v_i = 1 \ \forall i$. Let $\vec{0}$ denote the vector $\vec{v}$ of matching length with $v_i = 1 \ \forall i$.

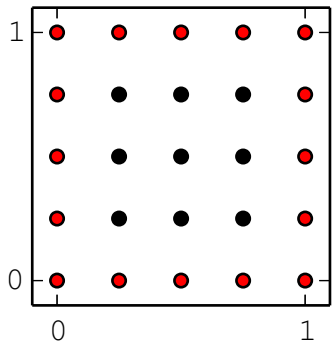


Figure 2.1: Full grid of grid level 2 in the space Ω_n of dimension 2. For our purposes, the boundary grid points in red are not included, and the index in each dimension is limited by $1 \leq i < 2^l$.

2.1.2 Time series addressing notations

For working on time series data, let the following notations be defined for use throughout this thesis. For a time-dependent measured value A, let $A^{(t)}$ be used to denote the value of A at time t. To address values of A in the past or future of time t, let Δt be the time difference size which will then be implicitly used from that point on until a new time difference is defined. With a defined Δt, let

$$\forall k \in \mathbb{Z}: \quad A^{(t+k)} := A\left(t + k\Delta t\right) \tag{2.9}$$

For differences between values across a certain time, let

$$\Delta A^{(t)} := A^{(t)} - A^{(t-1)} \tag{2.10}$$

and, for special uses where more than one time step difference has to be considered:

$$\Delta_k A^{(t)} := A^{(t)} - A^{(t-k)} \tag{2.11}$$

2.1.3 Statistics

Let the mean squared error (MSE) over a prediction $\vec{y}$ for expected values $\vec{x}$ be defined as

$$\begin{aligned} \mathrm{MSE}\left(\vec{x}, \vec{y}\right) &:= \frac{1}{N} \sum_{i=1}^{N} \left(\vec{y}_i - \vec{x}_i\right)^2 \\ &= \frac{1}{N} \left\| \vec{x} - \vec{y} \right\|_2^2 \end{aligned} \tag{2.12}$$

where $\dim \vec{x}, \dim \vec{y} \in \mathbb{R}^N$.

The mean squared error is a metric for errors which weighs the error by its severity. That is, bigger absolute errors have a greater impact on the mean squared error. However, it is a metric which is not easily comparable with other mean squared errors, because the size of the mean squared error is dependent on the range of the data it was applied on.

In fact, the MSE will scale quadratically with the data range if the relative error of each data point stays the same. So, for a comparison where the relative error is more relevant than the absolute error, the mean squared error is not suitable.

Let the **normalized mean squared error** NMSE as the quotient of the mean squared error and the discrete ℓ_2 norm of the reference data:

$$\begin{aligned} \mathrm{NMSE}\left(\vec{x}, \vec{y}\right) &:= \frac{\mathrm{MSE}\left(\vec{x}, \vec{y}\right)}{\mathrm{MSE}\left(\vec{x}, \vec{0}\right)} \\ &= \frac{\left\| \vec{x} - \vec{y} \right\|_2^2}{\left\| \vec{x} \right\|_2^2} \end{aligned} \tag{2.13}$$

Let the **hit rate** be defined for a test dataset of discrete values $\vec{d}$ and a prediction of those values $\vec{e}$ as the relative amount of correct predictions:

$$\text{hit}\left(\vec{d},\vec{e}\right) := \frac{\left|\{i \ : \ d_i = e_i \ , \ d_i \in D\}\right|}{N} \tag{2.14}$$

with $\vec{d},\vec{e} \in S^N, |S| < \infty$. The hit rate will be annotated in percentage points throughout this thesis.

2.2 Sparse Grids

In order to be able to work with continuous functions efficiently with numerical methods, those functions have to be discretised. This can be done by trying to approximate the function as precisely as possible using an underlying function space. One method of doing this is populating the attribute space with a grid of the same dimension, with a distance $h_l = 2^{-l}$ for a given grid level l between the grid points in each direction. The function value is then evaluated for each grid point, so that the difference from the resulting discrete function to the original function is as small as possible. Prediction of function values in between grid points is then done via interpolation.

2.2.1 Motivation for sparse grids

As the number of attributes, that is, the dimension of the attribute space, gets larger, the number of grid points generated grows exponentially with constant grid level l. This way, when working on higher-dimensional problems, very soon the needed resources for the approximation of the function become unsustainable. This problem is known as the **curse of dimensionality**[Bel03] and can be tackled by using a subspace of only the grid points that have the most influence on the representation of the approximated function. Such a subspace of grid points is called a sparse grid.

A sparse grid of dimension d and grid level n has

$$\mathcal{O}\left(2^n n^{d-1}\right)$$

grid points according to Garcke[Gar13], Bungartz and Griebel[BG04], while a full grid has

$$\mathcal{O}\left(2^{n\cdot d}\right)$$

grid points in total. For dimension $d = 3$, this is already a very clear difference, as seen in Tab. 2.1. So basically, we trade off some precision[BG04][Pfl10, 12] for a much less complex problem, as every basis function – that means every grid point – is represented by one row

in a matrix. As many algorithms on matrices have higher-than-linear scaling, this can get expensive both in time and storage very fast. Using sparse grids makes our calculation matrices way smaller, and thus, makes calculating the result much faster.

Table 2.1: Number of grid points for a full and a sparse grid of level 3, according to Bungartz[Bun15].

Grid level n	Full grid points	Sparse grid points
1	1	1
2	27	7
3	343	31
4	3 375	111
⋮	⋮	⋮
10	1 070 590 167	47 103

I will now explain the interpolation of a function on a full grid, and then explain how a hierarchical grid and later a sparse grid are generated from the full grid's subspaces.

2.2.2 Interpolation on a full grid

Before explaining sparse grids, it is important to understand how the approximation of a function using a finite function space works in general, that is, on a so-called full grid. It has a level l, a dimension d, and, subsequently, $\mathcal{O}\left(2^{l \cdot d}\right)$ grid points, cf. page 13. On a full grid, we use basis functions $\varphi_i(x)$ from the function space F_l, which is defined in Eq. 2.21, for the interpolation. The function is then evaluated as the sum over all these weighted basis functions. For a function $f(x)$ an interpolation $u(x)$ is done by calculating factors α_i for the respective basis function φ_i.

$$f(\vec{x}) \approx u(\vec{x}) := \sum_i \alpha_i \varphi_i(\vec{x}) \tag{2.15}$$

in such a way that, for a set of data points $\vec{x}_i \in \mathcal{D}$ and their respective function values $y_i = f(\vec{x}_i)$:

$$u(\vec{x}_i) = y_i \qquad \forall \vec{x}_i \in \mathcal{D} \tag{2.16}$$

The most simple basis function is the linear hat function:

$$\varphi(x) = \begin{cases} 0 & x < 0 \\ 2x & 0 \le x < \frac{1}{2} \\ -2x + 2 & \frac{1}{2} \le x < 1 \\ 0 & x \ge 1 \end{cases} \tag{2.17}$$

or

$$\varphi(x) = \max(1 - |x|, 0) \tag{2.18}$$

When more than one hat spanning the whole feature space is used, which is usually the case, this hat is dilated by the factor of h_l and translated by $i \cdot h_l$ for it's index i on the level, with $0 < i \leq 2^l$. This way, the following basis function $\varphi_{l,i}(x)$ for level l and index i are generated.

$$\varphi_{l,i}(x) = \varphi(2^l x - i) \tag{2.19}$$

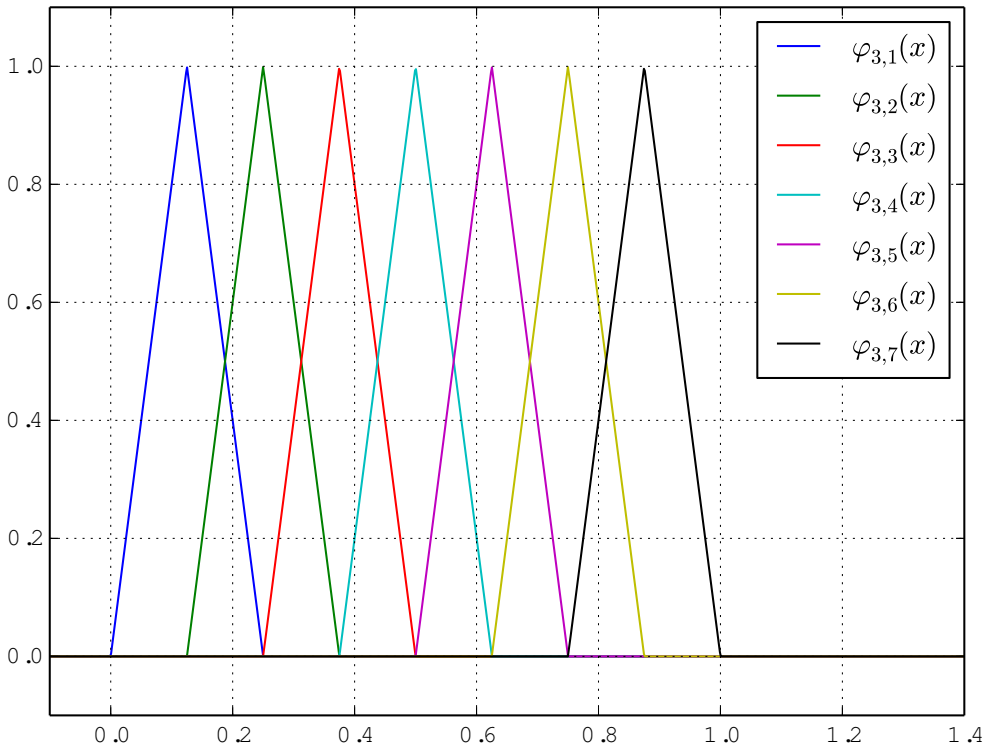


Figure 2.2: Nodal basis functions (hat) $\varphi_{l,i}$ of dimension 1 and level 3.

In Fig. 2.2, all basis functions for level 3 are shown. The basis functions overlap and, subsequently, their sum is simply the piecewise linear interpolation in between the peaks of the hats.

To use the discretization approach in higher dimension, the basis function has to be of higher dimension as well. This is done by simply calculating the tensor product of the basis function

in each direction. The level and index become vectors this way, with one level and index value for each dimension.

$$\varphi_{\vec{l},\vec{i}}(x) := \prod_{j=1}^{d} \varphi_{l_j,i_j}(x_j) \tag{2.20}$$

with $\vec{l},\vec{i} \in \mathbb{N}_0^d$ and $\forall j \leq d, j \in \mathbb{N} : 1 \leq i_j < 2^{l_j}$. The function space $F_{\vec{l}}^d$ of a maximum grid level $n \in \mathbb{R}$ and dimension d is then defined as

$$F_n^d := \operatorname{span}\left\{\varphi_{n\cdot\vec{1},\vec{i}} \;:\; 1 \leq i_j < 2^n \quad \forall j \in [1,d] \subsetneq \mathbb{N}\right\} \tag{2.21}$$

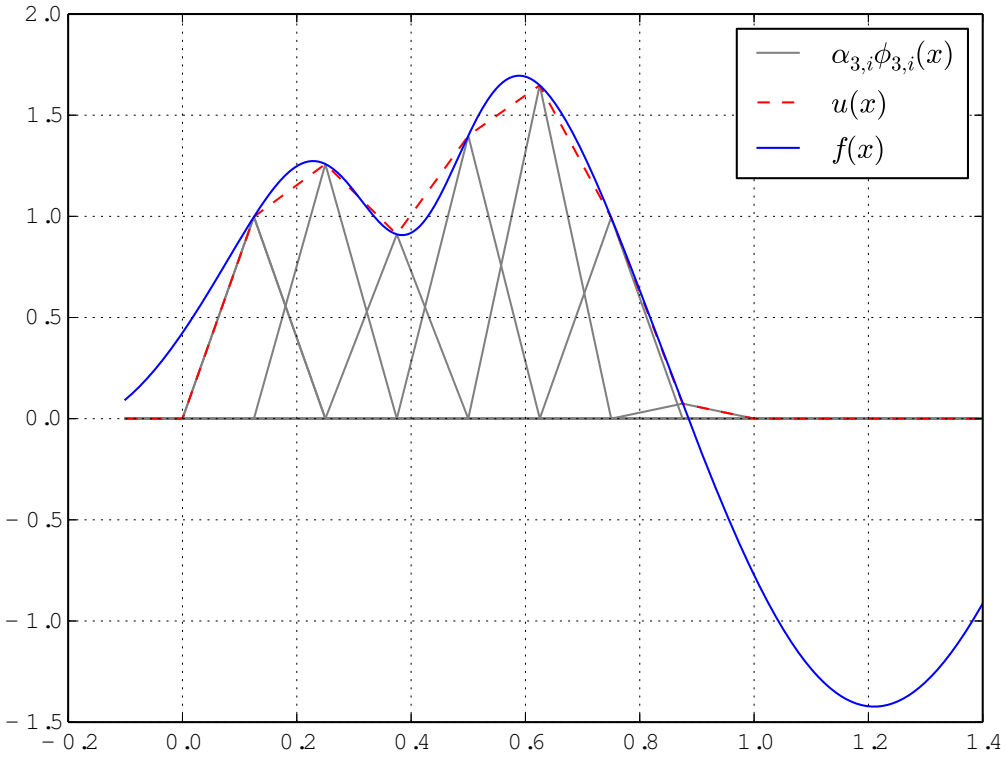


Figure 2.3: Nodal basis functions (hat) $\varphi_{l,i}$ of a full grid of dimension 1 and level 3 interpolating a function f (see Eq. 2.22) by their sum $u = \sum_{i=1}^{7} \alpha_{3,i}\varphi_{3,i}$.

In Fig. 2.2, the basis functions $\varphi_{3,i}, 1 \leq i \leq 7$ are shown. In Fig. 2.3, we approximate a function

$$f(x) = \sin(4x) + 0.6\sin(5x+5) - 0.7x + 0.01x^2 - \frac{0.3}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{2.22}$$

for Gaussian distribution factors $\sigma = 0.1, \mu = 0.4$ on a nodal basis of level $l = 3$. As can be seen from Fig. 2.2, the basis functions overlap one by one. Thus, their sum is a piecewise d-linear

interpolation over the function values in the grid points, starting in $(0,0)$, and ending in $(1,0)$.

The function f is approximated quite well by u, except for the boundaries of Ω_n. That cannot be helped with the current definition of the nodal basis. For applications where a better boundary treatment[Pfl10, 12–16] is of importance, the definition of the nodal basis can be modified so that the value at the edge of the attribute space does not have to be 0. To achieve this, we add two "half" hat functions at the edges, so that we basically have a function space F_l of level l given by

$$F_l := \operatorname{span}\left\{\varphi_{l,i} \;:\; 0 \le i \le 2^l\right\} \tag{2.23}$$

However, for our purposes, the standard grid without boundary grid points is sufficient, so we will not use this variation. The boundary problem can be bypassed by smart scaling of the input data. One should not forget to choose a reversible way of scaling the data however, that is, a bijection.

After defining the nodal basis of level $\vec{l}$, let the **hierarchical basis** $V_{\vec{l}}$ of level $\vec{l}$ be defined as follows: For a given level $\vec{l}$ and dimension d, let the **index set** $I_{\vec{l}}$ be defined as follows:

$$I_{\vec{l}} := \left\{i \in \mathbb{N} : 1 \le i \le 2^l - 1, i \text{ odd}\right\}^d \tag{2.24}$$

Let the hierarchical subspace of level $\vec{l}$ $W_{\vec{l}}$ be defined as the space spanned by all basis functions of level $\vec{l}$ whose index is in the index set:

$$W_{\vec{l}} := \operatorname{span}\left\{\varphi_{\vec{l},\vec{i}}(x) \;:\; i_j \in I_l \forall j\right\} \tag{2.25}$$

This can be visualized quite nicely in two or three dimensions, cf. Fig. 2.5. Every cell in the figure shows the grid points of one hierarchical subspace $W_{\vec{l}}$. For example, the lower left cell shows the grid points of $W_{\vec{l}}$ with

$$\vec{l} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$$

and the lowermost grid point in that cell is the point which represents the basis function $\varphi_{\vec{l},\vec{i}}$ with

$$\vec{l} = \begin{pmatrix} 1 \\ 3 \end{pmatrix} \text{ and } \vec{i} = \begin{pmatrix} 1 \\ 7 \end{pmatrix}$$

Let the hierarchical function space of level $n \in \mathbb{N}$, V_n, be the union of all subspaces with level $\vec{l}$ such that $\left|\vec{l}\right|_\infty \le n$:

$$V_n := \bigoplus_{\left|\vec{l}\right|_\infty \le n} W_{\vec{l}} \tag{2.26}$$

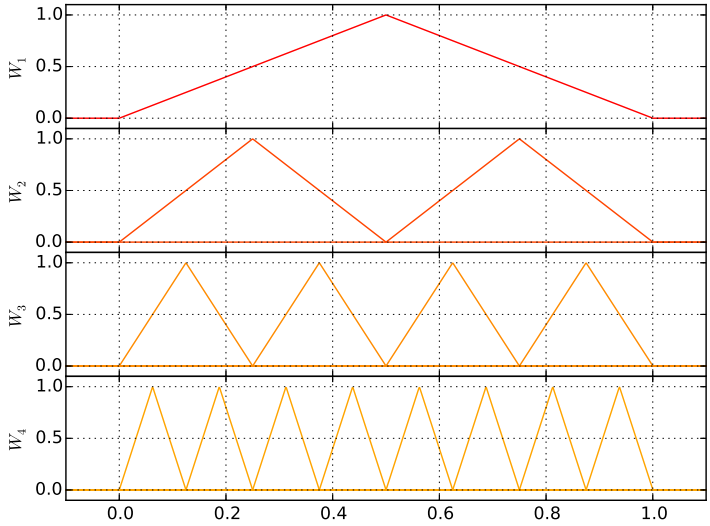


Figure 2.4: Hierarchical basis functions $\varphi_{l,i} \in V_4$. In contrast to the nodal basis, all levels up to the grid level are used.

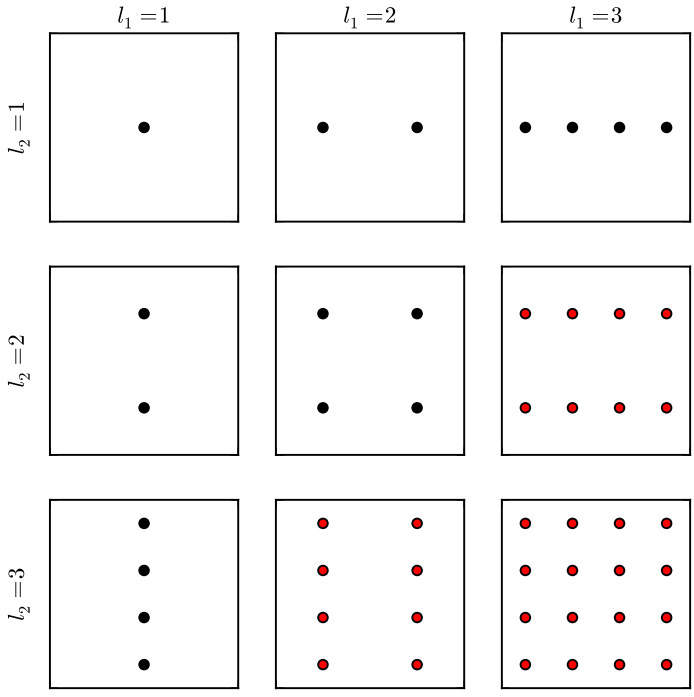


Figure 2.5: Hierarchical subspaces of dimension $d = 2$ and level $l_1, l_2 \leq 3$.

As an example, in Fig. 2.4, the basis functions spanning V_3 are plotted.

The interpolant $u(x) \in V_n$ of a function f in a hierarchical subspace is created by accumulating all basis functions $\varphi_{l,i} \in V_n$ with their respective weight $\alpha_{l,i}$:

$$u(x) = \sum_{|\vec{l}|_\infty \leq n, \vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \varphi_{\vec{l},\vec{i}}(x) \tag{2.27}$$

After defining the hierarchical basis, we now define the **sparse grid** as a grid where each grid point represents a basis function of a specific level vector and index vector, such that the function space $V_n^{(1)}$ is spanned over a subset of the same-leveled hierarchical basis. We define the sparse basis as $V_n^{(1)}$, as opposed to the hierarchical basis V_n, as the combination of all hierarchical subspaces $W_{\vec{l}}$ such that the sum of all levels $l_i, 1 \leq i \leq d$ is less or equal to the sum of the sparse basis level plus the dimension minus one:

$$V_n^{(1)} := \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}} \tag{2.28}$$

The criterion $|\vec{l}|_1 \leq n + d - 1$ for $n = 3$ and $d = 2$ can be seen in Fig. 2.5 as it is true for all subspaces with black grid points and false for all subspaces with red grid points. In Fig. 2.6, in the lower right, the resulting sparse grid of level 3 can be seen.

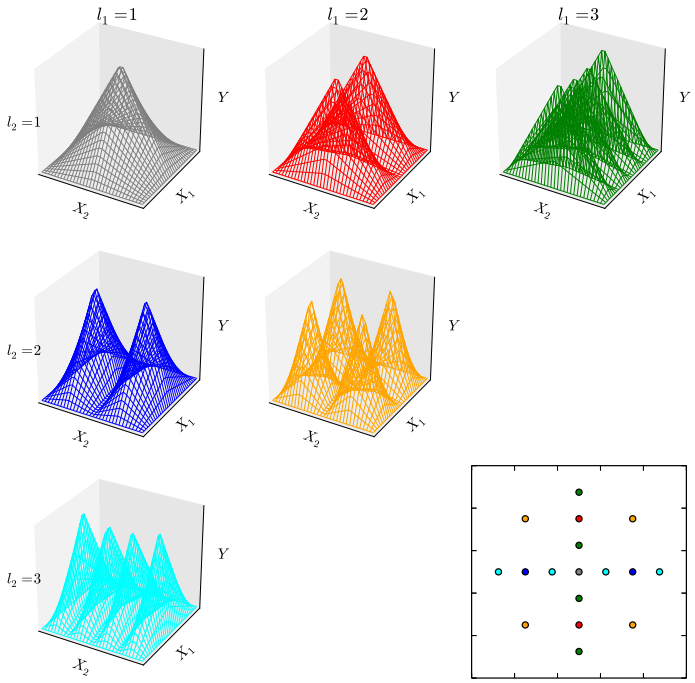


Figure 2.6: Sparse grid basis functions for dimension 2 and level 3. In the lower right, the grid points' positions are also shown, coded by color.

We now interpolate the function from Eq. 2.22 using the hierarchical function space V_4. According to our definition from Eq. 2.28, in the one-dimensional case, the hierarchical space is the same as the sparse space. In Fig. 2.7, the resulting weighted basis functions $\alpha_{l,i}\varphi_{l,i}, l \leq 4, i \in I_l$ are shown for an interpolant $u(x) \in V_4$. Their sum $u(x)$, given as the dashed red line, are the approximation of the function $f(x)$, which is shown as a blue line. The hierarchical basis is not as accurate as the nodal basis for approximating functions, however, it takes a lot less grid points in higher dimensions. That is because the number of grid points for a full grid – that is what a nodal basis means – of level n and dimension d has 2^n grid points per dimension, or $2^{n \cdot d}$ grid points in total. In Fig. 2.8, the basis functions up to each level were summed up. This way, we can see nicely how the hierarchical basis produces the approximation u.

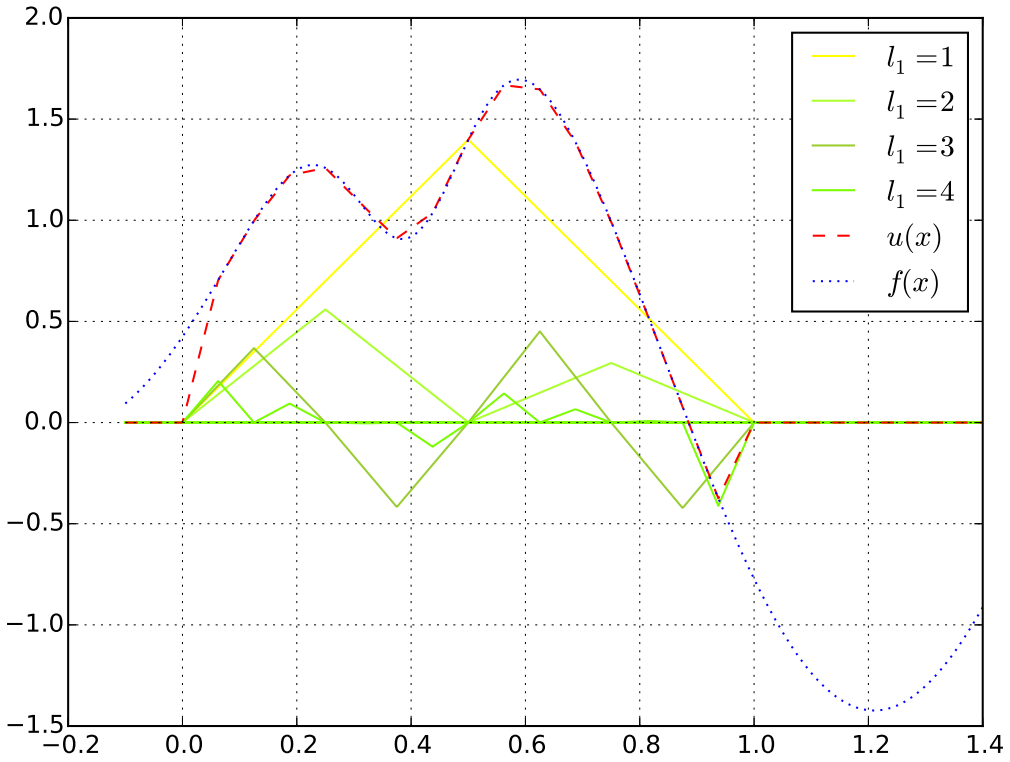


Figure 2.7: Interpolation of a function using a hierarchical basis. The interpolation u is giving by summing up the individual weighted basis functions $\alpha_{l,i}\varphi_{l,i}$. The function f is the same as for the nodal basis example in Fig. 2.3. However, for higher dimensions, the curse of dimensionality[Bel03] makes nodal basis functions too expensive. Here, the loss of precision is more than compensated[BG04] by the smaller grid size, and thus, less computational power required.

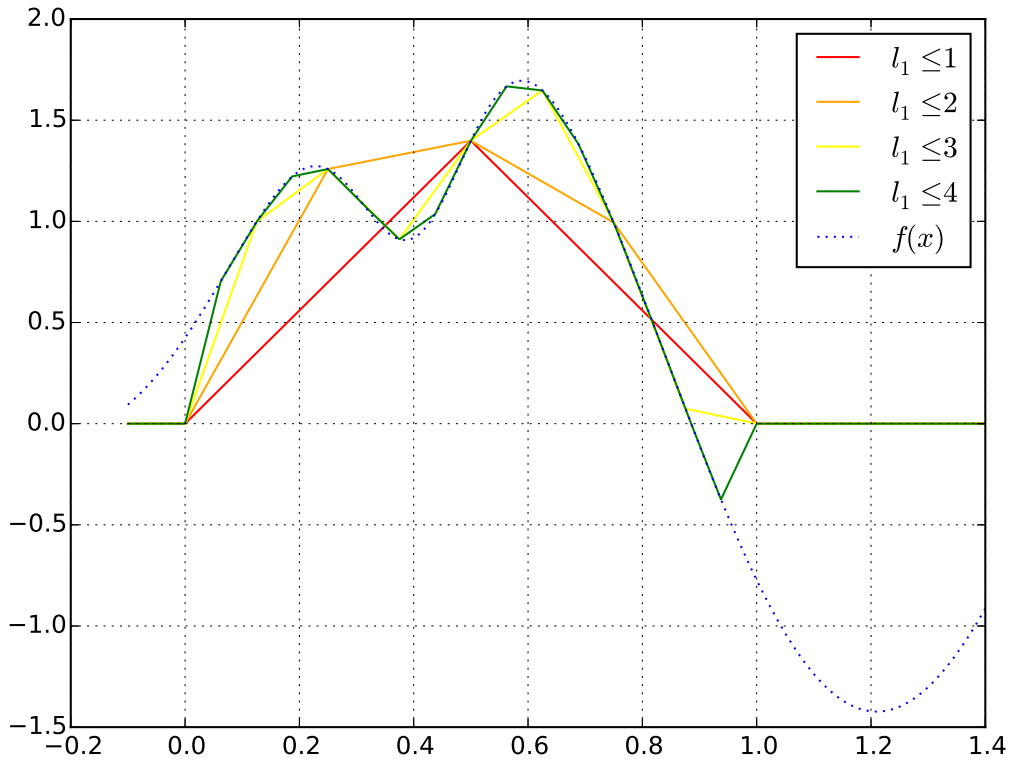


Figure 2.8: The hierarchical basis of level 4 used to interpolate the function f from Fig. 2.7. Instead of the basis functions $\alpha_{l,i}\varphi_{l,i}$, the accumulated basis functions up to that level $\hat{l}\left(\sum_{l=1}^{\hat{l}}\sum_{i\in I_l}\alpha_{l,i}\varphi_{l,i}\right)$ are shown, to demonstrate how they add up to produce u.

2.2.3 Adaptivity of Sparse Grids

Finer grids result in a better representation of the function. However, it also means an increase in grid size, which results in more computing power required for solving the grid and in higher storage space requirements. This poses an unnecessary problem when the represented function u is very simple in large parts of the attribute space, as it could be represented by a very low-level grid in those parts. Increasing the grid level over the whole attribute space is uncalled for in those situations.

For those situations, it is possible to increase the grid level only in small parts of the attribute space. This is called grid refinement by surplus and works as follows: We first select a grid point we want to refine. As it should be the grid point which would need refinement the

most, we take the absolute surplus $|\alpha_{l,i}|$ as criterion for grid point selection, as a high surplus indicates that the function has high variation around the grid point.

Refinement of a grid point in most cases means that we create its $2d$ children, that being its two children in each dimension:

$$\text{children}\left(\vec{l},\vec{i}\right) := \left\{\left(\begin{pmatrix} l_1+1 \\ l_2 \\ \vdots \\ l_d \end{pmatrix}, \begin{pmatrix} 2i_1 \pm 1 \\ i_2 \\ \vdots \\ i_d \end{pmatrix}\right), \left(\begin{pmatrix} l_1 \\ l_2+1 \\ \vdots \\ l_d \end{pmatrix}, \begin{pmatrix} i_1 \\ 2i_2 \pm 1 \\ \vdots \\ i_d \end{pmatrix}\right), \ldots, \left(\begin{pmatrix} l_1 \\ l_2 \\ \vdots \\ l_d+1 \end{pmatrix}, \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ 2i_d \pm 1 \end{pmatrix}\right)\right\} \tag{2.29}$$

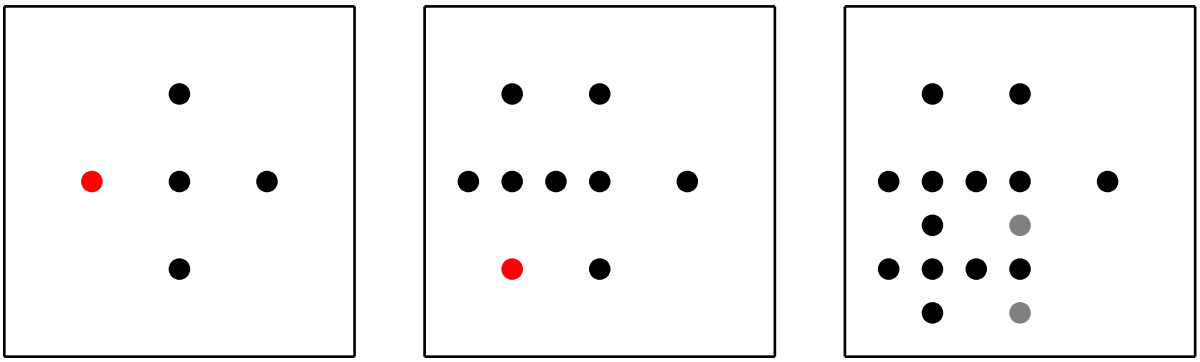


Figure 2.9: Adaptivity of a sparse grid demonstrated on two grid points. This example was taken directly from Pflüger[Pfl10, 21].

From Fig. 2.9, we can see how this works in detail: First, we have the grid point given by the tuple of it's level and it's index

$$P_a = \left(\begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}\right)$$

Now, we refine this point by it's $2d = 4$ children

$$P_b = \left(\begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}\right) P_c = \left(\begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 3 \end{pmatrix}\right) P_d = \left(\begin{pmatrix} 3 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}\right) P_e = \left(\begin{pmatrix} 3 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ 1 \end{pmatrix}\right)$$

Next, we refine P_c:

$$P_f = \left(\begin{pmatrix} 2 \\ 3 \end{pmatrix}, \begin{pmatrix} 1 \\ 5 \end{pmatrix}\right) P_g = \left(\begin{pmatrix} 2 \\ 3 \end{pmatrix}, \begin{pmatrix} 1 \\ 7 \end{pmatrix}\right) P_h = \left(\begin{pmatrix} 3 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}\right) P_i = \left(\begin{pmatrix} 3 \\ 2 \end{pmatrix}, \begin{pmatrix} 3 \\ 1 \end{pmatrix}\right)$$

Now, we notice that P_f and P_g miss one parent, so we need to add those parents (grey grid points in Fig. 2.9) to the grid:

$$P_j = \left(\begin{pmatrix}1\\3\end{pmatrix}, \begin{pmatrix}1\\5\end{pmatrix}\right) P_k = \left(\begin{pmatrix}1\\3\end{pmatrix}, \begin{pmatrix}1\\7\end{pmatrix}\right)$$

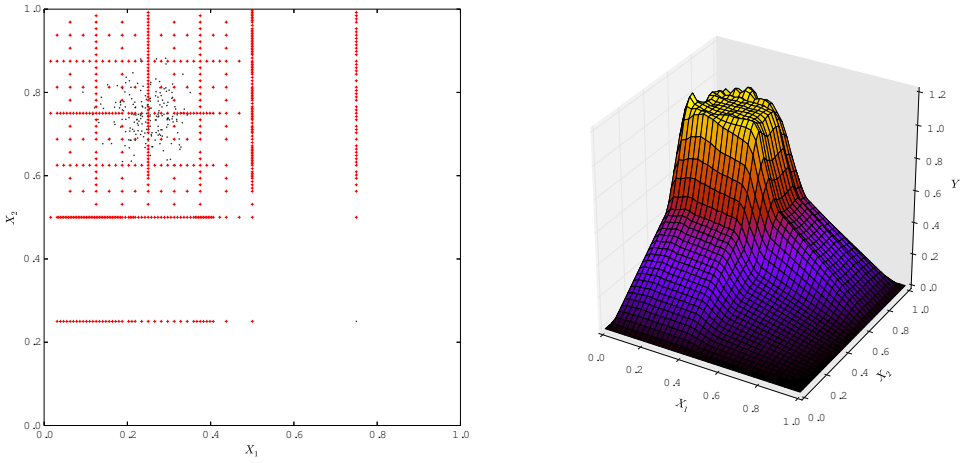


Figure 2.10: Interpolation of a function using an adaptive sparse grid. The red crosses in the left plot show the grid points, the black dots were the training data. On the right, the interpolated function is displayed.

In Fig. 2.10, a function was learned where all function values were ≈ 1 within an area around $(0.25 \mid 0.75)$. On the right of the plot, the interpolated function can be seen. On the left, the sparse grid's grid points are plotted in red. Around the data point cloud, the grid was heavily adapted. In sparsely populated areas, the grid points were not refined. By using an adaptive sparse grid, only 529 grid points were required to approximate the function. If a non-adaptive sparse grid of same maximum level would have been used, the number of grid points might easily have been around 50 000.

2.3 Regression

2.3.1 Mathematical definition

According to Sykes, regression is "a statistical tool for the investigation of relationships between variables. Usually, the investigator seeks to ascertain the causal effect of one variable

upon another—the effect of a price increase upon demand, for example, or the effect of changes in the money supply upon the inflation rate. To explore such issues, the investigator assembles data on the underlying variables of interest and employs regression to estimate the quantitative effect of the causal variables upon the variable that they influence."[Syk93].

On sparse grids, regression is the estimation $u \in V_n^{(1)}$ of a function $f : \mathbb{R} \to \mathbb{R}^d$, such that for an input point $\vec{x}$ it's function value $f(\vec{x})$ can be estimated by $u(\vec{x})$. As opposed to a interpolation, where the interpolating function is exact in all interpolated points, the regression function can – and in most cases does – have an error in the input points. In detail, we define a d-dimensional regression u as

$$u : \Omega \to \mathbb{R} \quad \text{with} \quad \Omega = \mathbb{R}^d \tag{2.30}$$

in general. However, for our numerical approach, we confine the attribute space to the d-dimensional hypercube over the interval $[0,1]$:

$$u : \Omega_n \to \mathbb{R} \tag{2.31}$$

To start, we have a set $\mathcal{P}$ of N input points $\vec{x}_i, y_i$ with

$$\mathcal{P} := \left\{ \left(\vec{x}_i, y_i \right) \in \Omega_n \times \mathbb{R} \right\}_{i=1}^{N} \tag{2.32}$$

and for each point $\vec{x}_i$ it's associated value y_i is the function value of f at $\vec{x}$:

$$f\left(\vec{x}_i\right) = y_i \tag{2.33}$$

We also have a function space V of functions

$$V := \{ v : \Omega_n \to \mathbb{R} \} \tag{2.34}$$

Last, we have a regularization operator $\mathcal{C}$ after Tichonov, Arsenin[TA77, 45–94, 211–235] and a regularization parameter $\lambda \in [0,1]$. Now, we can find an approximation u of f by solving the following equation:

$$u = \underset{u \in V_n}{\operatorname{argmin}} \left(\frac{1}{m} \sum_{i=1}^{m} \left(y_i - u\left(\vec{x}_i\right) \right)^2 + \lambda \mathcal{C}(u) \right) \tag{2.35}$$

that is, finding the $u \in V_n$ for which the MSE of the approximation of f is the smallest, considering a regularization operator $C(u)$. The first part

$$\frac{1}{m} \sum_{i=1}^{m} \left(y_i - u\left(\vec{x}_i\right) \right)^2$$

is the mean squared error between the approximation u in the input points and their real value. By using the mean squared error, we ensure that large discrepancies get weighted more than small ones. The second part

$$\lambda \mathcal{C}(u)$$

is the regularization term. It is used to prevent overfitting of the approximation to the data points. Given a sufficiently large n for a function space V_n and a regularization parameter $\lambda = 0$, we would get an approximation u which would satisfy

$$\forall \left(\vec{x}_i, y_i\right) \in \mathcal{P} : \; u\left(\vec{x}_i\right) = y_i \tag{2.36}$$

This is called overfitting and is generally considered bad, as the input points are only a sample of the function evaluations – and may contain a white noise error or measurement error – and fitting the function perfectly to the input points mostly results in very large errors for points not fitted. By choosing a good regularization term, we can smoothen the approximation so that overfitting gets reduced.

The regularization operator $\mathcal{C}$ was chosen as in [Pfl10, 104][Gar06] as the discrete ℓ_2 norm of the vector of hierarchical surplusses α of the grid for function on a sparse grid $u \in V_n^{(1)}$:

$$\mathcal{C}(u) := \|\vec{\alpha}\|_2^2 \tag{2.37}$$

2.4 Classification

2.4.1 Mathematical definition

Classification, as defined by Bishop, has the goal to "take an input vector **x** and assign it to one of K discrete classes $\mathcal{C}_k$ where $k = 1, \ldots, K$"[Bis06, 179]. Bishop also states that the classes are regionally disjunct in the most common scenario. A d-dimensional classificator function c over a finite set $\mathcal{C}$ of classes is defined as

$$c : \Omega_n \to \mathcal{C} \tag{2.38}$$

The classificator function c is learned using a **training dataset** $\mathcal{D}$ of size N, for which the classes of the input points are already given:

$$\mathcal{D} = \left\{(\omega_i, c_i) \in \Omega_n \times \mathcal{C}\right\}_{i=1}^{N} \tag{2.39}$$

On a learned classification, the classificator function can then estimate the class for an input point $p \in \Omega_n$.

2.4.2 Realization of a Classification

By Regression

A popular approach to solving a d-dimensional classification problem on a image set $\mathcal{C}$ is by learning $|\mathcal{C}|$ d-dimensional regressions. For each class $\mathcal{C}_i \in \mathcal{C}$, a regression is learned where all points in that class are mapped to +1 and all points not in that class are mapped to −1. Assuming we have a classification training dataset of size N

$$\mathcal{D}_{\mathcal{C}} \coloneqq \left\{ \left(p_j, \mathcal{C}_j \right) \in \Omega_n \times \mathcal{C} \right\}_{i=1}^{N} \tag{2.40}$$

with the training points

$$p_j \in \Omega_n \tag{2.41}$$

and the training points' classes

$$\mathcal{C}_j \in \mathcal{C} \tag{2.42}$$

So we define $|\mathcal{C}|$ training datasets $\mathcal{D}_i$:

$$\mathcal{D}_i \coloneqq \left\{ \left(p_j, \gamma_i(p_j) \right) \in \Omega_n \times \{-1, +1\} \right\}_{i=1}^{N} \tag{2.43}$$

for a binary classificator

$$\gamma_i : \Omega_n \to \{-1, +1\} \tag{2.44}$$

which is defined as

$$\gamma_i(p) \coloneqq \begin{cases} +1 & p \in \mathcal{C}_i \\ -1 & p \notin \mathcal{C}_i \end{cases} \tag{2.45}$$

We then learn $|\mathcal{C}|$ regressions $u_{\mathcal{C}_i}$ on the training datasets $\mathcal{D}_i$. To estimate the class of a new point p_n, we now evaluate the regression functions at the point and compare them for which one is the best one. For this, there are several approaches. One is to simply take the regression function with the highest value:

$$c(p_n) = \underset{\mathcal{C}_i \in \mathcal{C}}{\operatorname{argmax}} \, u_{\mathcal{C}_i}(p_n) \tag{2.46}$$

For several reasons, one being overfitting of a curve, the highest regression might not always be the best guess, so it is also possible – and sometimes necessary – to select the best regression as the one closest to the +1:

$$c(p_n) = \underset{\mathcal{C}_i \in \mathcal{C}}{\operatorname{argmin}} \left| 1 - u_{\mathcal{C}_i}(p_n) \right| \tag{2.47}$$

For my classification implementation, I undertook experiments comparing the two decision approaches (Eq. 2.46, Eq. 2.47), and concluded that the results did not differ enough to make a difference. Thus, the maximum approach from Eq. 2.46 was used.

The whole process is also listed here as an algorithm:

Algorithm 2.1 Learning a classification by learning $|\mathcal{C}|$ regression functions.
For each class, a binary classificating vector $\Gamma_i \in \{-1,+1\}^N$ is created and learned as regression.

Require: Training data $\mathcal{D} = P, C$
with $P = \{\omega_k \in \Omega_n\}_{k=1}^N$, $C = \{c_k \in \mathcal{C}\}_{k=1}^N$
for $i = 1 \to |\mathcal{C}|$ **do**
$\Gamma_i \leftarrow \gamma_i(P)$
$u_{\mathcal{C}_i} \in V_n^{(1)} \leftarrow$ `learn_regression`(P, Γ_i)
end for

To evaluate the class of an arbitrary point $\vec{x} \in \Omega_n$, we calculate it's most likely containing class $c(\vec{x})$ as follows:

Algorithm 2.2 Evaluating a classification from $|\mathcal{C}|$ regressions.

Require: Trained regressions $u_{\mathcal{C}_i} \quad \forall \mathcal{C}_i \in \mathcal{C}$
Require: Test point $\vec{x} \in \Omega_n$
for $i = 1 \to |\mathcal{C}|$ **do**
$y_i \leftarrow u_{\mathcal{C}_i}(\vec{x})$
end for
return $\underset{\mathcal{C}_i \in \mathcal{C}}{\operatorname{argmax}} y_i$

By density estimation

The approach by regression has several flaws, one of them being the difficulty to decide on which regression function fits the input point best. Another problem is how a regression function is behaving in places where there were no points in the training dataset. As the regression had no constraints whatsoever there, the function estimation in those areas does virtually have undefined behavior. Real world datasets unfortunately often have large areas of the attribute space which are very sparsely, if ever, populated.

The resulting effect is that for each added dimension of the dataset, the number of training data points also has to be increased. Ideally, this is done in such a way that there are no sparsely populated areas of the attribute space. This criteria is also called Hughes' phenomenon[Hug68]. However, for real world datasets, the criteria can rarely be satisfied. For example, in a dataset which contains an attribute for air temperature and one for wind speed, we will notice that the attribute subspace spanned by these two attributes contains some large sparsely populated areas in the high wind corners. This is absolutely natural, as there is seldom much wind on a very hot day. So, without very thorough preprocessing of the data, we will always have sparsely populated areas within the attribute space.

Therefore, another method for classification is sometimes the better approach. This method is called density estimation. Density estimation, according to Ripley, is the process of estimating the probability density function of the data, that is, the probability of a data point being located at a certain position in the attribute space, based only on a subset of data points scattered over the attribute space[Rip96][Rip94][GP14][EJHS00]. The resulting density estimator then is a function

$$\kappa : \Omega_n \to \mathbb{R} \tag{2.48}$$

which denotes the approximate density of data points in any point within the attribute space.

We use sparse grid density estimation on a grid with underlying sparse grid function space $V_n^{(1)}$, such that for a training dataset $\mathcal{S} = \{x_i \in \Omega_n\}_{i=1}^N$ and for an initial guess for the density at the training points f_e the density estimator κ is given as

$$\kappa = \underset{u \in V_n^{(1)}}{\operatorname{argmin}} \left(\int_{\Omega_n} \left(u(x) - f_e \right)^2 \mathrm{d}x + \lambda \mathcal{C}(u) \right) \tag{2.49}$$

as proposed by Garcke and Pflüger[GP14]. The regularization term $\mathcal{C}(u)$ is chosen to control the tradeoff between smoothness of κ and error of the estimation.

Ideally, the density should be ≥ 0 throughout the attribute space, however, with sparse grid density estimation, the density may be negative locally due to overregularization. This estimation is not perfect, but good enough for our purposes.

To do classification with density estimation, we again generate $|\mathcal{C}|$ training datasets $\mathcal{D}_i$ from our training dataset $\mathcal{D}_\mathcal{C}$ from Eq. 2.40. However, this time, we don't use a binary classificator function, so our training datasets have function values and do not have the size of our ultimate dataset $\mathcal{D}_\mathcal{C}$. Instead, they contain only the points which are in the respective class.

$$\mathcal{D}_i := \left\{ p_j \in \Omega_n \ : \ \mathcal{C}_j = \mathcal{C}_i \right\}_{j=1}^N \tag{2.50}$$

We now train $|\mathcal{C}|$ density estimations $\kappa_{\mathcal{C}_i}$. After doing this, estimating the class for an input point p_n is done by taking the class which has the highest density estimation at that point:

$$c(p_n) = \underset{\mathcal{C}_i \in \mathcal{C}}{\operatorname{argmax}} \left(\kappa_{\mathcal{C}_i}(p_n) \right) \tag{2.51}$$

It is now also easy to decide that a point's class cannot be estimated by defining a density threshold τ_{density}, and stating that the point's class cannot be estimated if no density estimation $\kappa_{\mathcal{C}_i}(p_n)$ manages to exceed this threshold:

$$c(p_n) = \begin{cases} \operatorname{argmax}_{\mathcal{C}_i} \kappa_{\mathcal{C}_i}(p_n) & \text{if } \kappa_{\mathcal{C}_i}(p_n) \geq \tau_{\text{threshold}} \\ \mathcal{C}_{\text{none}} & \text{else} \end{cases} \tag{2.52}$$

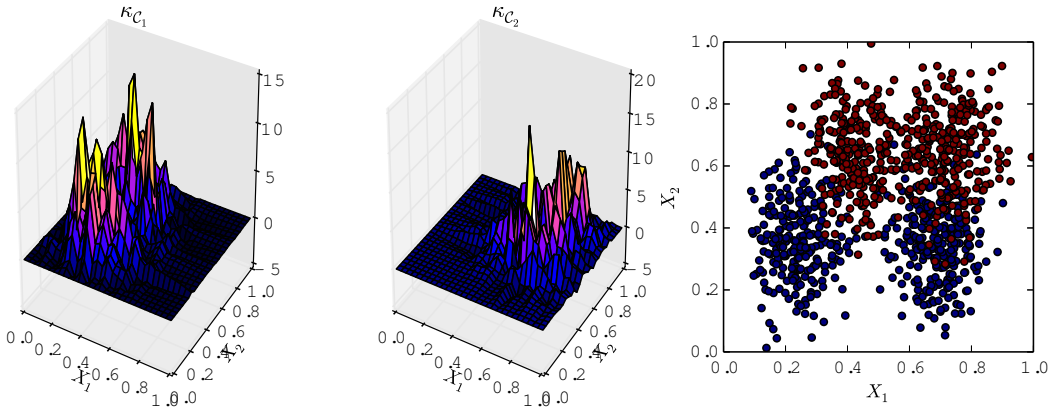


Figure 2.11: Using sparse grid density estimation, the density functions $\kappa_{\mathcal{C}_1}$, $\kappa_{\mathcal{C}_2}$ are calculated. The Ripley dataset[Rip94] contains two classes whereof each is generated by two overlapping Gaussian distributions.

In Fig. 2.11, the density estimators for the two-class **Ripley** dataset[Rip94] are shown. The density estimator evaluates to zero where no data points are given, and classification of new data points is pretty straightforward for this example as the density functions do not overlap in a larger area. The classes of the Ripley dataset can be seen by color in the scatter plot to the right in Fig. 2.11.

2.5 Clustering

2.5.1 Mathematical definition

Kriegel et al. define clustering as follows: "Clustering refers to the task of identifying groups or clusters in a data set"[KKSZ11]. This is done by topological resemblance of points, that is, points which lie closely together are assigned to the same cluster. For a d-dimensional input set of points

$$\mathcal{P} \subset \Omega_n \tag{2.53}$$

it's clustering function

$$c : \mathcal{P} \to \mathcal{C} \tag{2.54}$$

finds clusters based on certain criteria and then assigns each point in $\mathcal{P}$ to a cluster. We restrict the number of clusters as

$$1 \leq |\mathcal{C}| \leq |\mathcal{P}| \tag{2.55}$$

That means that in the edge cases, either every point is in the same cluster or every point is in it's own cluster. There can be no cluster with no points in it. For general purposes, we ignore clusters below a size threshold τ_{size}, as they most probably are not relevant:

$$\tau_{\text{size}} \leq |\mathcal{C}| \leq |\mathcal{P}| \tag{2.56}$$

2.5.2 Realization

k-nearest-neighbors

In the k-nearest-neighbors approach to clustering, clusters are detected by density estimation. This means that, for the purpose of this method, a cluster is defined as all data points within the attribute space within a contiguous region with a high density of data points[KKSZ11]. The clusters are then delimited by contiguous regions with a low density of data points. We will consider data points in regions where the density is below a threshold as noise or outliers.

For one, we will do a density estimation

$$\kappa : \Omega_n \to \mathbb{R} \tag{2.57}$$

over the attribute space, where ideally the density should never fall below zero. Also, we will construct a directed, unweighed graph

$$G = (\mathcal{P}, E) \tag{2.58}$$

over the data points $\mathcal{P}$, where for a cutoff distance δ

$$E := \left\{ (p_1, p_2) \,:\, \|p_1, p_2\|_2 \leq \delta \text{ and } \left| \left\{ (p_1, p_a) \,:\, p_a \neq p_2 \text{ and } \|p_1, p_a\|_2 \leq \|p_1, p_2\|_2 \right\} \right| < k \right\} \tag{2.59}$$

We now have a graph with a directed edge from every point to every other point within cutoff distance. Also, only the k edges from each point with least length are in the graph, which is assured by the second argument of the set notation. An example of such a graph can be seen in Fig. 2.12a.

Now, for a density threshold τ_v, we iterate over all nodes $v \in V$ of the graph and remove all nodes – and, subsequently, all edges which connect to these nodes in any way – from the graph where the density estimation does not exceed the threshold:

$$G' = \left(V', E'\right) \tag{2.60}$$

with

$$V' = \left\{ v \,:\, v \in V \text{ and } \kappa(v) \geq \tau_v \right\} \tag{2.61}$$

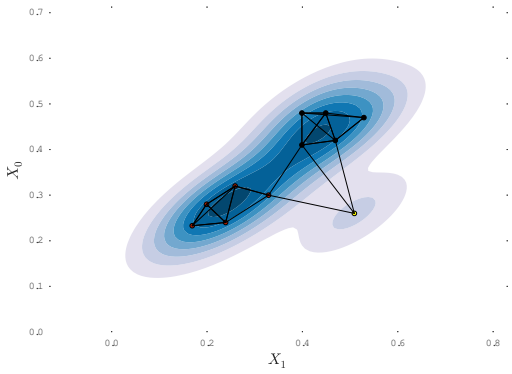
and

$$E' = \left\{ e = \left(v_i, v_j\right) \,:\, e \in E \text{ and } v_i, v_j \in V' \right\} \tag{2.62}$$

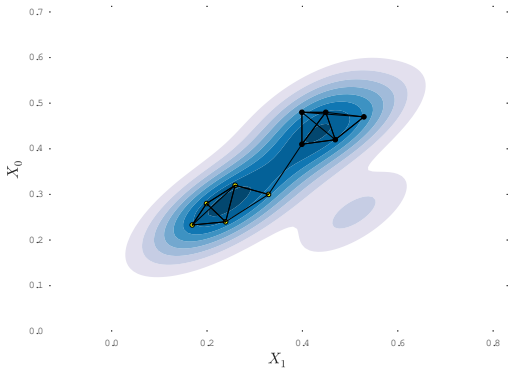
An example for the graph after this step can be seen in Fig. 2.12b.

As a last measure, we remove all edges which traverse areas of the attribute space where the density estimation is below a threshold τ_e. This can – but does not have to – be a different value than τ_v. Of course, we cannot evaluate the density estimation for every point along the edge e from v_i to v_j, so in practice we only evaluate it at a finite number of points along the edge, for example in the middle:

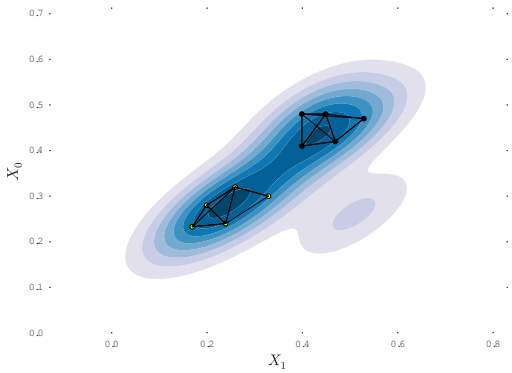
In our resulting graph $G'' = \left(V', E''\right)$ (cf. Fig. 2.12c), all vertices with too low density estimation value and all edges traversing low density areas of the attribute space are removed. Now, we can interpret each connected component of G'' as a distinct cluster. This makes sense because we ensured that only edges and vertices with high densities remained, so a gap between two clusters should no longer contain any vertices or edges.



(a) First, a graph is generated from the points with $k = 3$ edges from each point to it's k nearest neighbors.



(b) Then, the vertices with a density estimation below treshold are removed from the graph.



(c) In the third step, the edges which have too low density at their middle point are removed. Now, every connected component is a cluster.

Figure 2.12: Cluster detection by k-nearest-neighbors technique using sparse grid density estimation. The density estimator is given as colored and filled isolines.

2.6 k-fold Cross Validation

As defined by Witten et al.,k-fold cross validation is a method to either tune a data mining algorithm or assess the usefulness of a dataset for data mining[WFH11]. It is a non-exhaustive validation method, meaning it is not as accurate as an exhaustive test, but gives a rough measure. The benefit over an exhaustive test is that those take way too long for practical use.

k-fold cross validation is done by partitioning a dataset D in k equally large partitions D_i. Then, for each $1 \leq i \leq k$, the data mining method in question is applied using all partitions but one as the training dataset and the remaining partition as the test dataset:

$$D_{\text{train},i} \coloneqq \bigcup_{\substack{j=1 \\ j \neq i}}^{k} D_j \qquad D_{\text{test},i} \coloneqq D_i \tag{2.63}$$

The overall hit rate – for classifications – or MSE/NMSE – for regression – is then given by the mean of the individual results.

The reasoning behind k-fold cross validation is to reduce so-called type III errors[OD03], that is, to verify that the hypothesis derived from the data is accurate[Mos48]. It can be used really well to prevent **overfitting**, as described on page 28, because the data are learned multiple times in different variations and the probability that a genuinely bad overfitting happens is drastically reduced. A partition count of $k = 10$ is often used[MDA05], however, there is no real norm.

For my work, I used k-fold cross validation both for regression and regression-based classification to tune the regularization parameter λ of the regression functions. By running k-fold cross validation on a small training dataset for different λ, the best λ from a list of candidates was found for each experiment. The list of λ was generated as the negative powers of ten ($1 \ldots 10^{-8}$). For most experiments, values of λ between 10^{-3} and 10^{-6} were used.

3 Used Tools and Data

3.1 SG++

SG++ is an open source toolbox for spatially adaptive sparse grids. SG++ [Tec10] is a project of the University of Stuttgart and was originally created by Dr. rer. nat. Pflüger during his dissertation[Pfl10]. It contains a C++ API for a great number of data mining methods, using different specialized algorithms. The data mining done throughout this thesis was done nearly exclusively using the SG++ API.

3.2 The Used Data

3.2.1 The IPV data

The data was collected in the years 2006 to 2014 by the Institute of Photovoltaics at the University of Stuttgart by Hendrik Adler[Adl13][WA14] and Bastian Zinsser[Zin10]. Over the duration of seven years, a total of 210 sensors measured the environment at three locations. Those locations were Stuttgart in Germany, Nikosia in Cyprus and Cairo in Egypt. However, as the data of different locations would have no correlation whatsoever, the data were narrowed down to the 111 sensors which were located in Stuttgart.

The data contains a number of sensors containing environmental parameters, such as air temperature and wind speed. There were also sensors measuring electrical values and module temperature for 13 photovoltaic modules. In table 3.1 all sensors located in Stuttgart are listed with their respective scope.

Table 3.1: Description of the IPV data sensors located in Stuttgart. Seven sensors respectively measure the values for one module. Module enumeration starts at 4 according to the provided data. Additionally, solar radiation, temperature and wind data were measured.

Sensor ID	Measured quantity	Unit
11	Wind speed	$m \cdot s^{-1}$
12	Wind direction	°
13	Room temperature[1]	°C
	Module 4	
14	Direct voltage	V
15	Direct current	A
16	Module temperature	°C
17	DC power	W
18	DC energy	kWh
19	Effective AC power	W
20	AC energy	kWh
21	Outdoor air temperature	°C
22	Solar radiation (Pyranometer)	$W \cdot m^{-2}$
23	Solar radiation 1	$W \cdot m^{-2}$
24	Solar radiation 2	$W \cdot m^{-2}$
25	Solar radiation 3	$W \cdot m^{-2}$
26	Solar radiation 4	$W \cdot m^{-2}$
27	Solar radiation 5	$W \cdot m^{-2}$
28	Solar radiation 6	$W \cdot m^{-2}$
29	Solar radiation 7	$W \cdot m^{-2}$
30	Solar radiation 8	$W \cdot m^{-2}$
31 through 37	Sensors of module 5	see 14 through 20
38 through 44	Sensors of module 6	see 14 through 20
45 through 51	Sensors of module 7	see 14 through 20
52 through 58	Sensors of module 8	see 14 through 20
59 through 65	Sensors of module 9	see 14 through 20
66 through 72	Sensors of module 10	see 14 through 20
73 through 86	Sensors of module 11	see 14 through 20
87 through 93	Sensors of module 12	see 14 through 20
94 through 100	Sensors of module 13	see 14 through 20
101 through 107	Sensors of module 14	see 14 through 20
108 through 114	Sensors of module 15	see 14 through 20
115 through 121	Sensors of module 16	see 14 through 20

The data were already preprocessed in a way, as they were captured using a *complex event driven system*[Adl13, 20-23] which used piecewise constant upwind interpolation for missing data points. Piecewise constant upwind interpolation is the most simple method of repairing missing values in a time series, as it simply means filling gaps with the last known value. Thus, no preprocessing had to be done for single missing values, but because of the event driven system, the metering time period of 1s was not used to its full potential, as measured data would stay at the exact same value for longer periods of time. Most likely, a hysteresis was employed and only larger changes in value would even be registered by the event driven system. This was unfortunately a reduction in accuracy on the data which could not be reversed.

Additionally, the data were stored in a very inefficient way, using a separate PostgreSQL table for each sensor, with the timestamps as index. As the data contained 210 sensors with per-second values for roughly seven years, and the timestamp was stored separately for each of the 210 sensors, this lead to a database of roughly 2.5TB size, where 1TB was occupied with the data for Stuttgart. I preprocessed the data by merging all data into one table, with one column for each sensor. This resulted in a final database size of 100GB for the sensors from Stuttgart. Also, database access times were drastically reduced.

The data still contained missing values where the piecewise constant upwind interpolation was no longer an option because the amount of consecutive missing values would be too big. This typically meant that whole days or even months of data missed for a sensor. These missing data could unfortunately not be repaired.

3.2.2 The DWD data

Furthermore, for the classification scenario, weather warnings issued by the German Weather Service (*Deutscher Wetterdienst*, DWD) in the time from December, 2011 to July, 2015. The data are publicly available as the DWD is a governmental institution.

This data had multiple issues. For one did it only intersect with the IPV dataset in a period of only roughly three years, which of course results in far less available data points. Also, it only contained warnings, not actual weather events. This made the predictions less accurate, as not every warning was justified. The third and largest issue was that the weather warnings were for the city of Stuttgart, which generally has slightly different weather than the university. That is caused by the difference in geography: Stuttgart is situated in a basin with an altitude of roughly 250 meters above sea level, surrounded in every direction by hills of 500 meters above sea level. The university is situated outside the basin, on the heights, see Fig. 3.1. Thus, fog warnings for Stuttgart mostly don't apply to Stuttgart-Vaihingen, where the university is located. Also, while it very seldom snows in the city of Stuttgart, it does considerably more often in Stuttgart-Vaihingen.

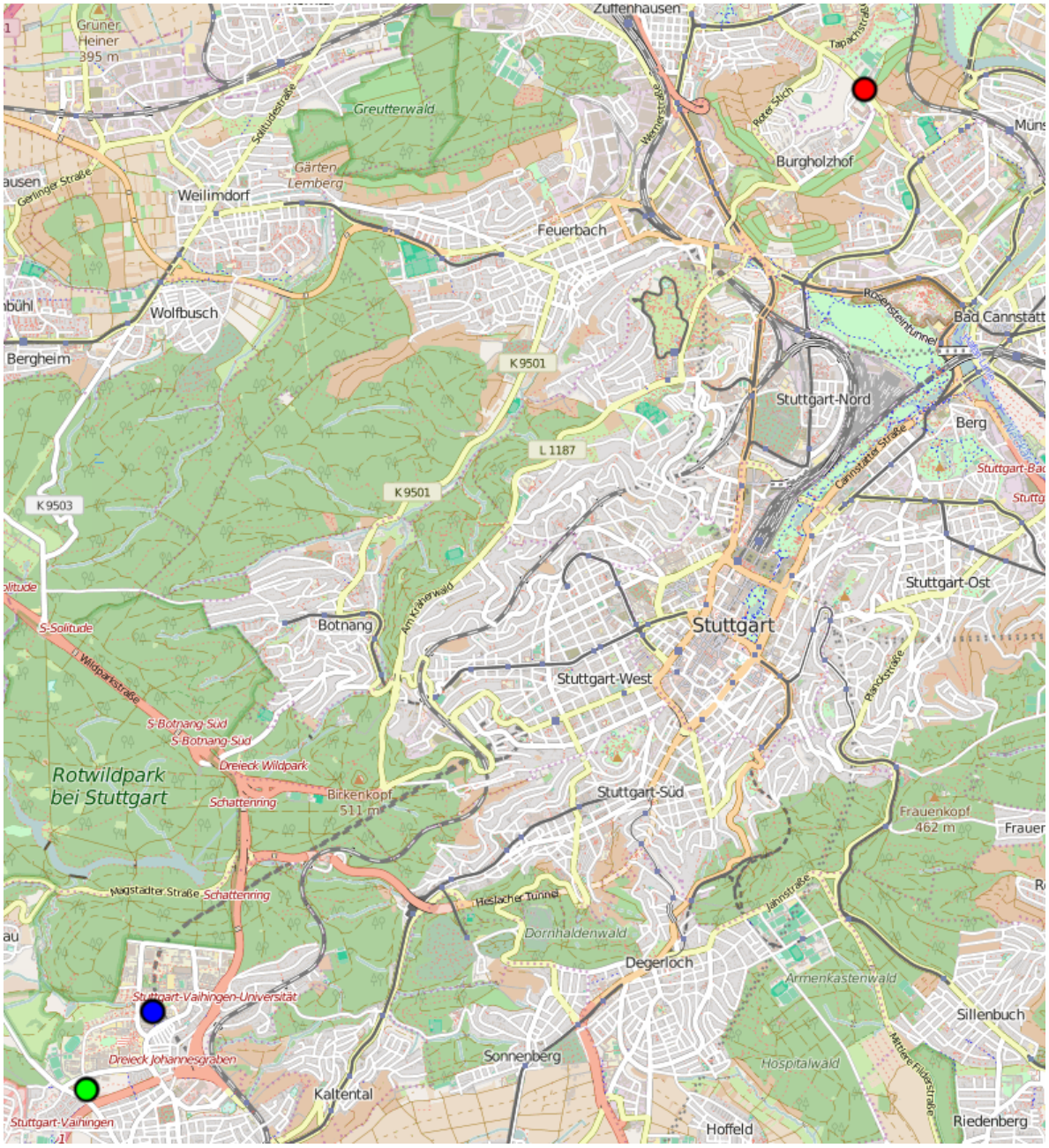


Figure 3.1: Map of the locations for which the data are relevant. The DWD data were issued for Stuttgart Schnarrenberg, which lies in the basis and is marked as a red dot on the map. The IPV data were collected on campus of the University in Stuttgart-Vaihingen, which is marked as a blue dot in the map. The IWS data were collected in Stuttgart-Vaihingen in Lauchäcker, which is marked as a green dot. The difference in altitude and location between the DWD data and the other data can be seen clearly.
Source: `http://www.openstreetmap.de/` . OpenStreetMap™ is open data, licensed under the Open Data Commons Open Database License (ODbL) by the OpenStreetMap Foundation (OSMF).

Table 3.2: DWD weather warning types, sorted by assigned class.

Warning type	Class
RAIN	Continuous rain
	Abundant continuous rain
	Torrential rain
	Severe torrential rain
FOG	Fog
THUNDER	Thunderstorm
	Heavy thunderstorm
	Extreme thunderstorm with torrential rain
SNOW	Snowfall
	Snowfall and snow drift
	Heavy snowfall and snow drift
Unused events	
Storm	Squall
	Heavy squall
	Gale-force squall
	Gusts of wind
Frost	Frost
	Severe frost
	Warning: Frost (Winter)
	Slippery roads
	Warning: Slippery roads
	Black ice
Thaw	Thaw
	Severe thaw

The data still was mostly usable. It contained of a time frame, a warning type, and in some cases an additional comment. The types of warnings were manifold. For example, there were multiple types of warning for snow, rain, and thunderstorm, as listed in Tab. 3.2. After noticing that using all those warning classes didn't yield very good results, I reduced the variety of warnings to the general ones, which were rain, thunderstorms, fog and snow.

3.2.3 The IWS data

Another dataset with weather data measured in Stuttgart-Vaihingen was provided by the Institute for Water and Environment Modeling of the University of Stuttgart (IWS). The data contained some attributes that seemed relevant and were not included in the IPV data, like

relative humidity and precipitation. The data were measured in Lauchäcker, only roughly 500m away from the photovoltaic modules, see Fig. 3.1. This made the data very attractive, as they were far more relevant for the location of the photovoltaic data than the DWD data.

Table 3.3: Sensor descriptions for the IWS data from Lauchäcker.

Sensor ID	Measured quantity	Unit
Ta_2m	Air temperature 2m above ground	°C
Ta_18m	Air temperature 19m above ground	°C
rh_2m	Relative humidity 2m above ground	%
rh_18m	Relative humidity 19m above ground	%
p	Air pressure	hPa
rr_01 ... rr_10	Precipitation	mm
u_2m	Wind speed 2m above ground	$\frac{m}{s}$
u_19m	Wind speed 19m above ground	$\frac{m}{s}$
dd_2m	Wind direction 2m above ground	°
dd_2m_sigma	Wind direction standard deviation (σ) 2m above ground	°
dd_19m	Wind direction 19m above ground	°
dd_19m_sigma	Wind direction standard deviation (σ) 19m above ground	°
TC_01	Temperature of precipitation	°C
TC_02	Temperature of precipitation	°C
More sensors which were not relevant for our purposes.		

In Tab. 3.3, a subset of the attributes measured by the IWS are listed. I ended up using the air pressure, relative humidity 19m above the ground – as the values were measured some distance from the photovoltaic modules and this were the less local values and because the photovoltaic modules are located on a rooftop – and precipitation. Wind direction, air temperature and wind speed were already part of the IPV dataset.

3.3 Other Tools and Libraries

For storing the data and the generated preprocessed data, a large PostgreSQL[Pos15] database was used. For operations on the PostgreSQL database, the `libpq` and `libpqxx`[Ver15] libraries for PostgreSQL in C/C++. Also, the data wrangling library **pandas**[McK12] for Python was used to restructure and preprocess data. The data mining and visualizing program **Weka**[Mac15] was used to look at potential attribute subspaces, however I did not use it on a larger scale. For the plots and visualizations in this thesis the open source plotting library **matplotlib**[Hun12] was used.

4 Generating Datasets

4.1 Database Preprocessing

Preprocessing of the data was largely done before the start of this thesis. The IPV data were already used in a student project I participated in in the beginning of 2015. There, I completely rebuilt the database because pf performance issues with the old database, which also had a size of 2.5TB.

The DWD data were given as a small file with only the types of warnings and a start and end timestamp. In order to efficiently to SQL join operations on this data, I generated a table with one row for every second in the time period covered by the weather warning data.

The IWS data were given as a HDF5 database, which is an hierarchical data format. The data were extracted from the HDF5 database and then inserted into the PostgreSQL database.

4.2 Delay Embedding

4.2.1 Introduction to Time Series Analysis

A time series is a sequence of successive data points. Box and Jenkins define a time series as "a set of observations made sequentially in time"[BJ76, 21]. Usually, the time difference between two measurements is the same for every two adjacent data points. Time series analysis is used to find reoccurring patterns in data and then predict those based on new data. On a more formal level, for a time-dependent measured attribute X, its *univariate* time series $\mathcal{T}_X$ of length k can be denoted as a tuple of timestamp-value pairs

$$\mathcal{T}_X^{(k)} = \left\{ \left(t_i, X^{(t_i)} \right) \in \mathbb{R} \times \mathbb{R} \right\}_{i=1}^{k} \tag{4.1}$$

for a chosen time difference between measurements of Δt. For a d-dimensional attribute space, the *multivariate* time series of length k – $\mathcal{T}_{\Omega_n}^{(k)}$ – consists of pairs of timestamps and d-dimensional vectors of attribute values.

$$\mathcal{T}_{\Omega_n}^{(k)} = \left\{ \left(t_i, \vec{X}^{(t_i)} \right) \in \mathbb{R} \times \mathbb{R}^d \right\}_{i=1}^{k} \tag{4.2}$$

An arbitrary time series can be separated into its *index* $\mathcal{I}_t$, a set of timestamps, and its *attribute space* Ω_n, such that

$$\mathcal{T}_{\Omega_n}^{(k)} = \mathcal{I}_t \mid\mid \Omega_n \tag{4.3}$$

Let the concatenation of two vectors of arbitrary dimensions be defined as

$$\circ : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n+m} \tag{4.4}$$

$$\vec{v} \circ \vec{w} := \begin{pmatrix} v_1 \\ \vdots \\ v_n \\ w_1 \\ \vdots \\ w_m \end{pmatrix} \tag{4.5}$$

Let for the purpose of this definition a real be treated as a one-dimensional vector, such that for $\vec{v} \in \mathbb{R}^n, r \in \mathbb{R}$:

$$\vec{v} \circ r = \begin{pmatrix} v_1 \\ \vdots \\ v_n \\ r \end{pmatrix} \tag{4.6}$$

Let the concatenation operator $\circ$ be defined for time series as follows: Let $\mathcal{A}, \mathcal{B}$ be two attribute spaces of arbitrary dimension, and $\mathcal{T}_{\mathcal{A}}^{(k)}, \mathcal{T}_{\mathcal{B}}^{(l)}$ be their respective time series. Then, let the concatenation be defined as

$$\mathcal{T}_{\mathcal{A}}^{(k)} \circ \mathcal{T}_{\mathcal{B}}^{(l)} := \left\{ \left(t_i, \vec{a}_i \circ \vec{b}_i \right) \;\middle|\; (t_i, \vec{a}_i) \in \mathcal{T}_{\mathcal{A}}^{(k)} \wedge \left(t_i, \vec{b}_i \right) \in \mathcal{T}_{\mathcal{B}}^{(l)} \right\}_{i=1}^{m \leq \min\{k,l\}} \tag{4.7}$$

in order to be able to merge two time series. Observe how the number of timestamp-vector tuples depends on how many timestamps are part of both time series. Ideally – and mostly – the time series we merge will have the same index (set of timestamps).

Time series data can then, as the most simple variant, be learned by a regression and the regression function obtained this way can be used to predict values on newly measured data. For instance, if we were to learn the function $f(t) = \sin(t)$ as a time series of length 2 with $\Delta t = \frac{\pi}{2}$, we would very soon have learned the function in Tab. 4.1 from the time series data.

Table 4.1: Time series prediction of $\sin(x)$ for a time series length of 3 – attribute space size 2 – and $\Delta t = \frac{\pi}{2}$, for four example value pairs $X^{(t-2)}, X^{(t-1)}$.

$X^{(t-2)}$	$X^{(t-1)}$	Prediction of $X^{(t)}$
0	1	0
1	0	-1
0	-1	0
-1	0	1
⋮	⋮	⋮
$\sin(t-2\Delta t)$	$\sin(t-\Delta t)$	$\sin(t)$

4.2.2 Using Delay Embedding

Delay embedding is the process of generating additional attributes from historical values of already present attributes' time series[SYC91][WG94]. Hereby, for a d dimensional attribute space, the resulting attribute space with embedded attributes may at most have a dimensionality of $2d+1$ according to Takens[Tak81], Sauer et al.[SYC91], Shalizi[Sha06]. While it is possible to simply embed historical values of an attribute as new dimension of the attribute space – that is, as a new attribute – it is in most cases better to embed a newly generated attribute A which is in some way generated by a function ξ. For a multivariate time series $\mathcal{T}_{\mathcal{A}}^{(k)}$ of dimension d and size k of a set $\mathcal{A}$ of attributes

$$\mathcal{A} = \{A_i\}_{i=1}^{d} \tag{4.8}$$

with

$$\mathcal{T}_{\mathcal{A}}^{(k)} = \left\{ (t_i, \vec{a}_i) \in \mathbb{R} \times \mathbb{R}^d \right\}_{i=1}^{k} \tag{4.9}$$

a new time series potentially containing additional attributes as data rows is generated using the existing time series $\mathcal{T}_{\mathcal{A}}^{(k)}$ and applying a function to it.

Let Ξ be the space of functions from one time series to another

$$\Xi := \left\{ \xi_i : \left(\mathbb{R} \times \mathbb{R}^d \right)^k \to \left(\mathbb{R}^{\tilde{d}} \times \mathbb{R} \right)^j \right\}_{i \in \mathbb{N}} \tag{4.10}$$

where $k \in \mathbb{N}$ and $j \in \mathbb{N}$ are the sizes of the respective time series, which do not necessarily need to be equal, and $d \in \mathbb{N}$ and $\tilde{d} \in \mathbb{N}$ are the dimensions of the in- and output attribute spaces.

Thus, from one arbitrary time series of arbitrary dimension, we can generate a time series of arbitrary dimension with arbitrary properties by choosing a fitting ξ_i. Usually, a function ξ_i is selected such that $j = k+1$, embedding one additional attribute's time series into the original time series.

In most cases, we limit ourselves to embed an attribute which is only generated from one other attribute. It is however important to keep in mind that in general, generation from multiple existing attributes is possible. For most cases, embedding historical values with delay $j\Delta t$ of an attribute as new attribute like in Tab. 4.1 specializes the function $\xi \in \Xi$ as, for the multivariate time series from Eq. 4.9 and an attribute $A_e \in \mathcal{A}$ for which the historical value is embedded:

$$\xi_{\text{historical}} : \mathcal{T}_{\mathcal{A}}^{(k)} \mapsto \mathcal{T}_{\mathcal{A}}^{(k-j)} \circ \left\{ \left(t_i, a_{i-j,x} \right) \in \mathbb{R} \times \mathbb{R} \;\middle|\; \left(t_{i-j}, a_{i-j} \right) \in \mathcal{T}_{A_e}^{(k)} \right\}_{i=1}^{k-j} \tag{4.11}$$

In many cases, embedding absolute historical values is not very feasible. Instead, just using the per-interval differences ΔX_i for a given time difference Δt, or, in special cases, also any other interval difference $\Delta_j X_i$, does yield far better results for an analysis:

$$\xi_{\text{difference}} : \mathcal{T}_{\mathcal{A}}^{(k)} \mapsto \mathcal{T}_{\mathcal{A}}^{(k-j)} \circ \left\{ \left(t_i, a_{i,x} - a_{i-j,x} \right) \in \mathbb{R} \times \mathbb{R} \;\middle|\; \left(t_i, a_{i,x} \right), \left(t_{i-j}, a_{i-j} \right) \in \mathcal{T}_{A_e}^{(k)} \right\}_{i=1}^{k-j} \tag{4.12}$$

It should be noted that, when embedding a new attribute via delay embedding, in the worst case, the resulting time series has a size of 0. Using time series with equidistant timestamps t_i and using a delay Δt that is a multiple of that Δt solves this problem. From the data used in this thesis, measurements were performed once a minute or second, respectively, so we can safely perform delay embedding, assuming the Δt is chosen as a multiple of 1s or 1min, respectively. It should also be noted that there is always a reduction in time series size when delay embedding. Consider a time series $\mathcal{T}_X^{(k)}$ of size k with equidistant timestamps of interval t_h. Then, when performing delay embedding with a delay $\Delta t = t_h$, we obtain a time series $\mathcal{T}_{X'}^{(k-1)}$ of size $k-1$, as we have to drop the lowest-valued timestamp according to the definitions from Eq. 4.11, Eq. 4.12. For time differences $\Delta t = j \cdot t_h, j \in \mathbb{N}$, we will subsequently lose j tuples.

While using absolute values as embedded attributes can predict a periodic function like the sinus, it cannot for example predict a function like

$$f(t) = t + \sin(t) \tag{4.13}$$

for new values of t. Using differences makes the attribute more meaningful in many ways. For my experiments, I nearly always used differences for delay embedding, with very good results.

Consider Fig. 4.1: It plots the gradation of the air temperature in Stuttgart Vaihingen from 12pm to 10pm on three consecutive days in the summer of 2011, and for one day exactly a

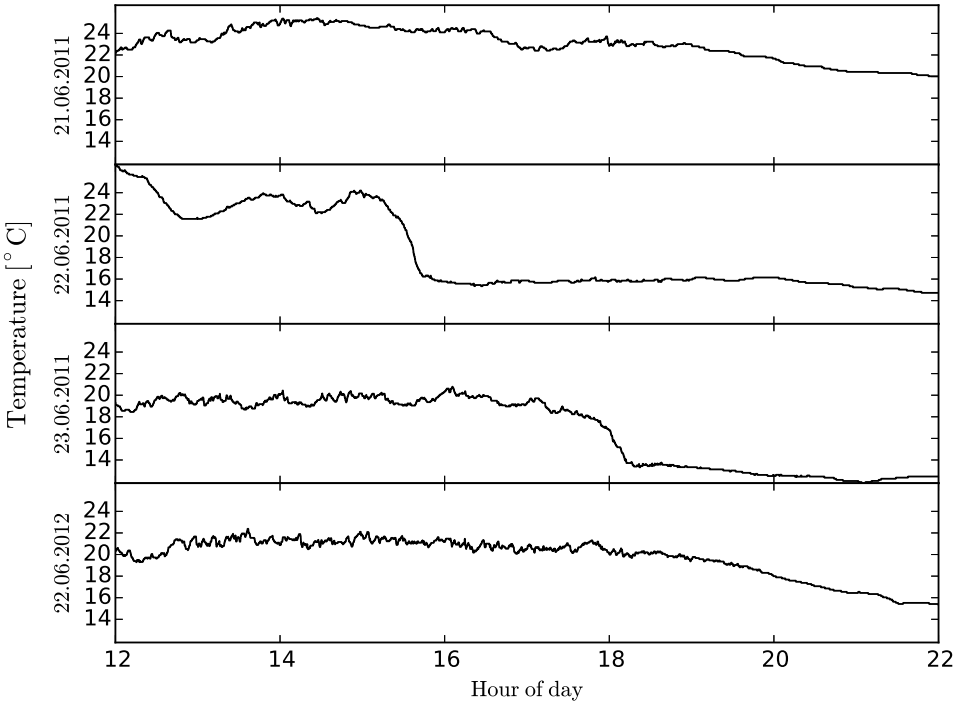


Figure 4.1: Air temperature gradation on three subsequent days and exactly a year later. On June 22nd, 2011, there was a big thunderstorm, during which the air temperature dropped by 6°C within less than an hour. By delay embedding of the temperature difference over 30min or 60min, events like this can be collated to thunderstorms very clearly.

year later. On June 22nd, around 3pm, there was a heavy thunderstorm, during which the temperature fell by 6°C within 30 minutes. By embedding the attribute[1]

$$\Delta T_{\text{air}} = T_{\text{air}}^{(t)} - T_{\text{air}}^{(t-1)} \tag{4.14}$$

with Δt = 30min, data points during a thunderstorm stand out clearly against data points without any rain. However, data points where rain was falling show a similar behavior, as seen from the data on June 23rd, 2011, where there was some rain around 6pm, resulting in a 4°C temperature drop.

Other suggestions for attribute generation are for example accumulation, for attributes that are already observing some accumulated value during the interval. For instance, in the

[1] To avoid confusion, it should be noted here that T denotes the physical quantity for temperature, which is an attribute, and not a time series, which is denoted using a calligraphic $\mathcal{T}$.

dataset I obtained from the Meteorological Institute of the University of Stuttgart, there was an attribute observing the amount of precipitation that had accumulated over the last minute, measured each minute. From this, I generated an attribute with $\Delta t = 5\text{min}$, which summed up the accumulated precipitation over all five measured values from those five minutes.

Another method is to use the discrete first derivative

$$A = \frac{X_i^{(t)} - X_i^{(t-1)}}{\Delta t} \tag{4.15}$$

as proposed by Garcke et al.[GGG13, 3–4], which is actually just a scaled version of time differences, assuming the time difference stays the same. When using different Δt, this becomes a handy method for generating attributes, however it has to be handled with care, as a discrete first derivative of $\frac{\mathrm{d}v}{\mathrm{d}t}$ over the interval $2\Delta t$ does not necessarily mean the same as the same discrete first derivative over the interval Δt.

4.3 Generation of Datasets

Generating datasets was done in several steps. From the preprocessed databases of the IPV, DWD and IWS data, a joined table was first generated with SQL `INNER JOIN` statements on the timestamp. As the IWS data was in a resolution of one minute, but the IPV data was in a resolution of one second, and because of the only partially overlapping time periods, this already drastically reduced the number of data points. By inner joining the IPV and IWS data, the number of data points was reduced to approximately 1.6 million.

From the join table, which only contained the desired attributes, with an SQL cursor object, the data was read row by row into memory using the `libpqxx` library for PostgreSQL with C++. As the time steps between each row were always the same, delay embedded attributes could easily be generated by taking differences of different rows. For example, for a $\Delta t = 1$ min, simply the previous row was subtracted from the current row to generate delay embedded attributes $\Delta_1 A_i$. For greater Δt, the equivalent number of rows were skipped in the subtraction.

The advantage of using a cursor on the raw database table was that the table would only need to be traversed once, which was a lot faster than querying all the data and loading all data into memory at once. The read data and the delay embedded attributes were then written back into another table using a `pqxx` tablewriter object.

In the last step, from the generated table, which contained all desired original and delay embedded attributes, the datasets could be generated with custom size using SQL queries which selected random rows from the table. For a list of attributes A_1, A_2, A_3, A_4 and an example dataset size of $N = 12\,345$, the SQL query would be

```
SELECT A_1, A_2, A_3, A_4
   FROM tabledelayembedded
   ORDER BY random()
   LIMIT 12345;
```

Listing 4.1: SQL query to generate dataset

Before being able to use the dataset, it had to be normalized to the $[0,1]^d$ hypercube. This was done by the most simple method possible, the linear scaling. For an attribute A of the dataset, its data points a_i were scaled using a function s:

$$s : \mathbb{R} \to [0,1], \qquad a_i \mapsto \frac{a_i - \min A}{\max A - \min A} \tag{4.16}$$

Of course, the scaling parameters, that is, $\max A, \min A$, had to be saved in order to be able to use the exact same parameters on the test dataset. For datasets with a large amount of data points near the edges of Ω_n, the scaling function was modified to include a margin, such that for example only the inner 80% were used in each dimension:

$$\tilde{s} : \mathbb{R} \to [0.1, 0.9], \qquad a_i \mapsto 0.8 \cdot s(a_i) + 0.1 \tag{4.17}$$

As a result of Hughes' phenomenon[Hug68], I chose a formula used by Good and Hardin [GH12] to gauge the necessary dataset size for dataset generation:

$$N = M^d \tag{4.18}$$

The formula states that, if only one independent variable exists and we perform some type of learning on it with a dataset of size M, we need approximately a dataset of size N for d independent variables in order to obtain a comparably good result. Therefore, when adding dimensions to attribute spaces, I also generated larger datasets whenever possible, keeping in mind that, for experiments, the dataset sizes had to be kept small in order to save some time.

5 Dataset Scenarios

5.1 Weather Events

5.1.1 The scenario

Utilizing weather warning data from the German metereological service (DWD)[KD15], I created a large dataset for classification. The dataset itself contained start and end points of weather warnings issued for Stuttgart and the type of warning. The warnings themselves were a bit specific, like **heavy snowfall and snow drift**, so I generalized the warnings into four categories that I chose to include. Those categories were `RAIN`, `SNOW`, `THUNDER`, and `FOG`. There was also a category `NOEVENT` which included all points in time where no other event was active. As the DWD dataset[KD15] only ranged from the end of 2009 to present, I could not use the whole photovoltaic database to generate the classification dataset, but instead only a subset of 121 841 918 data points. After also including the IWS dataset, which was only measured in one minute time steps, the number of data points was reduced to 1 649 910 data points.

First, relevant sensors for classification of weather events were selected. This was done by selecting those attributes which were obvious candidates for affecting the weather. Also, some not so obvious attributes were selected and then tested in the experiments. Last, time differences were delay embedded into the attribute space. Obvious choices for sensors were the wind speed and wind direction. Also, the air temperature and solar radiation were chosen, as they too indicate a lot about the current weather condition. To see whether it would be an interesting attribute, the direct current power generated by solar module 4 was also chosen. After first experiments which showed very large overlap between the 'snow' and 'no event' classes (see 5.1.2), I decided to also calculate the solar altitude at the current time for each data point and use it as an attribute. It was now possible to differentiate between nighttime and daytime data points. For calculating the sun's current altitude I used the formula found here[Gie15] with the coordinates for the University of Stuttgart in Vaihingen ($\phi = 48.7456724°, \lambda = 9.1024567°$):

$$d_{\text{year}} = (m-1) \cdot 30.3 + d \tag{5.1}$$

$$\delta = -23.45 \cdot \cos\left(2 \cdot \pi \cdot \frac{d_{\text{year}} + 10}{365}\right) \tag{5.2}$$

$$t = 60 \cdot \left(-0.171 \cdot \sin\left(0.0337 \cdot d_{\text{year}} + 0.465\right) - 0.1299 \cdot \sin\left(0.01787 \cdot d_{\text{year}} - 0.168\right)\right) \tag{5.3}$$

$$\theta_{\text{hour}} = 15 \cdot \left(h + \frac{min}{60} - \frac{15-\lambda}{15} - 12 + \frac{t}{60}\right) \tag{5.4}$$

$$x = \sin\left(\frac{\pi}{180°} \cdot \phi\right) \cdot \sin\left(\frac{\pi}{180°} \cdot \delta\right) + \cos\left(\frac{\pi}{180°} \cdot \phi\right) \cdot \cos\left(\frac{\pi}{180°} \cdot \delta\right) \cdot \cos\left(\frac{\pi}{180°} \cdot \theta_{\text{hour}}\right) \tag{5.5}$$

The solar altitude angle is then calculated as:

$$\theta_{\text{sun}} = \frac{\arcsin(x) \cdot 180°}{\pi} \tag{5.6}$$

Now, delay embedding was used on the sensor columns with a delay of 30 minutes and two time steps. From the resulting 12 columns, by experiment, the most relevant ones were selected for an ideal classification. In the following paragraphs, the results of various attribute subspaces are shown. In Tab. 5.2 in 5.1.3, the results are all compared and listed.

5.1.2 First experiments

Before starting the first real experiments, I tested what would happen when all specialized kinds of warning – like 'heavy snow', 'snow drift', see page 41 – were used, and concluded that using all those classes would deteriorate the resulting hit rates of the classification too much. Thus, I boiled the weather warnings down to their base classes. In the first experiments, the training and test datasets had a total of four classes:

RAIN The class for all data points where warnings for heavy rain were issued.

SNOW The class for all data points where warnings for any kind of snow were issued.

THUNDER The class for all data points where warnings for any kind of thunderstorm were issued.

FOG The class for all data points where warnings for fog were issued.

The experiments `exp_NO_NOEVENT_00` through `exp_NO_NOEVENT_12` were performed with arbitrarily chosen attributes. The experiments `exp_NO_NOEVENT_03`, `exp_NO_NOEVENT_04`, `exp_NO_NOEVENT_06`, `exp_NO_NOEVENT_07`, `exp_NO_NOEVENT_11`, and `exp_NO_NOEVENT_12` showed very good hit rates now, as shown in Tab. 5.2.

Table 5.1: Attributes for the weather event classification attribute space, with their unit and name.

Attribute	Unit	Attribute name	Source
$v_{\text{wind}}^{(t)}$	$\frac{\text{m}}{\text{s}}$	Wind speed	IPV
$d_{\text{wind}}^{(t)}$	°	Wind direction	IPV
$T_{\text{air}}^{(t)}$	°C	Air temperature	IPV
$E^{(t)}$	$\frac{\text{W}}{\text{m}^2}$	Solar radiation	IPV
$P_{\text{mod4}}^{(t)}$	W	DC power of module 4	IPV
$\Delta P_{\text{mod4}}^{(t)}$	W	DC power DC power of module 4 difference 1	Delay
$\Delta v_{\text{wind}}^{(t)}$	$\frac{\text{m}}{\text{s}}$	Wind speed difference 1	Delay
$\Delta E^{(t)}$	$\frac{\text{W}}{\text{m}^2}$	Solar radiation difference 1	Delay
$\Delta T_{\text{air}}^{(t)}$	°C	Air temperature difference 1	Delay
$\theta_{\text{sun}}^{(t)}$	°	Solar altitude	Calculated
$\phi_{\text{19m}}^{(t)}$	%	Relative humidity in 19m height	IWS
$p_{\text{air}}^{(t)}$	hPa	Air pressure	IWS
$\Delta p_{\text{air}}^{(t)}$	hPa	Air pressure difference 1	Delay
$\sum \text{rr}_{\text{rr07}}^{(t-1,t)}$	mm	Accumulated precipitation	IWS/Delay
$T_{\text{precip}}^{(t)}$	°C	Precipitation temperature	IWS

5.1.3 Second experiments

In the second experiments, I added data points from a disjoint class `NOEVENT`, which consisted of all data points which were left out of the other base classes. I first, as a test, checked what would happen when I used the whole attribute space, in `exp00`. This resulted in a hit rate of roughly 40%, which is not that bad considering that the dataset had 5 classes. In `exp01` and `exp02`, I tried out attribute spaces that I thought would yield better results. However, the hit rates did not really improve. In `exp03` through `exp16`, I removed one different attribute from the attribute space in each experiment to see how it would affect the hit rate. After the experiments were all done, I checked the amount of data points which were stored in the **confusion matrix**[WFH11] for each classification. The confusion matrix M_c is defined as

$$M_c = \left(m_{i,j}\right) := \text{Amount of data points in class } i \text{ which were classified to class } j \tag{5.7}$$

So, the diagonal elements of the matrix would count the hits, while the non-diagonal elements would count the wrongly predicted points (misses) and even specify the per-class confusion. The non-diagonal elements in the **column** of a class are **false positives** for that class, the non-diagonal elements in the **row** of a class are **false negatives** for that class. In Tab. 5.4, the confusion matrix of a classification experiment – `exp_3_11_1` – is shown as an example.

From the confusion matrix, I concluded that the `NOEVENT` class was too vague and thus confusing the classification too much. This made a lot of sense: The DWD would only issue

Table 5.2: Classification hit rates for different attribute space subsets for the weather event scenario for Δt = 30min.

Name	Size	λ	Wind speed	Wind direction	Air temperature	Solar radiation	DC power of module 4	DC power of module 4 difference 1	Wind speed difference 1	Solar radiation difference 1	Air temperature difference 1	Solar altitude	Relative humidity in 19m height	Air pressure	Air pressure difference 1	Accumulated precipitation	Precipitation temperature	Hit rate
With class NOEVENT																		
exp00	50 000	1E-6	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	40.68
exp01	50 000	1E-6	●	●	●	●	●					●	●	●		●	●	41.44
exp02	50 000	1E-6	●	●	●	●	●	●	●	●	●	●	●	●	●	●		43.36
exp03	50 000	1E-5	●		●	●	●	●	●	●	●	●	●	●	●	●	●	41.74
exp04	50 000	1E-6	●	●		●	●	●	●	●	●	●	●	●	●	●	●	42.24
exp05	50 000	1E-6	●	●	●		●	●	●	●	●	●	●	●	●	●	●	43.68
exp06	50 000	1E-6	●	●	●	●		●	●	●	●	●	●	●	●	●	●	47.56
exp07	50 000	1E-6	●	●	●	●	●		●	●	●	●	●	●	●	●	●	41.84
exp08	50 000	1E-6	●	●	●	●	●	●		●	●	●	●	●	●	●	●	42.66
exp09	50 000	1E-6	●	●	●	●	●	●	●		●	●	●	●	●	●	●	41.44
exp10	50 000	1E-6	●	●	●	●	●	●	●	●		●	●	●	●	●	●	41.26
exp11	50 000	1E-6	●	●	●	●	●	●	●	●	●		●	●	●	●	●	41.44
exp12	50 000	1E-5	●	●	●	●	●	●	●	●	●	●		●	●	●	●	41.64
exp13	50 000	1E-6	●	●	●	●	●	●	●	●	●	●	●		●	●	●	40.54
exp14	50 000	1E-5	●	●	●	●	●	●	●	●	●	●	●	●		●	●	41.22
exp15	50 000	1E-6	●	●	●	●	●	●	●	●	●	●	●	●	●		●	43.20
exp16	50 000	1E-5	●	●	●	●	●	●	●	●	●	●	●	●	●	●		48.64
Without class NOEVENT																		
exp_NO_NOEVENT_00	50 000	1E-6	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	43.34
exp_NO_NOEVENT_01	50 000	1E-6	●	●	●	●	●		●			●	●	●		●	●	44.70
exp_NO_NOEVENT_02	50 000	1E-5	●			●	●					●	●	●			●	44.84
exp_NO_NOEVENT_03	50 000	1E-5	●			●	●		●			●	●	●		●		69.54
exp_NO_NOEVENT_04	50 000	1E-5	●	●	●	●	●					●	●	●				70.74
exp_NO_NOEVENT_05	50 000	1E-6	●	●	●	●	●		●			●	●	●	●		●	45.26
exp_NO_NOEVENT_06	50 000	1E-4	●			●	●					●	●	●	●	●		65.70
exp_NO_NOEVENT_07	50 000	1E-5	●	●		●	●		●		●	●	●	●	●			69.04
exp_NO_NOEVENT_08	50 000	1E-5	●	●	●	●	●				●	●	●	●	●	●	●	43.98
exp_NO_NOEVENT_09	50 000	1E-5	●	●	●	●	●		●		●	●	●	●	●	●	●	42.86
exp_NO_NOEVENT_10	50 000	1E-5	●	●		●	●					●	●	●			●	44.24
exp_NO_NOEVENT_11	50 000	1E-6	●	●		●	●		●		●	●	●	●		●		68.74
exp_NO_NOEVENT_12	50 000	1E-5	●	●	●	●	●				●	●	●	●				68.74

More experiments on highlighted columns in Tab. 5.3.

warnings for very heavy rain, but none for everyday rainy weather. Thus, most of the times it rained, there would be no weather warnings, and thus, the class of those events would still be NOEVENT. Also, like already described in 3.2.2, the snow and fog classes were not very accurate because of the geographical differences between the location for which the weather warnings were issued and the location where the weather data were measured (see Fig. 3.1).

5.1.4 Third experiments

Based on the high hit rate experiments without use of the class NOEVENT – cf. 5.1.2 – another set of experiments was generated. The choice of attributes which was causing these high hit rates were not clearly deductible from the experiments, so I tried out every possible

Table 5.3: Classification hit rates for different attribute space subsets determined off the blue highlighted rows in Tab. 5.2 for the weather event scenario for Δt = 30min.

Name	Size	λ	Wind speed	Wind direction	Air temperature	Solar radiation	DC power of module 4	[unused]	Wind speed difference 1	[unused]	Air temperature difference 1	Solar altitude	Relative humidity in 19m height	Air pressure	Air pressure difference 1	Accumulated precipitation	[unused]	Hit rate
exp_3_00	50 000	1E-4	●	●	●	●	●		●		●	●	●	●	●	●		66.26 %
exp_3_01	50 000	1E-4	●		●	●	●		●		●	●	●	●	●	●		67.18 %
exp_3_02	50 000	1E-5	●	●		●	●		●		●	●	●	●	●	●		68.26 %
exp_3_03	50 000	1E-5	●			●	●		●		●	●	●	●	●	●		69.24 %
exp_3_04	50 000	1E-4	●	●	●	●	●				●	●	●	●	●	●		66.92 %
exp_3_05	50 000	1E-5	●		●	●	●				●	●	●	●	●	●		67.96 %
exp_3_06	50 000	1E-4	●	●		●	●				●	●	●	●	●	●		66.30 %
exp_3_07	50 000	1E-5	●			●	●				●	●	●	●	●	●		68.22 %
exp_3_08	50 000	1E-5	●	●	●	●	●		●			●	●	●	●	●		68.00 %
exp_3_09	50 000	1E-4	●		●	●	●		●			●	●	●	●	●		65.34 %
exp_3_10	50 000	1E-4	●	●		●	●		●			●	●	●	●	●		65.62 %
exp_3_11	50 000	1E-6	●			●	●		●			●	●	●	●	●		71.62 %
exp_3_12	50 000	1E-5	●	●	●	●	●					●	●	●	●	●		69.44 %
exp_3_13	50 000	1E-5	●		●	●	●					●	●	●	●	●		69.28 %
exp_3_14	50 000	1E-5	●	●		●	●					●	●	●	●	●		68.24 %
exp_3_15	50 000	1E-5	●			●	●					●	●	●	●	●		71.48 %
exp_3_16	50 000	1E-5	●	●	●	●	●		●		●	●	●	●		●		68.14 %
exp_3_17	50 000	1E-5	●		●	●	●		●		●	●	●	●		●		69.76 %
exp_3_18	50 000	1E-5	●	●		●	●		●		●	●	●	●		●		68.86 %
exp_3_19	50 000	1E-5	●			●	●		●		●	●	●	●		●		69.40 %
exp_3_20	50 000	1E-4	●	●	●	●	●				●	●	●	●		●		67.48 %
exp_3_21	50 000	1E-4	●		●	●	●				●	●	●	●		●		67.98 %
exp_3_22	50 000	1E-4	●	●		●	●				●	●	●	●		●		66.36 %
exp_3_23	50 000	1E-4	●			●	●				●	●	●	●		●		68.62 %
exp_3_24	50 000	1E-5	●	●	●	●	●		●			●	●	●		●		67.34 %
exp_3_25	50 000	1E-4	●		●	●	●		●			●	●	●		●		66.64 %
exp_3_26	50 000	1E-5	●	●		●	●		●			●	●	●		●		69.14 %
exp_3_27	50 000	1E-4	●			●	●		●			●	●	●		●		67.40 %
exp_3_28	50 000	1E-6	●	●	●	●	●					●	●	●		●		69.12 %
exp_3_29	50 000	1E-4	●		●	●	●					●	●	●		●		66.86 %
exp_3_30	50 000	1E-4	●	●		●	●					●	●	●		●		66.94 %
exp_3_31	50 000	1E-4	●			●	●					●	●	●		●		68.00 %
exp_3_32	50 000	1E-4	●	●	●	●	●		●		●	●	●	●	●			67.06 %
exp_3_33	50 000	1E-4	●		●	●	●		●		●	●	●	●	●			67.48 %
exp_3_34	50 000	1E-4	●	●		●	●		●		●	●	●	●	●			65.80 %
exp_3_35	50 000	1E-4	●			●	●		●		●	●	●	●	●			67.20 %
exp_3_36	50 000	1E-5	●	●	●	●	●				●	●	●	●	●			68.32 %
exp_3_37	50 000	1E-4	●		●	●	●				●	●	●	●	●			68.80 %
exp_3_38	50 000	1E-4	●	●		●	●				●	●	●	●	●			67.56 %
exp_3_39	50 000	1E-4	●			●	●				●	●	●	●	●			69.00 %
exp_3_40	50 000	1E-4	●	●	●	●	●		●			●	●	●	●			66.74 %
exp_3_41	50 000	1E-5	●		●	●	●		●			●	●	●	●			69.96 %
exp_3_42	50 000	1E-4	●	●		●	●		●			●	●	●	●			65.72 %
exp_3_43	50 000	1E-4	●			●	●		●			●	●	●	●			69.34 %
exp_3_44	50 000	1E-5	●	●	●	●	●					●	●	●	●			69.94 %
exp_3_45	50 000	1E-4	●		●	●	●					●	●	●	●			68.26 %
exp_3_46	50 000	1E-4	●	●		●	●					●	●	●	●			67.10 %
exp_3_47	50 000	1E-4	●			●	●					●	●	●	●			68.26 %
exp_3_48	50 000	1E-4	●	●	●	●	●		●		●	●	●	●				66.88 %
exp_3_49	50 000	1E-4	●		●	●	●		●		●	●	●	●				68.42 %
exp_3_50	50 000	1E-5	●	●		●	●		●		●	●	●	●				69.12 %
exp_3_51	50 000	1E-4	●			●	●		●		●	●	●	●				67.18 %
exp_3_52	50 000	1E-4	●	●	●	●	●				●	●	●	●				67.62 %
exp_3_53	50 000	1E-4	●		●	●	●				●	●	●	●				67.56 %
exp_3_54	50 000	1E-6	●	●		●	●				●	●	●	●				69.96 %
exp_3_55	50 000	1E-4	●			●	●				●	●	●	●				68.04 %
exp_3_56	50 000	1E-4	●	●	●	●	●		●			●	●	●				66.68 %
exp_3_57	50 000	1E-4	●		●	●	●		●			●	●	●				66.52 %
exp_3_58	50 000	1E-5	●	●		●	●		●			●	●	●				68.12 %
exp_3_59	50 000	1E-4	●			●	●		●			●	●	●				67.94 %
exp_3_60	50 000	1E-5	●	●	●	●	●					●	●	●				70.02 %
exp_3_61	50 000	1E-4	●		●	●	●					●	●	●				68.32 %
exp_3_62	50 000	1E-5	●	●		●	●					●	●	●				69.96 %
exp_3_63	50 000	1E-6	●			●	●					●	●	●				70.06 %
exp_3_11_1	200 000	1E-6	●			●	●		●			●	●	●	●	●		40.79 %

combination of the attributes which were different in the experiments with high hit rates, but using all attributes that they had in common. This resulted in the experiments listed in Tab. 5.3. Here, most attribute combinations resulted in hit rates between 60% and 70%. Four experiments, however, resulted in hit rates over 70%. Those are highlighted in Tab. 5.3. What is noticeable is that while exp_3_11 has the highest dimensionality, it also has the highest hit rate. Based on the formula (Eq. 4.18) provided by Good and Hardin[GH12], this implies that the attributes in this experiment are all relevant for the classification.

Using the results from the large experimentation batch, I did another experiment – exp_3_11_1 – on the attribute set from exp_3_11, but with a training dataset size of 200 000. However, this time I generated a dataset where I made sure that all classes had the same amount of data points in the dataset. The resulting hit rate showed that the previous hit rates were largely false positives, as the hit rate for exp_3_11_1 only was around 40%. However, it has to be considered that exp_3_11_1 had four classes to choose from. That means that a random function would only have classified 25% of the data points into the right class.

Table 5.4: Confusion matrix for exp_3_11_1. It can be seen that the THUNDER class has the best individual hit rate.

		Prediction			
		RAIN	SNOW	THUNDER	FOG
Reference	RAIN	2 349	49	4 440	3 162
	SNOW	124	2 513	6 047	1 316
	THUNDER	195	96	7 946	1 763
	FOG	96	219	6 179	3 506

In Tab. 5.4, the confusion matrix for exp_3_11_1 is shown. This matrix tells us exactly where the problems lie in a classification. For one, rain and snow events are often falsely classified as thunderstorm or fog. Also, fog is often falsely classified as thunderstorm. However, rain and snow are well confined against each other, meaning that they are seldom classified as each other. Also, thunderstorms are nearly always classified right, but also non-thunderstorm events are classified as thunderstorm.

That FOG events were badly classified was to be expected from the innate inaccuracy of the weather warning data, as discussed in 3.2.2. That the RAIN and SNOW classes are so often classified as THUNDER or FOG is probably caused by the thunderstorm class being very dominant. As seen from the confusion matrix, the amount of false positives is huge in that class. The exact cause for this cannot easily be determined as the dataset for exp_3_11_1 is 9-dimensional, meaning it cannot be easily visualized. All that can be done is to do further preprocessing on the dataset, removing all data points which are prone to being false positives, and thus creating a dataset with more clearly confined classes.

One method of doing this would be to compute the probability of a point being in its reference class for each point in the dataset and storing those values. This way, individual datasets

could then be generated with a freely chooseable confusion parameter. Doing this, however, would go beyond the scope of this thesis.

5.2 Power Prediction

5.2.1 About the scenario

The goal of this scenario was to be able to predict the power generation of a photovoltaic module a fixed time into the future based on measured or calculated values only from the present or past. So, assuming we have a set of k attributes

$$\mathcal{A} = \{A_i\}_{i=1}^{k} \tag{5.8}$$

and a index $\mathcal{I}_t$ such that

$$\mathcal{T}_{\mathcal{A}}^{(k)} = \mathcal{I}_t \parallel \mathcal{A} \tag{5.9}$$

where we have an attribute $A_P \in \mathcal{A}$ for the power. Then we would learn a regression from a time series, which would serve as dataset

$$\mathcal{T} = \left\{ \left(t_j, \vec{\omega}_j \circ p_k \right) \;\middle|\; \left(t_j, \vec{\omega}_j \right) \in \mathcal{T}_{\mathcal{A}}^{(k)} \wedge \left(t_k, p_k \right) \in \mathcal{I}_t \parallel A_p \wedge t_k = t_j + \tau \right\} \tag{5.10}$$

from the attribute space and the time series of power measured at the time $t + \tau$, that is, one time step into the future. The *lead time*[BJ76] τ was in all experiments set to the same value as the delay embedding interval Δt. The non-normalized attribute space Ω would then be chosen as a subset of the attribute space of the time series of measured attributes and all possible delay embedded attributes:

$$\mathcal{T}_{\Omega^*}^{(k)} = \bigcirc_{\xi \in \Xi} \xi \left(\mathcal{T} \right) \tag{5.11}$$

$$\Omega \subsetneqq \Omega^* \tag{5.12}$$

As the space of delay embeddable attributes is infinitely large, we only use a true subset of it. In fact, in most cases this subset is even smaller or equal in size to the number of original attributes, as stated in Taken[Tak81], Sauer et al.[SYC91], as we don't want to deteriorate the precision of our regression.

The goal is to obtain a regression function

$$u(x) : \Omega_n \to \mathbb{R},\ \vec{\omega}^{(t)} \overset{\approx}{\mapsto} P^{(t+1)} \tag{5.13}$$

which approximates the power generated one time step Δt into the future based only on already known attributes. This way, theoretically, we could approximate the power generation live if we had a data feed from all sensors available. Instead, we will just collect random data points from the database and compare the predicted power generation $u(\omega_j)$ with the measured value $P^{(t+1)}$.

5.2.2 Attribute Space

The attribute space consisted of a number of data dimensions from both the IPV database and the weather station of the IWS. Additional attributes were added by delay embedding of already present attributes. The full list of selected attributes can be seen in Tab. 5.5.

For the full attribute space, I selected the more obvious attributes. I selected the module temperature as according to Luque, Hegedus[LH03] and Nelson[Nel03], the current of a photovoltaic cell can be calculated as

$$I = I_{\mathrm{L}} - I_{\mathrm{D}} - I_{\mathrm{SH}} \tag{5.14}$$

where I_{L} is the current produced by the photovoltaic cell, I_{D} is the current flowing through the diode of the cell and I_{SH} is the current which flows through the **Shunt resistor**. Those two currents are lost, and the formula for the diode current is

$$I_{\mathrm{D}} = I_0 \cdot \left(\exp\left(\frac{qV_j}{nkT} \right) \right) \tag{5.15}$$

where I_0 is the reverse saturation current of the diode, which is constant. q is the elementary charge, a constant. V_j is the voltage across the diode and Shunt resistor. n is the diode ideality factor, a constant for the diode. k is the Boltzmann constant, and T is the module temperature. Thus, the module temperature in fact has an effect on the module's generated power.

The second attribute I chose was the module power itself. I then chose the air temperature and the solar radiation.

From the IWS database, I chose the relative humidity of the air. There were two attributes for this, one measured two meters above the ground, one measured 19 meters above the ground. As the data were not measured in the same place as the module is located, and the module is installed on a roof anyways, I selected the 19 meters above ground data.

I also calculated the height of the sun from Eq. 5.6 and inserted it as an additional attribute. The reasoning here was that the height of the sun differentiates low power because of low sun height from low power because of covered sky, which might be useful for the prediction.

I then embedded some attributes by generating the differences of most already present attributes – except for the sun height – over the last time frame for a time difference of $\Delta t = 5\,\mathrm{min}$. The *lead time*[BJ76, 2] – the Δt used to embed the attribute $P^{(t+1)}$ – was also chosen as 5 min. I also calculated the accumulated precipitation over the time frame from the per-minute precipitation data from the IWS database.

Now I started to experiment with different subsets of this attribute space to gauge which ones were relevant and helpful for the prediction and which ones were not.

Table 5.5: Attribute space for the power prediction dataset. For individual experiments, a subset of the attribute space was used as experiment attribute space. The image set of all regression functions was the module power in the coming time step $P_{\text{mod}}^{(t+1)}$.

Attribute	Unit	Attribute name	Source
$T_{\text{mod}}^{(t)}$	°C	Module temperature at time t	IPV
$\Delta T_{\text{mod}}^{(t)}$	°C	Module temperature change in time frame $[t-1,t]$	Delay
$P_{\text{mod}}^{(t)}$	W	Module power at time t	IPV
$\Delta P_{\text{mod}}^{(t)}$	W	Module power change in time frame $[t-1,t]$	Delay
$T_{\text{air}}^{(t)}$	°C	Air temperature at time t	IPV
$\Delta T_{\text{air}}^{(t)}$	°C	Air temperature change in time frame $[t-1,t]$	Delay
$E^{(t)}$	$\frac{\text{W}}{\text{m}^2}$	Solar radiation at time t	IPV
$\Delta E^{(t)}$	$\frac{\text{W}}{\text{m}^2}$	Solar radiation change in time frame $[t-1,t]$	Delay
$\phi_{19\text{m}}^{(t)}$	%	Relative humidity of air at 19m height at time t	IWS
$\Delta\phi_{19\text{m}}^{(t)}$	%	Relative humidity of air at 19m height change in time frame $[t-1,t]$	Delay
$\text{rr}_{07}^{(t-1,t)}$	$\frac{\text{dm}^3}{\text{m}^2}$	Accumulated precipitation over the time frame $[t-1,t]$	IWS/Delay
$\theta_{\text{sun}}^{(t)}$	°	Sun height at time t	Calculated

5.2.3 Experiments

For all the experiments, I let the regularization parameter λ be determined by k-fold cross-validation. I then learned the regression on the selected attributes. I generated a reference dataset

$$\mathcal{R} = \left\{ \left(\omega_j, P_j \right) \in \Omega_n \times \mathbb{R} \right\}_{j=1}^{N} \tag{5.16}$$

with

$$\mathcal{R}_{\Omega_n} = \left\{ \omega_j \in \Omega_n \right\}_{j=1}^{N} \tag{5.17}$$

and

$$\mathcal{R}_P = \left\{ P_j \in \mathbb{R} \right\}_{j=1}^{N} \tag{5.18}$$

and predicted the power of the test points $\tilde{P}$ via the regression function $u \in V_n^{(1)}$ as

$$\tilde{P} = u\left(\mathcal{R}_{\Omega_n} \right) \tag{5.19}$$

Then, I compare the prediction $\tilde{P}$ to the reference $\mathcal{R}_P$ by calculating the mean squared error, the normalized mean squared error, the mean error and the relative mean error. The experiments can be seen in Tab. 5.6, with the respectively chosen attribute space.

Table 5.6: Power prediction scenario experiments. For each subset of the attribute space that was tested, the mean squared error and the normalized mean squared error are shown. Also, the mean error and relative mean error were calculated. The time difference chosen was Δt = 5min.

	λ	Training dataset size	Module temperature at time t	Module temperature change in time frame $[t-1,t]$	Module power at time t	Module power change in time frame $[t-1,t]$	Air temperature at time t	Air temperature change in time frame $[t-1,t]$	Solar radiation at time t	Solar radiation change in time frame $[t-1,t]$	Relative humidity of air at 19m height at time t	Relative humidity of air at 19m height change in time frame $[t-1,t]$	Accumulated precipitation over the time frame $[t-1,t]$	Sun height at time t	MSE	NMSE	ME	rME
exp0	1E-3	50 000	●	●	●	●	●	●	●	●	●	●	●	●	119 867.07	0.9977	136.167	0.0740
exp1	1E-3	50 000	●		●				●	●	●			●	12 450.68	0.1030	34.798	0.0202
exp2	1E-4	50 000	●		●		●		●	●	●			●	13 099.24	0.1079	35.147	0.0192
exp3	1E-5	50 000	●		●		●		●	●	●	●		●	13 997.94	0.1215	37.584	0.0204
exp4	1E-4	50 000	●		●	●	●		●	●	●	●		●	12 749.79	0.1033	34.188	0.0199
exp5	1E-4	50 000	●		●	●			●	●	●	●		●	13 967.55	0.1103	35.454	0.0210
exp6	1E-3	50 000	●		●	●	●		●	●	●			●	16 265.94	0.1245	38.463	0.0221
exp7	1E-4	50 000	●		●	●			●	●	●			●	14 839.04	0.1115	37.489	0.0202
exp8	1E-5	50 000	●		●				●	●	●		●	●	114 418.94	0.9955	132.592	0.0758
exp9	1E-1	50 000	●		●		●		●	●	●		●	●	112 295.55	0.9999	132.592	0.0751
exp10	1E-5	1 612 754	●	●	●	●	●	●	●	●	●	●	●	●	125 046.73	0.9982	142.217	0.0778
exp2_0	1E-4	50 000			●		●		●	●	●			●	14 838.23	0.1254	36.124	0.0197
exp2_1	1E-4	50 000	●				●		●	●	●			●	37 880.85	0.2907	85.856	0.0499
exp2_2	1E-4	50 000	●		●				●	●	●			●	11 263.67	0.0934	33.689	0.0191
exp2_3	1E-4	50 000	●		●		●		●		●			●	13 632.42	0.1074	35.888	0.0212
exp2_4	1E-4	50 000	●		●		●		●	●				●	11 879.82	0.0954	32.852	0.0197
exp2_5	1E-4	50 000	●		●		●		●	●	●				12 812.85	0.1035	33.923	0.0190
exp2_2_0	1E-4	50 000			●				●	●	●			●	12 026.75	0.0961	33.218	0.0194
exp2_2_1	1E-3	50 000	●						●	●	●			●	41 083.83	0.3178	90.776	0.0493
exp2_2_2	1E-4	50 000	●		●					●	●			●	11 415.73	0.0954	33.337	0.0198
exp2_2_3	1E-4	50 000	●		●				●		●			●	11 351.45	0.0976	33.125	0.0195
exp2_2_4	1E-4	50 000	●		●				●	●				●	13 764.72	0.1165	34.232	0.0186
exp2_2_5	1E-3	50 000	●		●				●	●	●				13 066.45	0.1030	34.565	0.0195
exp2_2_v1	1E-4	1 200 000	●		●				●	●	●			●	11 388.98	0.0911	32.379	0.0170
exp2_2_v2	1E-4	1 200 000	●		●				●	●	●			●	12 072.22	0.0963	33.245	0.0180
exp2_2_v3	1E-4	1 200 000	●		●				●	●	●			●	11 753.13	0.0947	32.879	0.0178
exp2_2_v4	1E-4	1 200 000	●		●				●	●	●			●	11 417.68	0.0935	32.287	0.0174
exp2_2_v5	1E-4	1 200 000	●		●				●	●	●			●	11 736.47	0.0945	32.749	0.0177
exp2_2_v6	1E-4	1 200 000	●		●				●	●	●			●	11 685.45	0.0942	32.475	0.0174

First, in exp0, I checked how the prediction would fare on the full attribute space. It did not work very well. However, this might have be an instance of Hughes' phenomenon[Hug68], as the dataset size was selected to 50 000. So, according to the heuristics provided by Good and Hardin[GH12], I re-ran the experiment with a larger sample size as exp10. That, too, didn't work out very well, so I concluded that the attribute space contained some attributes that lessened the precision of the prediction and thus should not be part of the attribute space.

In experiments exp1 through exp9, I tested some arbitrarily chosen subsets. I then selected the experiment with the lowest mean squared error and relative mean squared error and refined it. This was done in exp2_0 through exp2_5 by removing one attribute from exp2, and removing a different one in each experiment. However, I figured the solar radiation itself was too important for the power to be removed, so I didn't test it.

The experiments exp2_0 through exp2_5 revealed some interesting correlations. For one, and little surprising, removing the current power attribute $P_{\text{mod}}^{(t)}$ drastically worsened the prediction. Removing other attributes did not lead to significant improvement of the result, except for a slight improvement when removing the air temperature attribute, which, to be honest, is coherent, as the air temperature should not have that large an influence on the power. Thus, exp2_2 was selected for further refinement.

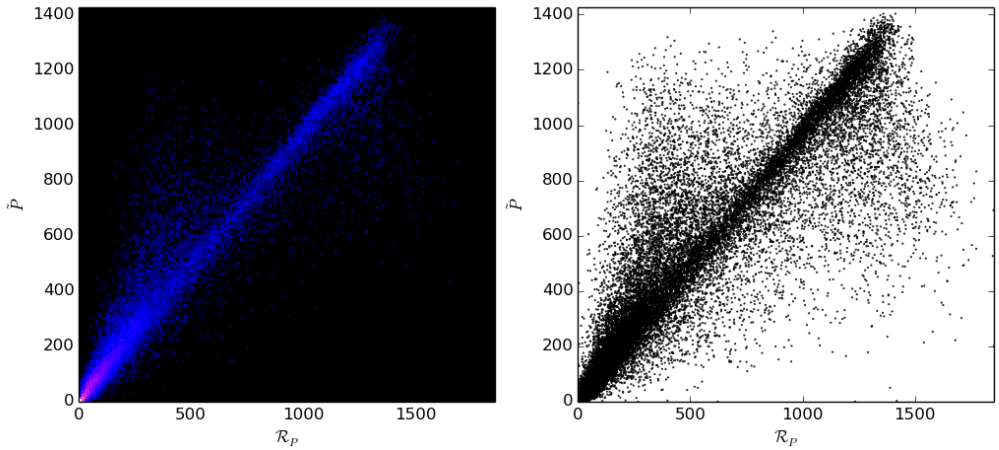


Figure 5.1: Result of the regression for exp2_2_v3. The predicted power $\tilde{P}^{(t+1)}$ is plotted against the reference power $\mathcal{R}_{P(t+1)}$ taken from the reference dataset. On the left side, the density of the data points is estimated by using a two-dimensional histogram. This way, the amount of points on the diagonal can be grasped. On the right side, the $(\mathcal{R}_P, \tilde{P})$ pairs are plotted in a scatter plot, to show the outliers. It can be seen that the number of points near the origin is very high. This is understandable as the dataset used was distributed evenly for values across all hours, so approximately half of the data points are at night.

Again, for exp2_2, every attribute was removed once and the prediction was tested, resulting in the experiments exp2_2_0 through exp2_2_5. Here, we did not get considerably better results. Therefore, I reran those experiments with a much higher training set size of 1 200 000 in the experiments exp2_2_v1 through exp2_2_v6. Those also yielded slightly better results. Getting NMSEs < 0.1 was in itself a very good result. This meant we had a pretty good prediction of the solar power in the next time step in most cases.

In Fig. 5.1, the prediction $\tilde{P}$ of each data point is plotted over the expected value $\mathcal{R}_P$, as defined in Eq. 5.18, Eq. 5.19. As can be seen, especially from the heat map, most of the points lie near the diagonal, which means a good prediction of the power. Of course, there are outliers where the prediction did not match the reference.

Outliers can be explained by one of multiple causes: Either they are points in sparsely populated areas of the attribute space, where the regression function could not be learned as precisely because of missing data. Or there were contradictory data points in the training dataset, resulting in the regression function trying to fit all of them as closely as possible and thus accumulating some error for every data point. As the power generated by a photovoltaic module can be deterministically calculated with perfect information, contradictory points in the dataset might mean that there is a relevant attribute that influences the power generation and which is not included in the attribute space.

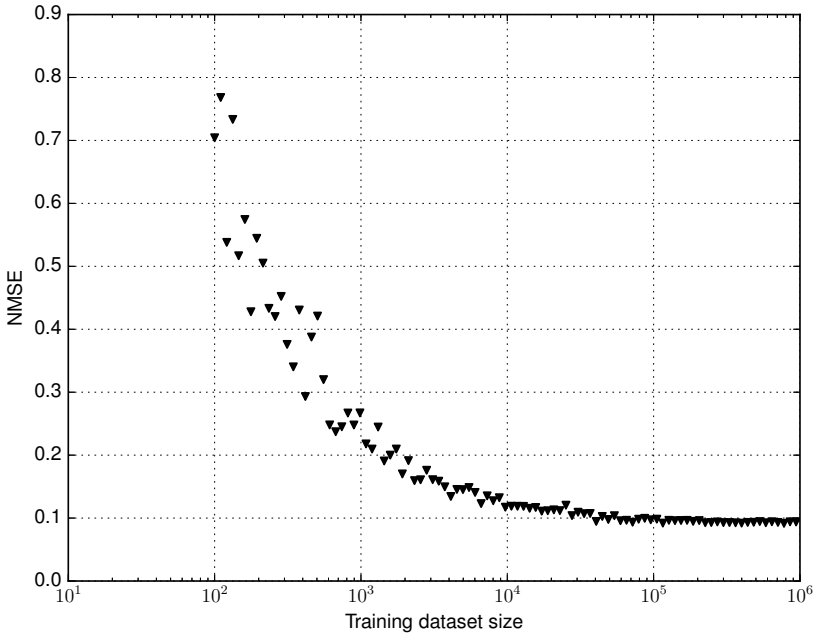


Figure 5.2: NMSE plotted over the size of the training dataset. The test dataset was always the same.

To illustrate the correlation between dataset size and accuracy, I generated datasets of variable size for the experiment `exp2_2_0` and learned a regression from them. I also generated one fix test dataset, with which I tested every learned regression and got the NMSE from the test. The results of those experiments can be viewed in Fig. 5.2, where the NMSE is plotted against the dataset size. It can be seen how the prediction gets more accurate with dataset size. For dataset sizes < 10000, this is simply caused by undersampling of the data, which leads to a largely unpopulated attribute space. Therefore, the regression simply cannot be accurate, as it has no information whatsoever on large areas of the attribute space. It should also be noted how the accuracy of the prediction converges around a dataset size of 100 000. Here, the vagueness of the prediction is the only error left, as, of course, the fact that we chose a time difference of $\Delta t = 5\,\text{min}$ means that we lose some precision on the steadiness of our attributes during that time. Within five minutes, a lot can change. For instance, it is enough time for a cloud to move in front of the sun. For a lower time difference, we should get even better results, and for higher time differences, the power at $t+1$ becomes more and more nondeterministic.

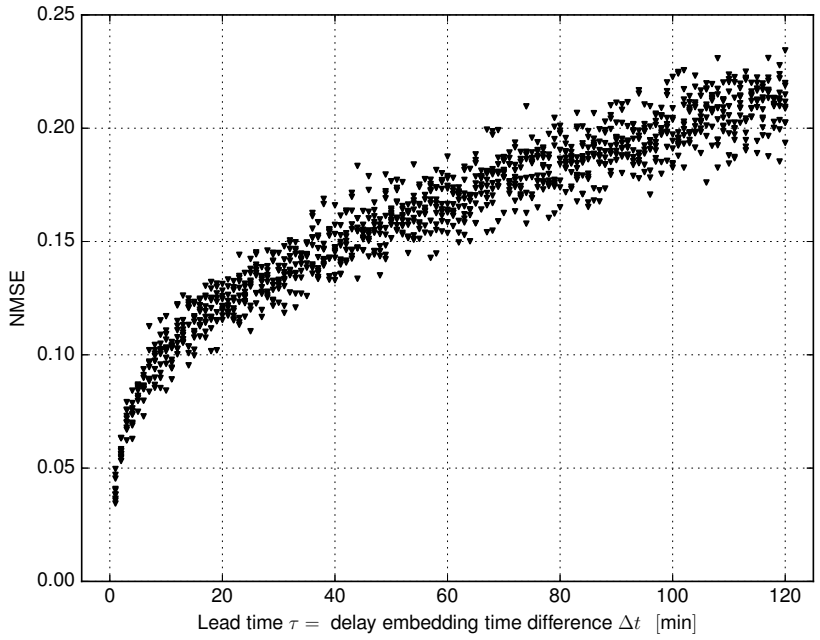
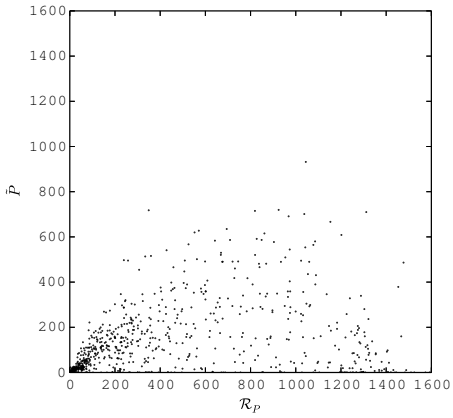


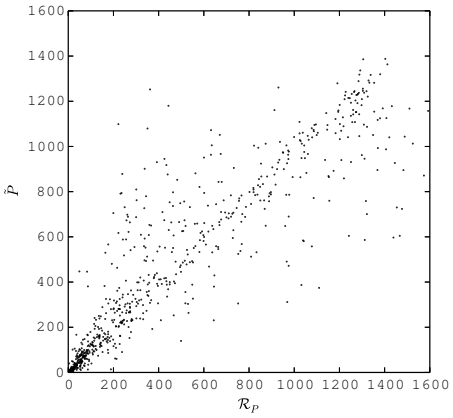
Figure 5.3: NMSE plotted over the time difference chosen for the prediction. The increasing inaccuracy with increasing Δt can be seen very clearly. The dataset size was chosen as 50 000, with a regularization parameter $\lambda = 10^{-4}$.

To illustrate the correlation between time difference for the prediction and accuracy, I generated datasets for variable time differences $1\,\text{min} \leq \Delta t \leq 120\,\text{min}$. The trivial $\Delta t = 0$ was omitted as the error for that time difference would be given by the error of the regression function's

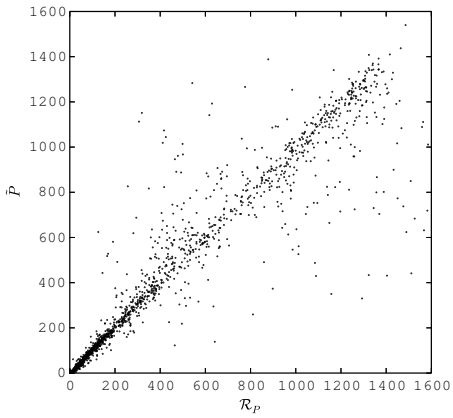
interpolation u in the training data points, and had no predictive significance. The resulting NMSEs can be seen in Fig. 5.3. For a time difference of only one minute, the prediction was very accurate, resulting in an NMSE of ≈ 0.05. However, such a prediction is most probably not much better than assuming that the power stays the same as the previous minute. For larger Δt, the predictions get less accurate fast, so that the NMSE is around 0.1 already for $\Delta t = 8\,\text{min}$. For a $\Delta t = 2\,\text{h}$, the NMSE is already around 0.2. This is in line with how hard predictions get with increasing time difference.



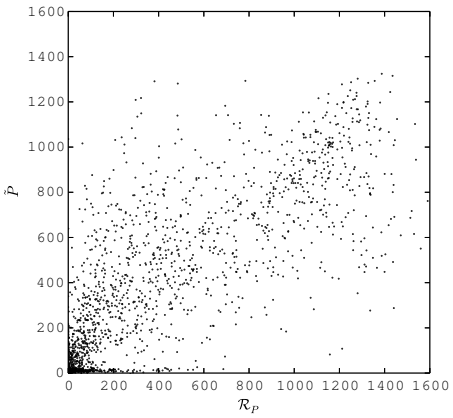
(a) $N = 100$ and $\Delta t = 5\,\text{min}$: Undersampling



(b) $N = 10^6$ and $\Delta t = 5\,\text{min}$: Sufficient sample size



(c) $N = 50\,000$ and $\Delta t = 2\,\text{min}$: Good prediction



(d) $N = 50\,000$ and $\Delta t = 119\,\text{min}$: Vague prediction

Figure 5.4: Comparison of prediction $\tilde{P}$ and reference $\mathcal{R}_{P(t+1)}$ for different training dataset sizes N and different time differences Δt.

In Fig. 5.4c, the predicted power at $t+1$ – $\tilde{P}$ – is plotted against its reference value $\mathcal{R}_{P(t+1)}$ for a time difference of $\Delta t = 2\,\text{min}$. In Fig. 5.4d, the same is done for $\Delta t = 119\,\text{min}$. For the larger Δt, the prediction gets more inaccurate, which can be seen by comparing how dense the points are scattered along the diagonal in both plots.

In Fig. 5.4a, the predicted power at $t+1$ $\tilde{P}$ is plotted against its reference value $\mathcal{R}_{P(t+1)}$ for a training dataset size of $N = 100$. In Fig. 5.4b, the same plot is shown for a dataset size of $N = 1\,000\,000$. It is noticeable how many less predictions are plain wrong for a higher dataset size.

What stands out is the fact that dataset sizes and time differences generate different kinds of inaccuracies: With too small datasets, we simply have too large areas in the attribute space with too sparse support of training data, and thus, the function evaluations in those areas are not very accurate[Hug68][GH12]. However, with increasing Δt, the change in power simply becomes a more and more nondeterministic quantity. Therefore, the training data in a way become more and more noised, and the prediction often deviates more from the real value.

6 Conclusions and Outlook

As the scope of this work limited the depth into which I could work on this topic, I will propose here some things that can still be done. For one, from the classification experiment `exp_3_11_1` which is listed in Tab. 5.3 on page 55, by removing data points which can not be classified accurately enough or by rating each data point by it's probability to be in each class, a better dataset can be generated which can then be used to test classification implementations. The experiment used wind speed, solar radiation, DC power generated by a photovoltaic module, difference in wind speed, solar altitude, relative humidity, air pressure, air pressure difference and accumulated precipitation over the last time step to classify data points into the four classes `RAIN`,`SNOW`,`FOG`, and `THUNDER`. The hit rates using a training and test dataset with the same amount of points for each class was 40.79%.

Removing data points which lie between classes, a clustering dataset can also be generated. By confining the classes and creating a dataset where a classification can be learned with $\geq 90\%$ hit rate, this dataset can be used for clustering as well. Clustering does only need one dataset, which is training and test dataset in one: The attribute space part of the dataset is the training data and the image set is the reference. It can be expected that one class will be split up in multiple clusters, however the total result should correlate with the classes in some way. Another idea would be to get a better source for weather event data (rain, snow, thunderstorms, fog) from a source in better proximity to the other sensor stations. However, it is quite difficult to find reliable historical data that is also freely available. Using the precipitation sensors from the IWS data, the `RAIN` class could be refined so that it contains most, if not all, data points where any rain was falling.

For the regression scenario, different time differences can be tested for their predictive accuracy, as already done in the experiments graphed in Fig. 5.3. It might also help a lot to do this based on some weather theory literature, leading to better designs in the attribute spaces. One idea would be to use different time differences Δt for different attributes. The idea behind this is that some physical quantities like air temperature or solar radiation change relatively fast, but others like air pressure change more slowly. For example, choosing $\Delta t = 4\,\mathrm{h}$ for the air pressure and $\Delta t = 30\,\mathrm{min}$ for the air temperature might be more sensible than choosing it as the same amount for both attributes. Throughout this thesis, the lead time τ was always selected as $\tau = \Delta t$. When selecting separate Δt for separate attributes while embedding, a specific lead time might also be selected, potentially leading better results. Also, additional attributes might be embedded based on weather theory or photovoltaic theory.

Bibliography

[Adl13] H. Adler. Langzeitüberwachung von Photovoltaikanlagen, 2013. (Cited on pages 37 and 39)

[Bel03] R. Bellman. **Dynamic Programming**. Dover Books on Computer Science Series. Dover Publications, 2003. URL https://books.google.de/books?id=fyVtp3EMxasC. (Cited on pages 16 and 23)

[BG04] H.-J. Bungartz, M. Griebel. Sparse grids. **Acta numerica**, 13:147–269, 2004. (Cited on pages 16 and 23)

[Bis06] C. M. Bishop. **Pattern Recognition and Machine Learning (Information Science and Statistics)**. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. (Cited on page 28)

[BJ76] G. E. Box, G. M. Jenkins. Time series analysis: forecasting and control rev. 1976. (Cited on pages 43, 57 and 58)

[Bun15] H.-J. Bungartz. Dünne Gitter und ihre Anwendung zur numerischen Quadratur, 2015. URL http://www5.in.tum.de/lehre/vorlesungen/algowiss2/WS06/Handout_02_2auf1.pdf. (Cited on page 17)

[DR11] J. Drechsler, J. P. Reiter. An empirical evaluation of easily implemented, non-parametric methods for generating synthetic datasets. **Computational Statistics & Data Analysis**, 55(12):3232 – 3243, 2011. doi:http://dx.doi.org/10.1016/j.csda.2011.06.006. URL http://www.sciencedirect.com/science/article/pii/S0167947311002076. (Cited on page 11)

[EJHS00] B. Engquist, L. Johnsson, M. Hammill, F. Short. **Simulation and Visualization on the Grid: Paralleldatorcentrum Kungl Tekniska Hoegskolan Seventh Annual Conference, Stockholm, Sweden, December 1999, Proceedings**. Springer-Verlag New York, Inc., 2000. (Cited on page 31)

[Fra11] F. Franzelin. Classification with Estimated Densities on Sparse Grids, 2011.

[Gar06] J. Garcke. Regression with the optimised combination technique. In **Proceedings of the 23rd international conference on Machine learning**, pp. 321–328. ACM, 2006. (Cited on page 28)

[Gar13] J. Garcke. Sparse grids in a nutshell. In **Sparse grids and applications**, pp. 57–80. Springer, 2013. (Cited on page 16)

[GGG13] J. Garcke, T. Gerstner, M. Griebel. Intraday foreign exchange rate forecasting using sparse grids. In **Sparse grids and applications**, pp. 81–105. Springer, 2013. (Cited on page 48)

[GH12] P. I. Good, J. W. Hardin. **Common errors in statistics (and how to avoid them)**. John Wiley & Sons, 2012. (Cited on pages 49, 56, 61 and 65)

[Gie15] J. Giesen. Formula for calculation of solar altitude, 2015. URL http://www.geoastro.de/SME/tk/index.htm. (Cited on page 51)

[GP14] J. Garcke, D. Pflüger. **Sparse Grids and Applications - Munich 2012**. Lecture Notes in Computational Science and Engineering. Springer International Publishing, 2014. URL https://books.google.de/books?id=tezCBAAAQBAJ. (Cited on page 31)

[HHR00] M. Hegland, G. Hooker, S. Roberts. Finite Element Thin Plate Splines in Density Estimation, 2000.

[Hug68] G. Hughes. On the mean accuracy of statistical pattern recognizers. **Information Theory, IEEE Transactions on**, 14(1):55–63, 1968. doi:10.1109/TIT.1968.1054102. (Cited on pages 31, 49, 61 and 65)

[Hun07] J. D. Hunter. Matplotlib: A 2D graphics environment. **Computing In Science & Engineering**, 9(3):90–95, 2007.

[Hun12] J. Hunter. matplotlib Online Documentation, 2012. URL http://matplotlib.org/1.5.0/index.html. (Cited on page 42)

[KD15] C. Klein, Deutscher Wetterdienst. Wetterwarnungsarchiv DWD, Standort Stuttgart, 2015. URL https://www.chklein.de/wetter7/dwdwarnings/bypath/BW/SXX. (Cited on page 51)

[KKSZ11] H.-P. Kriegel, P. Kröger, J. Sander, A. Zimek. Density-based clustering. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, 1(3):231–240, 2011. doi:10.1002/widm.30. URL http://dx.doi.org/10.1002/widm.30. (Cited on page 33)

[Kod14] Y. Kodratoff. **Introduction to machine learning**. Morgan Kaufmann, 2014.

[LH03] A. Luque, S. Hegedus. **Handbook of Photovoltaic Science and Engineering**. Wiley, 2003. URL https://books.google.de/books?id=u-bCMhl_JjQC. (Cited on page 58)

[Mac15] Machine Learning Group at the University of Waikato. Weka Online Documentation, 2015. URL http://www.cs.waikato.ac.nz/ml/weka/documentation.html. (Cited on page 42)

[McK12] W. McKinney. **Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython**. O'Reilly Media, 2012. URL http://books.google.de/books?id=v3n4_AK8vu0C. (Cited on page 42)

[MDA05] G. McLachlan, K.-A. Do, C. Ambroise. **Analyzing microarray gene expression data**, volume 422. John Wiley & Sons, 2005. (Cited on page 36)

[Mos48] F. Mosteller. A k-Sample Slippage Test for an Extreme Population. **Ann. Math. Statist.**, 19(1):58–65, 1948. doi:10.1214/aoms/1177730290. URL http://dx.doi.org/10.1214/aoms/1177730290. (Cited on page 36)

[Nel03] J. Nelson. **The physics of solar cells**, volume 1. World Scientific, 2003. (Cited on page 58)

[OD03] A. J. Onwuegbuzie, L. G. Daniel. Typology of analytical and interpretational errors in quantitative and qualitative educational research. **Current Issues in Education**, 6, 2003. (Cited on page 36)

[Pfl10] D. Pflüger. **Spatially Adaptive Sparse Grids for High-Dimensional Problems**. Verlag Dr. Hut, München, 2010. URL http://www5.in.tum.de/pub/pflueger10spatially.pdf. (Cited on pages 16, 20, 25, 28 and 37)

[Pos15] PostgreSQL Global Development Group. PostgreSQL 8.4.22 Documentation, 2015. URL http://www.postgresql.org/docs/8.4/static/. (Cited on page 42)

[PPB12] B. Peherstorfer, D. Pflüger, H.-J. Bungartz. Clustering Based on Density Estimation with Sparse Grids. In **KI 2012: Advances in Artificial Intelligence**, volume 7526 of **Lecture Notes in Computer Science**. Springer, 2012.

[Rip94] B. D. Ripley. Neural networks and related methods for classification. **Journal of the Royal Statistical Society. Series B (Methodological)**, pp. 409–456, 1994. (Cited on pages 31 and 32)

[Rip96] B. D. Ripley. **Pattern Recognition and Neural Networks**. Cambridge University Press, 1996. URL https://books.google.com/books?id=2SzT2p8vP1oC. (Cited on page 31)

[Sha06] C. Shalizi. Methods and Techniques of Complex Systems Science: An Overview. In T. Deisboeck, J. Kresh, editors, **Complex Systems Science in Biomedicine**, Topics in Biomedical Engineering International Book Series, pp. 33–114. Springer US, 2006. doi:10.1007/978-0-387-33532-2_2. URL http://dx.doi.org/10.1007/978-0-387-33532-2_2. (Cited on page 45)

[SYC91] T. Sauer, J. Yorke, M. Casdagli. Embedology. **J. Stat. Phys.**, 65(3-4):579–616, 1991. (Cited on pages 45 and 57)

[Syk93] A. O. Sykes. An introduction to regression analysis. 1993. (Cited on page 27)

[TA77] A. N. Tichonov, V. J. Arsenin. **Solutions of ill-posed problems**. Winston, Washington, D.C., 1977. URL http://digitool.hbz-nrw.de:1801/webclient/DeliveryManager?pid=2198436&custom_att_2=simple_viewer. (Cited on page 27)

[Tak81] F. Takens. **Detecting strange attractors in turbulence**. Springer, 1981. (Cited on pages 45 and 57)

[Tec10] Technical University of Munich. SG++ Online Documentation, 2010. URL http://www5.in.tum.de/SGpp/releases/index.html. (Cited on page 37)

[TSN99] Y. Theodoridis, J. Silva, M. Nascimento. On the Generation of Spatiotemporal Datasets. In R. Güting, D. Papadias, F. Lochovsky, editors, **Advances in Spatial Databases**, volume 1651 of **Lecture Notes in Computer Science**, pp. 147–164. Springer Berlin Heidelberg, 1999. doi:10.1007/3-540-48482-5_11. URL http://dx.doi.org/10.1007/3-540-48482-5_11. (Cited on page 11)

[Ver15] J. T. Vermeulen. libpqxx C++ PostgreSQL Online Documentation, 2015. URL pqxx.org/devprojects/libpqxx/doc/2.6.4/html/Reference/. (Cited on page 42)

[WA14] T. Wurster, H. Adler. SenseTrace. Internes Dokument, 2014. (Cited on page 37)

[Was13] L. Wasserman. Old Faithful dataset, 2013. URL http://www.stat.cmu.edu/~larry/all-of-statistics/=data/faithful.dat. (Cited on page 11)

[WFH11] I. H. Witten, E. Frank, G. Holmes. **Data mining : practical machine learning tools and techniques**. The Morgan Kaufmann series in data management systems. Morgan Kaufmann, Amsterdam, Boston, Paris, 2011. URL http://opac.inria.fr/record=b1132751. (Cited on pages 36 and 53)

[WG94] A. S. Weigend, N. A. Gershenfeld. **Time Series Prediction: Forecasting the Future and Understanding the Past**. Addison-Wesley, Reading, Massachusetts, 1994. (Cited on page 45)

[Zin10] B. Zinßer. Jahresenergieerträge unterschiedlicher Photovoltaik-Technologien bei verschiedenen klimatischen Bedingungen, 2010. (Cited on page 37)

All links were last followed on November 22nd, 2015.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature