ALGORITHMS FOR SCALABLE ON-LINE MACHINE LEARNING ON REGRESSION TASKS

JAN H. SCHOENKE

UNIVERSITÄT OSNABRÜCK

In partial fulfillment of the requirements
for the Doctorate degree
in Computer Science (Dr. rer. nat.)

Department of Mathematics and Computer Science
Institute of Computer Science
University of Osnabrück

Osnabrück
25 October, 2018

SUPERVISOR:
Prof. Dr. Joachim Hertzberg

DISPUTATION:
18 April, 2019

LOCATION:
Osnabrück

ABSTRACT

In the realm of ever increasing data volume and traffic the processing of data as a stream is key in order to build flexible and scalable data processing engines. On-line machine learning provides powerful algorithms for extracting predictive models from such data streams even if the modeled relation is time-variant in nature. The modeling of real valued data in on-line regression tasks is especially important as it connects to modeling and system identification tasks in engineering domains and bridges to other fields of machine learning like classification and reinforcement learning. Therefore, this thesis considers the problem of on-line regression on time variant data streams and introduces a new multi resolution perspective for tackling it.

The proposed incremental learning system, called Adaptive Simplicial Multi Resolution Approximation (AS-MRA), comprises a new interpolation scheme for symmetric simplicial input segmentations, a layered approximation structure of sequential local refinement layers and a learning architecture for efficiently training the layer structure. A key concept for making these components work together in harmony is a differential parameter encoding between subsequent refinement layers which allows to decompose the target function into independent additional components represented as individual refinement layers. The whole AS-MRA approach is designed to form a smooth approximation while having its computational demands scaling linearly towards the input dimension and the overall expressiveness and therefore potential storage demands scaling exponentially towards input dimension.

The AS-MRA provides no mandatory design parameters, but offers opportunities for the user to state tolerance parameters for the expected prediction performance which automatically and adaptively shape the resulting layer structure during the learning process. Other optional design parameters allow to restrict the resource consumption with respect to computational and memory demands. The effect of these parameters and the learning behavior of the AS-MRA as such are investigated with respect to various learning issues and compared to different related on-line learning approaches. The merits and contributions of the AS-MRA are experimentally shown and linked to general considerations about the relation between key concepts of the AS-MRA and fundamental results in machine learning.

ZUSAMMENFASSUNG

Angesichts ständig wachstender Datenmengen und insbesondere mit Blick auf das Wachstum des Datenaufkommens ist die Verarbeitung von Daten als kontinuierlicher Strom von entscheidender Bedeutung, um flexible und skalierbare Datenverarbeitungsmechanismen zu etablieren. Maschinelles On-line Lernen bietet in diesem Zusammenhang mächtige Algorithmen, um zuverlässig und schnell Vorhersagemodelle aus Datenströmen zu generieren, selbst wenn sich die dabei beobachteten Zusammenhänge im Verlauf der Zeit verändern. Eine besondere Rolle spielt hierbei die Modellierung von reellwertigen Größen im Rahmen der On-line Regression, diese Art von Problemstellung ist sowohl in vielen Bereichen des Ingenieurwesens für die Modellierung physikalischer Zusammenhänge notwendig, als auch wichtiger Bezugspunkt zu anderen Bereichen des Maschinellen Lernens wie etwa der Klassifikation oder dem Bestärkenden Lernen. Insgesamt behandelt diese Arbeit das Problem der On-line Regression auf zeitvarianten Datenströmen und führt dabei eine neue Multiskalenperspektive für dessen Lösung ein.

Der hier vorgestellte Ansatz der *Adaptive Simplicial Multi Resolution Approximation* (AS-MRA) für Inkrementelles On-line Lernen umfasst eine neue Interpolationsvorschrift auf symmetrischen simplizialen Strukturen, eine Multi-Skalen-Struktur lokaler Approximationen und eine Lernarchitektur zur effizienten Verarbeitung von Lerndaten innerhalb der Multiskalen-Approximation. Dreh- und Angelpunkt für die erfolgreiche Zusammenarbeit dieser einzelnen Teile ist ein differenzielles Kodierungsschema zwischen benachbarten Schichten der Multi-Skalen-Struktur, welches es erlaubt das zu lernende Modell in einzelne additiv aufeinander aufbauende Teile zu zerlegen, die jeweils durch eine Schicht der Multiskalen Approximation repräsentiert werden. Insgesamt erzeugt das AS-MRA so eine glatte Approximation des gesuchten Modells, wobei der Aufwand zum Lernen und Auswerten der Approximation hinsichtlich Rechenbedarf linear mit der Anzahl der Dimensionalität der betrachteten Daten skaliert, während der potentielle Speicherbedarf hinsichtlich dieser Größe exponentiell wächst.

Die allgemeine und einfache Anwendbarkeit der AS-MRA wird durch das Vermeiden von notwendigen Designparametern sichergestellt. Dabei bietet der Ansatz gleichzeitig über verschiedene optionale Parameter die Möglichkeit die vom Nutzer erwartete Vorhersagegenauigkeit festzulegen, wodurch sich die Struktur des Ansatzes dynamisch zur Laufzeit an die jeweilige Aufgabe anpasst. Außerdem lassen sich Beschränkungen für die genutzten Ressourcen hinsichtlich Rechenaufwand und Speicherbedarf definieren. All diese Faktoren beeinflussen das Lernverhalten von AS-MRA und dieser Einfluss wird zusammen mit verschiedenen weiteren Aspekten des On-line Lernens in den Untersuchungen behandelt und es werden Vergleiche mit anderen verwandten Verfahren aus dem Bereich gemacht. Die dabei herausgestellten Leistungen des AS-MRA knüpfen an grundsätzliche Erkenntnisse und Fragestellungen im Bereich des Maschinellen Lernens an.

PUBLICATIONS

The following list of previously published publications mostly contributes to the motivation if this work while the most recent one contains ideas and figures of this original work.

[O 1] Schoenke, J. H., and Brockmann, W. (2016, November). Skalierbare inkrementelle On-line Regression auf Simplex-Strukturen. In *roceedings 26. Workshop Computational Intelligence* (p. 51).

[O 2] Schoenke, J. H., and Brockmann, W. (2015, December). Incremental Learning on Decorrelated Approximators. In *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on* (pp. 1062-1069). IEEE.

[O 3] Schoenke, J. H., and Brockmann, W. (2015, June). Robustification of Self-Optimising Systems via Explicit Treatment of Uncertain Information. In *IFSA-EUSFLAT*.

[O 4] Schoenke, J. H., and Brockmann, W. (2015, November). Approximatoren für On-line Regression. In *Proceedings 25. Workshop Computational Intelligence* (p. 199).

[O 5] Schoenke, J. H., and Brockmann, W. (2014, November). Machine Learning in Predictive Filtering. In *Proceedings 24. Workshop Computational Intelligence* (p. 55).

[O 6] Buschermohle, A., Schoenke, J., Rosemann, N., and Brockmann, W. (2013, October). The incremental risk functional: basics of a novel incremental learning approach. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on (pp. 1500-1505).* IEEE.

[O 7] Brockmann, W., Buschermöhle, A., and Schoenke, J. H. (2013, September). COBRA - A Generic Architecture for Robust Treatment of Uncertain Information. In *GI-Jahrestagung* (pp. 2727-2741).

[O 8] Buschermöhle, A., Schoenke, J., and Brockmann, W. (2012, July). Uncertainty and trust estimation in incrementally learning function approximation. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* (pp. 32-41). Springer, Berlin, Heidelberg.

[O 9] Buschermöhle, A., Schoenke, J., and Brockmann, W. (2012, October). Inkrementelles Lernen von Takagi-Sugeno Fuzzy-Systemen 1. Ordnung. In *Proceedings. 22. Workshop Computational Intelligence, Dortmund, 6.-7. Dezember 2012* (p. 71). KIT Scientific Publishing.

[O 10] Buschermoehle, A., Schoenke, J., and Brockmann, W. (2011, April). Trusted learner: An improved algorithm for trusted incremental function approximation. In *Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2011 IEEE Symposium on* (pp. 16-24). IEEE.

ACKNOWLEDGMENTS

Having the opportunity to write this doctoral thesis is a gift which I appreciate and I am very grateful to all people who made this possible. First of all, I want to thank the three professors who guided and supported me on my path for this work. Prof. Dr.-Ing. W. Brockmann has been my first contact to machine learning and initiated this thesis by introducing me to the unique challenges of on-line learning in embedded systems. The things I learned during my time in his *Smart Embedded Systems Group* are reflected in the perspectives onto learning approaches here. Prof. Dr. J. Hertzberg made it possible for me to finish this thesis due to his continuous feedback and patient support. Further, I am pleased and honored to have Prof. Dr. E. Hüllermeier in my committee as a co-examiner.

I also wish to thank my former colleagues Andreas Buschermöhle, Jens Hülsmann and especially Jonas Schneider for many interesting discussions and great cooperation in projects and teaching. Some of the ideas and missing features we discussed during my time as a student actually motivated me to keep heading towards the solution presented here.

The journey towards this thesis has been patiently supported by my family and friends. I want to thank my parents Edith and Heinz Schoenke for enabling this scientific journey by encouraging me right from the start and of course in all the years prior to this research. My special thanks go to my girlfriend Anne Sudau for enduring support, respect and motivation.

Osnabrück, 25 October, 2018 Jan H. Schoenke

CONTENTS

ACRONYMS

AS-MRA	Adaptive Simplicial Multi Resolution Approximation
KWIK	Knows What It Knows
ILS	Incremental Learning System
CSL	Cumulative Square Loss
CL	Cumulative Loss
CTL	Cumulative Target Loss
GL	Ground truth Loss
DL	Data Loss
RCL	Root Cumulative square Loss
GLT	Grid-based Look-up Table
RBF	Radial Basis Function network
TSKFS	Takagi-Sugeno-Kang Fuzzy System
MLP	Multi Layer Perceptrons
FFNN	Feed-Forward Neural Network
kNN	k-Nearest Neighbor
SVR	Support Vector Regression
MAR	Memory Access Ratio
PA	Passive Aggressive
RLS	Recursive Least Squares
LRLS	Local Recursive Least Squares
IRMA	Incremental Risk Minimization Algorithm
SIRMA	Second order Incremental Risk Minimization Algorithm
AROW	Adaptive Regularization Of Weights
AROWR	Adaptive Regularization Of Weights for Regression

GH	Gaussian Herding
AMRules	Adaptive Model Rules
LASSO	Least Absolute Shrinkage and Selection Operator
ANFIS	Adaptive Neural Fuzzy Inference System
ELM	Extreme Learning Machines
VL-RLS	Variance-based Local Recursive Least Squares
FLEXFIS	FLEXible Fuzzy Inference Systems
FIMTDD	Fast Incremental Model Trees with Drift Detection

LIST OF SYMBOLS

This thesis follows some general rules for notation. Scalars are represented by lower case letters, e.g. x. Vectors are represented by bold letters, e.g. $\mathbf{x}$. The state of a scalar or vector at a certain point t in time is represented by a subscript index, i.e. $\mathbf{x}_t$. All other subscript index letters represent elements of a vector $\mathbf{x}$, e.g. $\mathbf{x}_i$ for the i-th element. Representing single elements of a vector at a certain point in time t requires two subscript indexes, e.g. $\mathbf{x}_{t,i}$. The following list shows commonly used symbols in this document:

f	model
$\mathbf{x}$	instance, input to a model
$\mathbf{y}$	label
F	Hypotheses space
$\boldsymbol{\alpha}$	linear parameter vector
$\boldsymbol{\beta}$	non-linear parameter vector
$\boldsymbol{\phi}$	set of basis functions
$\boldsymbol{\nu}$	feature vector
t	time step
n	input dimension
m	output dimension
q	feature space dimension
e	error
l	loss function
CL	cumulative loss
p	node position
l_r	r-th refinement layer
r	layer index
$L_p(\mathbf{a})$	Standard L_p norm of vector $\mathbf{x}$
P_n	n-th order Legendre Polynomial

1 INTRODUCTION

1.1 MOTIVATION

The processing and analysis of data streams [1] is a key feature in many classical engineering domains like signal processing, system identification or system control and gains more and more attraction in recent Big Data related domains like (Wireless) Sensor Networks [65], the Internet of Things or Industry 4.0. In engineering related topics the data stream perspective is easy to grasp as the field frequently deals with embedded systems which read sensors and steer actors in order to fulfill some task. Sensor readings come in one-by-one at a certain rate, thus they form a stream of data over time.

In Big Data related applications the data stream perspective is less obvious as here one is concerned with analyzing huge amounts of data that are not necessarily are stream-based. The issues of dealing with large data sets are considered in [20] on a general and formal level. For large data sets the complexity of the processing algorithm and the processing time to get reasonable results becomes relevant. In [20] the good generalization properties of a sequential processing of the data are highlighted as they naturally allow to steer the trade-off between quality and time consumption in a fine-grained manner. An additional problem with Big Data is the need to process data sets that not necessarily fit into main memory and thus many commonly used data analysis algorithms that need to look at all the data as one batch fail to handle these data sets. Here the data stream perspective comes in handy as the whole data set is split into chunks or processed incrementally, i.e. in a one-by-one manner, which allows a good scalability concerning the size of the data set to be processed. All in all, the analysis of data streams is useful in different realms and fosters the perspective of incremental data processing which requires appropriate algorithms in order to consume data in a stepwise manner.

One common task in engineering and Big Data applications on data streams is to learn a model [173, 74, 199]. Models in engineering are used e.g. in order to separate signal from noise, to estimate system state variables or to find optimal control values. In data analysis applications the models describe relations like the predicted quality of a product depending on the quality of its ingredients, the estimated mean time to failure in predictive maintenance or a forecast of power consumption and production in renewable energy. With respect to Big Data, models may as well be simply used to compress the incoming data in an easily accessible manner in order to find aggregations and abstractions. The models f in the above examples all describe mappings between some input values $x \in X$ and output values $y \in Y$. Learning such models from data streams is the task of on-line regression by means of incremental or on-line learning algorithms in which the input and output values are represented as vectors $\mathbf{x} \in X \subset \mathbb{R}^n$ and $\mathbf{y} \in Y \subset \mathbb{R}^m$, respectively.

On-line learning is a subfield of Machine Learning which studies algorithms for processing data streams. Machine Learning itself consists of three main parts, namely supervised learning, unsupervised learning and reinforcement learning. Supervised learning comprises classification as well as the already mentioned regression and is based on learning a certain mapping based on examples of the from $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ is an instance and $\mathbf{y}$ the corresponding label. In classification y belongs to a set of discrete categories, e.g. $\mathbf{y} \in \{0, 1\}^m$ in binary classification. Whereas in regression $\mathbf{y}$ belongs to a real vector space as mentioned above. Algorithms from unsupervised learning only deal with instances $\mathbf{x}$ and thus do not discover a relation between input and output variables, but perform tasks like clustering, dimensionality reduction or density estimation in order to give insights about the structure of the data itself. Another quality of learning is covered by reinforcement learning which is based on reward and punishment. It aims at improving the behavior of agents acting in an environment by rating the actions an agent is able to perform. Reinforcement learning algorithms learn from experience as they observe the actions an agent takes and the rewards it gets and optimize the agent's behavior in order to maximize the expected future reward.

These three traditional parts of Machine Learning are accompanied by the younger fields of semi-supervised learning, where only a small fraction of all available instances is labeled, and active learning, where the learning algorithm is allowed to ask a supervisor to label an instance chosen by the algorithm in order to best improve its prediction performance. Both of these fields make use of supervised and unsupervised algorithms and thus may be counted as part of one of these categories, but they likewise extend the standard supervised learning schema in substantial ways.

The different parts and fields of Machine Learning are all connected on a conceptual level as they strive to make use of data in order to extract useful information. On an algorithmic level, many on-line learning algorithms internally make use of clustering and dimensionality reduction methods in order to adapt to the data at hand and use them efficiently. In addition to that, some approaches use density estimation techniques in order to estimate the validity of their analysis which provides valuable additional information for the user or other consuming algorithms. An even stronger connection is recently bound between regression and classification in data stream analysis as the mean squared error provides a valuable surrogate for typical classification losses and allows for an incremental data processing, [68]. Thus, it makes many classification algorithms capable of efficiently handling data streams of potentially imbalanced classes. Even reinforcement learning for continuous states and action has already been reduced to the task of on-line regression in the Knows What It Knows (KWIK) framework and other adaptive control related approaches in reinforcement learning, [113].

Hence, on-line regression is important for many application domains and is strongly connected to many fields of machine learning. Moreover, the on-line and incremental processing of data makes it possible to explicitly treat and handle time variance, i.e. a change in the observed model over time, because the temporal ordering of the data is preserved while processing them. A time variant data stream is called evolving data stream in the evolving fuzzy systems and computational intelligence realm. The time variance may affect the way the instances of the stream are generated or how they are labeled or both in combination. A change in the labeling process sometimes is called *real* because here the target function to be learned actually changes over time while a time-variant instance generation only affects the kind of examples the algorithm gets over time from the same target function.

Time variant effects in data streams are usually divided into two classes: drifts and shifts. Drifts are slow continuous changes over time, e.g. caused be wear in sensors and actuators, while shifts mark instantaneous and abrupt changes from one step to another, e.g. when holidays cause an abrupt change in the energy consumption of a city. One issue, that naturally rises when dealing with time-variance, is the stability-plasticity dilemma or the question whether to follow a novel example or not. Following a novel example would mean to be plastic and thus to assume that this example contains valuable information about a new and changed target function making it necessary to adapt or even discard the current model. Being static would mean to ignore the novel example assuming that it does not belong to a new target function but rather is contaminated by noise and the current model needs to be protected from being disturbed by this faulty example. This dilemma is always present when dealing with noisy and time-variant data streams and can only be solved in hindsight by comparing different examples or applying statistical tests.

Looking at the rapid growth of data volume to be processed in Big Data applications due to new sources of data like wearables, sensor networks and process monitoring it is reasonable to look at the hardware that runs the algorithms to process the resulting data streams. The computer hardware and architecture changes from time to time and different architectures make some kinds of algorithms preferable to other. Today two main resources a designer needs to keep in mind when developing algorithms seem to be unlimited, namely processing power and memory. Thanks to multi-core processors the processing power of modern CPUs is still increasing although the processor frequency has reached physical limits due to the power wall [82]. This trend makes the parallel programming paradigms even more important as they not only apply to clusters but to single machines.

The available memory today is huge, relatively cheap and is still getting faster in terms of clock-rate and bandwidth. A limiting factor today is the memory latency which has even increased due to the increased size of each single module. Thus, although a huge memory may be available, accessing this memory is costly with respect to latency. Modern caching technologies allow a fast access but only to a small fraction of the whole memory and they are limited by memory bandwidth and latency as well. In consequence, algorithms for current computer hardware may build complex models but should only need a very small fraction of the whole model in order to answer queries or learn from new data due to the bandwidth and latency limitations. In multi-core processors the algorithms can use a reasonable amount of processing power for parallel tasks, but all of these tasks compete against each other for memory access and thus should share the same piece of the model. Further parallelization of on-line regression algorithms on different machines for distributed learning, e.g. on a cluster, requires the splitting of the stream and network communication with even harder bandwidth and latency limitations and is thus outside the scope of this work.

All of these general considerations are now phrased and condensed into the formal problem this thesis considers and the requirements that form the perspective towards the solution this work is heading for.

1.2 THE INCREMENTAL LEARNING PROBLEM

The On-line learning problem is the formal description of learning from data streams. It is formulated as a task for an Incremental Learning System (ILS) that receives a data stream of instances $x_t \in \mathbb{R}^n$ and labels $y_t \in \mathbb{R}^m$ in consecutive time steps $t = 0, 1, 2, \ldots$. The ILS consists of two parts. A hypothesis $\hat{f}_t \in F$ represents the current knowledge of the system about the mapping $y_t = \hat{f}_t(x_t)$. This hypothesis belongs to some set F of parameterized functions $F \ni f_{\alpha,\beta} : \mathbb{R}^n \to \mathbb{R}^m$ of the form in equation (1) where $\boldsymbol{\alpha}$ are parameters influencing the output in a linear way, the parameters $\boldsymbol{\beta}$ influence the output in a non-linear way and the potentially non-linear mapping $\boldsymbol{\phi}$ defines a set of basis functions for the linear parameters $\boldsymbol{\alpha}$.

$$f_{\alpha,\beta}(x) = \boldsymbol{\alpha}^T \boldsymbol{\phi}(x, \boldsymbol{\beta}) \tag{1}$$

A learning algorithm updates the hypothesis $\hat{f}_t$ based on samples of the form (x_{t+1}, y_{t+1}) by adjusting the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ and possibly changing their size while the kind of basis functions ϕ remain fixed as they essentially define the structure of the representation.

The process of learning one sample (x_{t+1}, y_{t+1}) from the data stream is always the same. For each new instance x_{t+1} the ILS predicts the label $\hat{y}_{t+1}$ based on its current hypothesis $\hat{f}_t$ about the relation $\hat{y} = \hat{f}(x)$. After predicting the label $\hat{y}_{t+1}$ the ILS suffers a loss $l(y_{t+1}, \hat{y}_{t+1})$ by receiving the corresponding label y_{t+1} from the data stream and the ILS updates its hypothesis to $\hat{f}_{t+1}$. The goal of the ILS is to minimize the cumulated loss defined in equation 2 over all learning steps and thus learn the relation $\hat{y} = \hat{f}^*(x)$ that best predicts the labels y_t from the instances x_t. Depending on the set of possible hypotheses $\hat{f} \in F$ the best hypothesis $\hat{f}^*$ may perfectly fit the target function f, i.e. $\hat{f}^* = f$, or only approximate the target function $\hat{f}^* = argmin_g\{||f - g||\}$. The former case is referred to as a realizable case but the latter case of approximate solutions represents most real world scenarios.

The labels y_t the ILS receives from the data stream are assumed to be of the form $y_t = f(x_t) + \epsilon$ where f is the actual labeling function and ϵ is an additive noise term. Thus, the ILS has to reconstruct the actual mapping f from noisy samples in order to find the best hypothesis $\hat{f}^*$. This task becomes even more complex when dealing with time-variance as the actual labeling function f_t becomes time dependent. The stability-plasticity dilemma formally originates from labels generated according to $y_t = f_t(x_t) + \epsilon$, which generalizes the task of the learner to track a non-stationary and potentially non-linear target function f_t from noisy samples (x_t, y_t) and highlights the difficulty in judging whether a prediction error is caused by noise or time-variance.

In regression the considered loss $l(y_t, \hat{y})$ is usually chosen to be the squared error as it has many desirable properties like smoothness and convexity, see equation (3). This loss gives rise to one of the central performance measures in on-line learning, i.e. the Cumulative Square Loss (CSL) as shown in equation (4). The CSL measures the performance of a learning system in predicting the unknown target function $f(x)$ over the whole course of learning.

$$\mathrm{CL}(t) = \sum_{i=0}^{t} l(y_i, \hat{y}_i) \tag{2}$$

$$e_t^2 = (y_t - \hat{y}_t)^2 \tag{3}$$

$$\mathrm{CSL}(t) = \sum_{i=0}^{t} (y_t - \hat{y}_t)^2 \tag{4}$$

There are other loss metrics in on-line learning like the absolute error $|e_t| = |y_t - \hat{y}_t|$ which does not prefer small errors to large ones like the squared error does. This absolute error treats all errors equally irrespective of their magnitude which directly affects the behavior of all on-line learning algorithms which make use of the gradient information. Selecting an appropriate loss metric for a particular on-line learning problem can greatly simplify the learning task but requires corresponding prior knowledge. This thesis only focuses on the squared error as a widely used metric in machine learning. Thus, throughout this document the cumulative squared loss is referred to as Cumulative Loss (CL).

1.3 REQUIREMENTS

This section will sort, group and complete the different perspectives onto on-line learning sketched in the introduction so far and state them as requirements for on-line learning systems. The following list of categorical requirements spans the coordinate system for rating algorithms in this thesis but they are inherently connected to each other and these connections may be most important in some applications while negligible in others.

- Accuracy

  The prediction accuracy of a learning algorithm is a quite obvious and easy to grasp measure in rating on-line learning algorithms. The squared cumulative loss is a natural choice to measure the accuracy of an ILS as it takes into account all available samples and indirectly even the development of the hypothesis of the ILS because the course of the CL over time converges to a linear function as the hypothesis of the ILS converges.

  The slope of this linear function depends on the noise term and the approximation quality of $\hat{f}^*$. The former is inevitably determined by the data while the latter entirely depends on the hypothesis set F the ILS makes use of, i.e. the representation, and the ability of the learning algorithm to generalize from samples to a model. Thus, accuracy is strongly related to representation as well as generalization.

  Moreover, representation and generalization are directly connected to each other because a rich representation allows to exactly match many target functions which may appear favorable in terms of accuracy but makes the generalization for the learner very hard due to the problem of overfitting. The term overfitting states that the out-of-sample error in the learning process is much high than the in-sample error, i.e. the learning system only memorizes the data but performs no valid generalization. Thus, a rich representation featuring a huge hypothesis space F requires much more data to achieve a good generalization compared to a restricted representation with a small hypothesis space F. This observation is formally supported by the bias-variance decomposition of the expected approximation error.

$$E_{X,D}\left[(y - \hat{f}(x))^2\right] = \sigma^2 + \mathrm{Bias}\left[\hat{f}\right]^2 + \mathrm{Var}\left[\hat{f}\right] \qquad (5)$$

This decomposition accompanies the consideration of the convergence towards an accurate approximation by the progress of learning in order to get to such an approximation. This raises the question whether the application requires a reasonable hypothesis right from the start or only after a certain amount of time, i.e. learning samples. It may be important to provide a good hypothesis in every step like in any-time prediction settings or only after certain learning episodes of reasonable length. A suitable incremental learning system should support both options, but providing a reasonable hypothesis in every step is usually harder to achieve as the conflict represented in the bias-variance decomposition does not relax in time which is especially important in non-stationary learning tasks.

- Scalability

  A huge or growing amount of the data to be processed is a fundamental motivation for incremental learning as their processing inherently scales linearly in data size. However, there are other aspects of scalability which are more severe. A factor especially important in regression is the scalability towards the dimensionality n of the instances, i.e. the dimension of the input space of the target function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$. This scaling is quite tricky as it is directly related to the *curse-of-dimensionality* which basically states that any fixed number of samples d provides an exponentially less dense sampling of the input space as the dimension n of the input space increases. Thus, samples in low dimensions are much more useful and informative than they are in high dimensional spaces. This fact makes dimensionality reduction techniques very powerful and in some applications fundamental, but they are not within the scope of this thesis and will only appear as components of certain on-line learning algorithms.

  While the sampling gets exponentially sparse with increasing input dimensionality, the potential complexity of the target function f grows exponentially regarding input dimension. Thus, high dimensional data cause two inevitable and severe problems that directly link to the bias-variance decomposition and make the problems discussed there even worse. The increased potential complexity of the target function requires an accordingly rich representation which threatens generalization and increases the necessary amount of data in addition to the exponentially less dense sampling due to the *curse-of-dimensionality*.

For every learning algorithm this increases the conflict between choosing a huge hypothesis set to account for the potential complexity of the target function or to choose a small hypothesis set in order to foster a reasonable generalization from the available data. The design of an algorithm with respect to this fundamental conflict also determines its scalability towards input dimension. The extreme cases of input scalability are linear and exponential as best and worst cases, respectively.

Unfortunately, the scalability of an algorithm is not a scalar measure but is in general different for evaluation and learning. The scalability of an ILS towards evaluation itself is twofold as it touches the computing and memory demands. The scaling of the memory demand refers to the amount of parameters needed to answer a single query, i.e. to predict the label of one instance x_t. The extreme cases here are the need to process the whole model or only a single parameter. The computational demand at least scales like the memory demand but may be even higher, e.g. if the evaluation requires to solve some kind of optimization problem.

The scalability of an ILS toward learning splits into memory and computational demands as well. However, here the memory demands span two different aspects. On the one hand, the amount of parameters needed to do one update step of the model and on the other hand, the memory demands of the total model. The computational demands for learning follow the same considerations as for evaluation.

In total, there are five different aspects of scalability of an ILS towards dimensionality:

- Computational demands for evaluation
- Memory demands for evaluation
- Computational demands for learning
- Memory demands for learning
- Overall memory demands

All of them correspond to key hardware aspects like computing power, memory size and memory access performance. The limiting factor for algorithms that process a large amount of parameters for each sample is memory bandwidth, while algorithms that only require a small amount of parameters are limited by memory latency in their overall processing performance, i.e. the amount of data they can process per time. This processing performance relates to the throughput rate of the learning system and its response time which may be crucial in embedded systems applications with real-time constraints or impact user experience in service applications.

The top four of the above aspects of scalability are easy to state in Bachmann-Landau notation for most algorithms, while the last one - the overall memory demand - depends on many factors like the relation between hypothesis space and target function, the complexity of the target function, the amount of samples, the distribution of the instances, noise and time-variance. Identifying how the total memory amount of an algorithm scales towards the input dimension and to clearly separate this influence from all others is hardly achievable by analyzing an algorithm formally. Thus, the formal consideration of the total memory amount of an algorithm will be simplified to just look at a minimal scaling for a realizable case in order to have a valid lower bound for this value as its natural upper bound would be the memory needed to store all samples.

When focusing on memory access, only the three memory related aspects of scalability from above are relevant which essentially condense into a single ratio between the total memory demand of the model and the necessary subpart for evaluation and learning. This Memory Access Ratio (MAR) simplifies many important aspects of certain approaches but it allows to easily compare them in terms of their memory access efficiency and thus gives insights about the scalability of the memory efficiency towards dimensionality.

- Usability

  Usability is a complex topic in software engineering, but for on-line learning algorithms usability boils down to the number of parameters the user needs to set and how difficult it is to tune these parameters. While the number of parameters an algorithm provides is easy to count formally, the tuning of these parameters is hard to judge beforehand. Nevertheless, on a formal level it is possible to state whether a user-defined parameter belongs to some internal mechanism of the algorithm and thus, forces the user to have prior knowledge about the algorithm or if a parameter offers the user the ability to express prior knowledge about the application at hand or define certain standards in terms of accuracy or response time the user expects the ILS to have in operation.

  Of course, this separation is not strict as every parameter affecting the internals of an algorithm influences the behavior the user can expect and parameters guiding the behavior of the ILS need to affect the internals of the algorithm to make that happen, but most parameters have a strong tendency to either require knowledge about the algorithm or allow to express knowledge about the application and the expected behavior, respectively. Many algorithms offer default values for most of their user-defined parameters and thus, it is reasonable to count the number of necessary parameters and the optional ones separately.

Usability of on-line learning algorithms in general should as well cover the issue of prior knowledge about the application in more detail as for many applications there is plenty of expert knowledge available, but this topic is too rich to be covered here and does not fit the scope of learning from time-variant data streams as the desired algorithms for this problem are fully automatic.

- Validity

  One theoretically sound aspect of validity for on-line learning algorithms is to ask for their convergence against batch processing results after receiving a sufficiently large amount of data. For time-variant data streams there is no actual batch counterpart as the time dependent target is not representable in batch processing which handles all data equally and at once. In on-line learning the concepts for validity are related to estimating the out-of-sample error and reflect the uncertainty about the current hypothesis. Such concepts are not unique to on-line learning, but the processing of a data stream inherently restricts the considered measures to incremental statistical estimators and similar aggregated values which do not require retrospective data access.

  What supports the validity of a hypothesis is the data that contributes to it and what threatens the validity is a lack of data or the quality of the data which may be reduced by noise in labels y_t or by poorly distributed instances x_t. Thus, the validity of a hypothesis is rather a local measure than a global one as in some regions of the input space many samples with low noise may be available while other regions are sparsely covered by noisy samples.

  Validity as an estimation of the out-of-sample error is connected to the generalization power of a hypothesis and to the uncertainty about this hypothesis. Generalization is the superordinate concept of interpolation and extrapolation whose discriminative factor is the position of the evaluation point in the input space with respect to the sample distribution.

  Interpolation is performed in regions with many samples which fully surround the evaluation point, making it an inner point of the sample distribution. Extrapolation is the opposite of interpolation and applies to points that are outside the input regions which are populated with samples or in very sparsely populated ones, making the evaluation point an outer point of the sample distribution. Although the concepts of inter- and extrapolation seem clearly separable by definition, the transition between them is smooth. Depending on the actual sample distribution it may be infeasible to definitely classify certain evaluation points as inter- or extrapolation.

How reliable inter- and extrapolation are depends on the hypothesis set and the sample distribution. A small set of hypotheses together with a uniform sample distribution may allow for valid estimations as basically there is no extrapolation. Other situations with imbalanced instance distributions and huge hypothesis sets may be challenging as the potential extrapolation turns into blind guessing.

Overall, the uncertainty about a hypothesis is based on ignorance and conflict. Ignorance is directly related to the sample distribution properties of inter- and extrapolation as the ignorance in sparsely populated input regions is high which is related to extrapolation regions and the ignorance is low in input regions with many samples where interpolation applies.

Conflict is related to noise in the labels as well as to the bias of the hypothesis set. The connection between conflict and random noise is quite natural as higher noise levels result in more contradicting labels and increases the amount of data needed in order to average out the noise. The effect of contradicting labels also appears if the hypothesis set of the ILS is not rich enough to grasp the target concept.

In such a case the labels only appear to the learning system as noisy, but by averaging out this noise the learning system actually loses some fine grained structure of the target function. Thus, conflict is a good measure to see if there is a problem in learning but does not directly tell why. This aspect again is directly connected to the bias-variance decomposition as any measure of conflict only represents the sum of bias and variance errors but never indicates their relation.

In summary, the validity of a hypothesis should reflect ignorance and conflict with respect to the chosen hypothesis set and position in the input space. To some degree, this can be achieved independent of any particular learning algorithm by tracking the necessary statistics about density and variance of the labels but this lacks the connection to the hypothesis set and may cause considerable additional processing and memory demands.

- Efficiency

Efficiency in computer science covers many different aspects, for on-line learning the traditional topics like power or cost efficiency of course apply, but more interesting is the issue of data efficiency. As pointed out in the description of accuracy and validity above it may appear useful to have a hypothesis set that grows by processing new data. The bias-variance decomposition and the Hoeffding inequality support this idea as a small hypothesis set bounds the variance term and enhances generalization. A possible downside of a growing hypothesis set may be the loss of information due to a coarse representation in the beginning while the hypothesis set is small, i.e. a high bias of the model to foster generalization may yield an inefficient data handling. In an optimal case, the user has some knowledge about the expected complexity of the target function and can choose the hypothesis set accordingly. This way the variance term may cause a poor generalization before enough data is gathered to train the model, but the data efficiency is high, because no information from the data gets lost by training a model with insufficient expressiveness.

The topic of data efficiency becomes even more urgent when dealing with time-variance as in this case there is no guarantee that a static target function lasts for a certain period of time when shifts occur and for drifting target functions the classical concept of convergence towards a static function does not apply at all. This links the data efficiency to the stability-plasticity dilemma and to accuracy as well in the coordinate system for rating incremental learning systems in this thesis. But it also includes the perspective of convergence rates in stationary on-line learning as faster convergence relates to higher data efficiency. In summary, accuracy asks how precise a hypothesis might become, while efficiency asks how long it takes to get there in terms of sample demand.

1.4 GOAL AND OUTLINE OF THE THESIS

The requirements defined in section 1.3 span the coordinate system to rate different approaches in this work. These requirements cover general aspects of on-line learning which apply to all incremental learning systems irrespective of their particular application context. When focusing on the potential of incremental learning systems to support reinforcement and control application contexts the smoothness of the learned approximation becomes relevant in terms of a valid derivative which is learned together with the actual output. Smooth approximations not only provide a globally well-defined derivative, but also ensure a reasonable correspondence between the learned output surface and its local derivatives without explicitly stating it as learning samples. As this smoothness aspect is not important in on-line learning in general, it is not part of the considered rating system, but it is part of the goal definition in this thesis as it is an important feature for an incremental learning system which is widely applicable, especially for supporting other machine learning domains. Hence, the goal of this thesis is to show the current boundary of on-line learning approaches in terms of approximation properties and memory efficiency and to develop an on-line learning system that pushes this boundary towards scalable and smooth approximations. Other aspects of the coordinate system like accuracy, validity and efficiency are considered, too, but focus and priority is on smoothness and scalability in terms of a low memory access ratio.

2 RELATED WORK

As the goal of this thesis is to develop a new incremental learning system, the relate work treats the fundamentals of approximation structures and on-line learning algorithms as well as current approaches in the field of on-line and incremental learning systems. The special emphasis on scalability and smoothness highlights the role of the hypothesis representation in on-line learning systems and thus, the approximation structure being used. Moreover, the memory access ratio of every on-line learning system is mainly determined and limited from below by the approximation structure it uses and thus, the related work focuses on scalability properties of the approximation structures. The accompanying fundamentals for on-line learning algorithms cover techniques for estimating the linear parameters of an approximation structure. In this field on-line learning algorithms consider stationary target functions while adaptive learning algorithms extend the scope to tracking non-stationary targets. The particular on-line learning approaches reviewed here are organized according to the approximation structure they make use of, as most of them combine an approximation structure, an on-line learning algorithm and a structure-specific component for non-linear parameter estimation.

2.1 APPROXIMATION STRUCTURES

Approximation structures are inherently important in on-line learning as they describe the representation of the desired solution and thus, define the shape and size of the hypothesis set. These representations are not restricted to on-line learning and build the foundation for many approximation tasks. Some important properties to categorize different approximation structures are already mentioned in the introduction and are described here in more detail in order to highlight their relation to the considered coordinate system for ranking.

The general formal description of approximation structures in equation (1) introduces two different kinds of parameters, the linear ones $\boldsymbol{\alpha}$ and the non-linear ones $\boldsymbol{\beta}$. This distinction is fundamental for on-line learning as the impact of the parameters onto the output shapes the optimization problem that needs to be solved in order to update the parameters to new data and this links to the topics of data efficiency and hypothesis set size.

On-line learning algorithms focus on adapting the linear parameters $\boldsymbol{\alpha}$ as optimizing them yields a convex optimization problem which can be efficiently solved on-line in one step. The on-line adaptation of non-linear parameters $\boldsymbol{\beta}$ yields a non-convex optimization problem with subsequent problems of iterative optimization like local minimums, exploration-exploitation dilemma, quality uncertainty and so on. While the non-linear parameters $\boldsymbol{\beta}$ are harder to train they allow for a more compact representation of a hypothesis set in terms of number of parameters. This parameter efficiency may impact the data efficiency accordingly if the non-convex optimization problem is easily traceable. As this is not the case in general, the adaptation of non-linear parameters in on-line learning is usually done indirectly by means of heuristics or surrogates. Thus, when looking at different approximation structures it is important distinguish which parameters are linear and which ones are non-linear, because only the linear parameters are directly handled by on-line learning algorithms.

The basis functions $\boldsymbol{\phi}$ define a certain kind of approximation structure and are shaped by the non-linear parameters $\boldsymbol{\beta}$. In on-line learning each sequential sample $(\mathbf{x}_t, \mathbf{y}_t)$ only provides local information about the target mapping at one point in the input space. Hence, when integrating a new sample into the current hypothesis, it is important to take care of both, the appropriate representation of the new sample and the preservation of knowledge about the target mapping in regions far away from this sample. Whether the local information of a sample can be incorporated locally into the hypothesis depends on the actual shape and support of the basis functions $\boldsymbol{\phi}$. One central aspect of locality is the support of each basis function ϕ_i. A basis function ϕ_i is called locally supported if $|supp(\phi_i)| \ll |X|$, i.e. the parameter corresponding to this basis functions affects the output only in a small subspace of the whole input space $X \subset \mathbb{R}^n$. Globally supported basis functions ϕ_i span the whole input space $supp(\phi_i) = X$ and thus, the parameters corresponding to them have a global impact onto the output. The support of the basis functions $\boldsymbol{\phi}$ determines the sparsity of the resulting feature vector $\mathbf{v}_t = \boldsymbol{\phi}(x_t) \in \mathbb{R}^q$ and thus, the ratio between total model parameters and model parameters necessary for learning and evaluation. This ratio greatly affects the scalability of an approximation structure with respect to MAR, see equation 6 for a formal definition of MAR.

$$MAR(\boldsymbol{\phi}) = \max_{x \in X} \frac{\sum_{i=1}^{q} \mathbb{1}_{\phi_i \neq 0}}{q} \tag{6}$$

A second aspect of locality is the shape of a basis function ϕ_i as even globally supported basis functions may effectively be local if their influence is centralized to a small region of the input space. A basis function with centralized shape shows only one global maximum and vanishes rapidly outside the small region of the input space it belongs to. In terms of approximation properties the opposite of a centralized shape is a monotone one, i.e. a function that is monotonically increasing or decreasing over the whole input space.

The described concepts for locality based on support and shape cover the extreme cases along these axes, i.e. local and global support as well as centralized and monotone shape. The rich variety of approximation structures in the literature fills the entire spectrum between these extreme cases and even includes periodic shapes and further special cases.

Another important formal property of the basis functions $\boldsymbol{\phi}$ is their continuity as the basis function with the lowest continuity defines the continuity of the approximation structure as a whole. Here only non-continuous, non-differentiable and differentiable or smooth functions are distinguished as they relate to three different classes of target functions.

As discussed in the requirements concerning efficiency, an increasing complexity of the approximation structure that adapts to the available data may be beneficial for balancing and bounding bias and variance components of the expected approximation error. Although in principle all approximation structures allow to change their expressiveness on-line, it depends on the exact properties of the approximation structure how demanding such an adaptation is. On the one hand, it is important to see how fine grained the adaptation of the expressiveness can be and how this scales with input dimensionality. In the best case the expressiveness can be adjusted on a single parameter level, in the worst case any adjustment causes an exponential growth of the total number of parameters. On the other hand, it is questionable if the current hypothesis remains valid partly or in total when increasing the expressiveness of the approximation structure. So, expressiveness adaptation can support learning as long as the approximation structure allows for a fine grained adaptation and the validity of the hypothesis is preserved.

The scope of the considered approximation structures is rather broad, but the focus on their scalability and continuity properties is quite narrow. So, for more general considerations and formal introductions to the principle approximation structures reviewed in this chapter, please refer to the corresponding literature like [77, 63, 56, 135, 188, 59]. The textbooks [63, 135, 188, 59] handle the approximation structures with a more general perspective and in contexts different from on-line learning. The review in [77] focuses on monotonic functions, but covers many different structures and has a similar perspective with respect to the requirements defined here. Closely related to the scope and requirements of this thesis is [56], but the list of approximation structures there lacks some approaches and its review follows a different focus.

2.1.1 *Polynomials*

Polynomials are among the oldest and best studied approximation structures, [154, 178, 61]. They form globally supported basis functions that cover the full spectrum from monotone to centralized shapes. The following review of different polynomial bases is restricted to the one dimensional case, followed by the description of ways to use polynomials in higher dimension.

In their monomial basis $1, x, x^2, ...$, polynomials form a mainly monotone shape and are either defined by their degree or by choosing certain powers. In this basis, polynomials are hard to train due to an ill-conditioned optimization problem for the parameter update. A common alternative yielding a well-conditioned optimization problem are Legendre- or Tschebyscheff-Polynomials which more or less form an oscillating shape of the basis functions ϕ_i. Although one may omit certain powers in Legendre- and Tschebyscheff-Polynomials as well, usually they are defined only by their degree. The centralized shape of a polynomial basis is realized using Bernstein-Polynomials which even form a partition of unity, i.e. the sum of all basis functions adds up to one everywhere in the input space and Bernstein-Polynomials are only defined by their degree.

Lagrange-Polynomials introduce another concept of locality to polynomials as they are defined over interpolation points $x_0, ..., x_n$ and each basis function ϕ_i belongs to one such point as $\phi_i(x_j) = \delta_{ij}$, i.e. each basis function equals one at their corresponding interpolation point and zero at all other interpolation points. The overall shape of the basis function is oscillating and the corresponding interpolation point not necessarily marks a global maximum. Thus, the shape of the basis functions and the polynomial degree of the approximation are completely determined by the positions of the interpolation points and can even produce a behavior similar to the bases mentioned above. A crucial aspect for placing the interpolation points is again the resulting condition for the parameter update.

When dealing with high dimensional problems polynomials range from linear scaling to exponential scaling. In the monomial basis all kinds of scalings are possible as one can use cross terms like in equation (7) or only pure monomials as in equation (8)

$$\phi(x)_{\text{cross}} = (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2, ...) \tag{7}$$

$$\phi(x)_{\text{pure}} = (1, x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3 ...) \tag{8}$$

A restricted basis set without cross terms scales linearly towards the dimensionality at the cost of limited expressiveness and strong assumptions about the interaction between the different inputs. The variant with exponential scaling uses all cross terms and defines the degree of the polynomial by the maximum of the powers of all monomials which builds a very expressive hypothesis set at the cost of high computational and memory demands.

Due to their construction Legendre-, Tschebyscheff-, Bernstein- and Lagrange-Polynomials only allow for an exponential scaling as their extension to higher dimensions is based on the tensor product of their one-dimensional basis functions. For Lagrange-Polynomials the tensor expansion has a geometric interpretation as the interpolation points in higher dimensions form a regular grid. The issue of ill-conditioned parameter updates becomes even more severe in high dimensions. This favors well-conditioned and local basis functions for the tensor expansion as these properties are preserved in higher dimensions due to the tensor expansion.

For the Lagrange-Polynomial, the positions of the nodes are non-linear parameters and the corresponding heights for the interpolation are linear ones. For all other polynomials the coefficients are their only linear parameters and their degree or their set of powers are non-linear ones.

The expressiveness of a polynomial is determined by its degree. Thus, a growing expressiveness in the on-line case results in an increasing degree. This yields an additional amount of parameters that grows exponentially in dimensionality and thus does not allow for a fine grained steering of the expressiveness of the approximation structure. In a monomial basis it is possible to add individual monomials at the cost of additional memory demands to explicitly store the actual monomial basis. Changing the degree of a polynomial may also invalidate the former hypothesis as an increased expressiveness may turn a non-realizable case into a realizable one and thus yield a completely different optimal parameter vector. Due to the globally supported basis functions any change in the set of basis functions affects all parameters and thus makes an invalidation of the current parameter vector likely.

Characteristics: Polynomials

n : Number of inputs		
p : Polynomial degree		
Basis:	Lagrange	all other
Memory demand	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((p+1)^n\right)$
Linear parameters	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((p+1)^n\right)$
Non-linear parameters	$\mathcal{O}\left((p+1)\cdot n\right)$	$\mathcal{O}\left(1\right)$
Evaluation effort		
Linear parameters	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((p+1)^n\right)$
Feature vector	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((p+1)^n\right)$
memory access	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((p+1)^n\right)$
MAR Evaluation	1.0	1.0
Learning effort		
Linear parameters	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((p+1)^n\right)$
Feature vector	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((p+1)^n\right)$
memory access	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((p+1)^n\right)$
MAR Learning	1.0	1.0

2.1.2 *Grid-based Look-up Tables*

A Grid-based Look-up Table (GLT) forms a strictly local approximation, [191, 54, 14]. This approximation structure is based on a regular grid and uses only locally supported basis functions which are centered at their corresponding interpolation points. GLTs are either a pure look-up table ([180]) that maps every input to one element of the table and thus, forms a piecewise constant output or they are combined with linear interpolation between all vertexes of the grid cell, which contains the current instance x_t.

As a non-trivial piecewise constant output is discontinuous, a high quality approximation of smooth or at least continuous target functions usually requires a high grid resolution. Even with linear interpolation the output only becomes continuous and is non-differentiable at the grid points and the boundary of the hypercubes that form the grid in high dimensions, respectively.

Due to their strictly local nature GLTs are easy to train and by design form a sparse feature vector. In the extreme case of pure look-up only one element of the feature vector is non-zero, which is best in terms of scalability towards dimensionality andMAR. With linear interpolation the feature vector is still sparse but the scaling is already exponential as the interpolation takes into account every vertex of the hypercube an input falls into.

The great scalability of the GLTs without interpolation comes at the cost of a discontinuous approximation which requires a high grid resolution to form accurate approximations and thus yields a huge total amount of parameters especially in high dimensions as the total number of parameters of a GLT always scales exponentially towards input dimension. The overall memory demand scales accordingly and is dominated by the stored heights for the grid vertexes as linear parameters. The only non-linear parameters in GLTs are the grid resolution and vertex positions which scale linear towards input dimension.

The approximation quality of a GLT is greatly improved by using linear interpolation, but this raises the scalability directly to the exponential level as all vertexes of a hypercube need to be considered for interpolation. Nevertheless, the feature vector remains sparse even when using linear interpolation because the total amount of parameters in the grid is huge compared to the number of vertexes in one hypercube.

These properties and the fact that GLTs are easy to store and to evaluate make them very powerful for applications with no more than four or five input dimensions. With such low-dimensional inputs the GLTs even benefit from current memory development as they allow finer grids and thus, higher accuracy.

Increasing the expressiveness of a GLT on-line equals a grid refinement. The additional amount of parameters introduced in this step grows exponentially with respect to the dimensionality. The grid refinement has global impact as the grid needs to remain regular, but the heights of the additional nodes can easily be set by interpolating neighboring heights. This allows a nearly seamless migration to more expressive hypothesis sets in terms of the validity of the former hypothesis as the local information of the coarser grid is mainly preserved. The increased grid resolution raises the learning sample demands accordingly, but the validity of the old hypothesis is only affected in regions where the resolution is actually increased and remains the same elsewhere. All in all, GLTs do not allow for a fine grained expressiveness steering but guarantee the validity of the hypothesis due to the locally supported basis functions and the proper initialization of the additional parameters using interpolation.

Characteristics: GLTs

n : Number of inputs		
r : Nodes per input		
Interpolation:	flat	linear
Memory demand	$\mathcal{O}(r^n)$	$\mathcal{O}(r^n)$
Linear parameters	$\mathcal{O}(r^n)$	$\mathcal{O}(r^n)$
Non-linear parameters	$\mathcal{O}(r \cdot n)$	$\mathcal{O}(r \cdot n)$
Evaluation effort		
Linear parameters	$\mathcal{O}(1)$	$\mathcal{O}(2^n)$
Feature vector	$\mathcal{O}(n)$	$\mathcal{O}(2^n)$
memory access	$\mathcal{O}(n)$	$\mathcal{O}(2^n)$
MAR Evaluation	r^{-n}	$(r/2)^{-n}$
Learning effort		
Linear parameters	$\mathcal{O}(1)$	$\mathcal{O}(2^n)$
Feature vector	$\mathcal{O}(n)$	$\mathcal{O}(2^n)$
memory access	$\mathcal{O}(n)$	$\mathcal{O}(2^n)$
MAR Learning	r^{-n}	$(r/2)^{-n}$

2.1.3 *Splines*

Splines fill the gap between polynomials and GLTs as they are based on a grid like the GLTs, but introduce additional local interpolation schemes, [40, 41, 168, 117, 164]. They aim at combining the good approximation properties of low degree polynomials with the benefits of local approximation in order to form globally accurate and easy to learn approximation structures.

The classical spline approximation performs an input segmentation using a grid which not necessarily needs to be regular and defines a polynomial of a certain degree over each input segment. Boundary conditions for the different polynomials link each pair of neighboring input segments to each other and hence, form a set of global continuity conditions for all local polynomials in the different input segments. Altogether, this forms a set of linear equations that incorporates the data, the local polynomials and the boundary conditions and its solution yields the globally smooth approximation of the data.

For on-line learning, (Cardinal) B-Splines form a reasonable alternative to the batch learning approach of classical spline approximation. The B-Splines form a basis of the Spline function space and thus define the set of basis functions in equation (10). The shape of a single B-Spline depends on its degree and the placement of the nodes it spans. A 0th order B-Spline function spans one node and forms a locally constant function, just like the GLT without interpolation. The 1st order one spans three nodes in one dimension and yields a piecewise linear approximation, identical to the GLT with linear interpolation. Thus, GLTs are special cases of low degree B-Spline interpolation. The 2nd order B-Spline is the first differentiable kind of this family and already spans 5 nodes. In general a k-th order B-Spline is in C^{k-1} and spans $2k+1$ nodes. By spanning multiple nodes, the B-Splines implicitly encode the boundary conditions of the general splines and this directly affects the sparsity of the feature vector. The resulting MAR and scalability depend on the B-Spline order the same way.

$$B_{i,0}(x) = \mathbb{1}_{[p_i,p_{i+1}]} \tag{9}$$

$$B_{i,k}(x) = \frac{x-p_i}{p_{i+k}-p_i}B_{i,k-1}(x) + \frac{p_{k+i+1}-x}{p_{k+k+1}-p_{i+1}}B_{i+1,k-1}(x) \tag{10}$$

The extension of B-Splines to higher dimensions scales exponentially as all interpolation is based on a grid. An alternative are Simplicial B-Splines, which are defined on a simplicial mesh and not on a hypercubical grid. The simplex tessellation of the input space allows for a linear scalability for evaluating a Simplicial B-Spline approximation, but comes at the cost of handling boundary condition explicitly for learning. Thus, the scalability for learning a Simplicial B-Spline depends on the chosen continuity. In the base case, the scalability is linear towards input dimension, but this is only possible up to C^1 continuity requirements in Simplicial B-Splines. For all higher continuity degrees the geometry of the boundary conditions yields an exponential scaling because the parameter scope for learning becomes global, i.e. all parameters need to be considered for learning one sample while taking into account the continuity constraints. Nevertheless, Simplicial B-Splines are remarkable as they allow for a linear scalability at least for evaluation while having a high flexibility in meeting certain properties of the target function in terms of continuity. Their main drawback is the scaling for learning which restricts their application to low-dimensional inputs like for GLTs.

For this approximation structure the positions of the nodes are nonlinear parameters and the corresponding heights for the interpolation are linear ones. The expressiveness of B-Splines behaves the same way as for GLTs when subject to on-line adaptation. Only the Simplicial B-Splines behave worse because the degree of the polynomials localized in each simplex is easy to increase but the resulting change in the boundary conditions is much more complex and has global impact.

There are many special applications of Simplicial B-Splines in two and three dimensions with optimized computational demands and for arbitrary node configurations. Unfortunately, they do not scale to higher dimensions and building a problem-specific simplex mesh on-line in higher dimensions is cumbersome as well. The need to explicitly define and store irregular structures in high dimensions increases both computational and memory demands, however a well suited mesh structure based on prior knowledge greatly reduces the challenges of learning and improves data efficiency and accuracy.

Characteristics: Splines

n : Number of inputs

d : Polynomial degree

r : Nodes per input

Spline type:	Base	1st Order Simplex	dth Order Simplex
Memory demand	$\mathcal{O}(r^n)$	$\mathcal{O}(r^n)$	$\mathcal{O}\left(\binom{n}{d} \cdot n!r^n\right)$
Linear parameters	$\mathcal{O}(r^n)$	$\mathcal{O}(r^n)$	$\mathcal{O}\left(\binom{n}{d} \cdot n!r^n\right)$
Non-linear parameters	$\mathcal{O}(r \cdot n)$	$\mathcal{O}(r \cdot n)$	$\mathcal{O}(r \cdot n)$
Evaluation effort			
Linear parameters	$\mathcal{O}((2d)^n)$	$\mathcal{O}(n)$	$\mathcal{O}\left(\binom{n}{d}\right)$
Feature vector	$\mathcal{O}((2d)^n)$	$\mathcal{O}(n \log n)$	$\mathcal{O}\left(\binom{n}{d} \cdot n \log n\right)$
memory access	$\mathcal{O}((2d)^n)$	$\mathcal{O}(n)$	$\mathcal{O}\left(\binom{n}{d}\right)$
MAR Evaluation	$\left(\frac{r}{2d}\right)^{-n}$	$n(r)^{-n}$	$d \cdot n(r)^{-n}$
Learning effort			
Linear parameters	$\mathcal{O}((2d)^n)$	$\mathcal{O}(n)$	$\mathcal{O}\left(\binom{n}{d} \cdot n!r^n\right)$
Feature vector	$\mathcal{O}((2d)^n)$	$\mathcal{O}(n \log n)$	$\mathcal{O}\left(\binom{n}{d} \cdot n!r^n\right)$
memory access	$\mathcal{O}((2d)^n)$	$\mathcal{O}(n)$	$\mathcal{O}\left(\binom{n}{d} \cdot n!r^n\right)$
MAR Learning	$\left(\frac{r}{2d}\right)^{-n}$	$n(r)^{-n}$	1

2.1.4 *Radial Basis Function Networks*

The basic idea of a RBF is to represent a complex non-linear target function using a set of local representatives, i.e. local models, with corresponding receptive fields which define the part of the input space the representative covers, [148, 149, 49, 24, 84, 85]. The receptive fields are defined based on a distance metric between the center of the receptive field, i.e. the position of the representative, and a non-linear radial basis function. Together they form the shape of the reception field. The output of the RBF is calculated as the normalized weighed interpolation of all local representatives. Typically, the Euclidean distance in combination with a Gaussian basis function is used, but there are many different kinds of radial basis functions as well as distance metrics designed for particular purposes and applications.

This basic kind RBFs achieves linear scalability for learning, evaluation and total memory amount, but may suffer from a high constant in big-o-notation due to the number of local representatives in order to achieve an accurate approximation. The approximation provided by Gaussian RBFs is smooth and due to the local nature of the linear parameters RBFs are easy to handle in on-line learning.

An unweighted distance metric like the standard Euclidean one allows to define receptive fields shaped like a ball which not necessarily need to fit to the actual target function and data distribution. Weighing each single input dimension allows to form axis-parallel ellipsoids and thus enriches the flexibility of the receptive fields without compromising the scalability of the approximation structure. A fully arbitrary and unrestricted definition of the ellipsoids requires a weighing matrix to equip the distance matrix with and increases the scalability to quadratic.

The feature vector $\mathbf{v}$ of a typical RBF is asymptotically sparse as the Gaussian decays exponentially and the close-to-zero elements are usually omitted, but the ratio of non-zero elements is not structured like for GLT as it depends on the distribution of the receptive fields and the current evaluation point. This makes it hard to tell in advance which local representatives are relevant for an evaluation and requires to evaluate all receptive fields in order to find the relevant ones. Thus, the ratio between effective and total model parameters may be small for the linear parameters but equals one for non-linear ones. Indexing techniques can help to speed up the search for the relevant receptive fields and thus improve the Memory Access Ratio at the cost of increased overall memory demands. Using an approximate nearest neighbor approach to identify the relevant receptive fields threatens the smoothness of the approximation, as the indexing impacts the output of the RBF.

Another common extension to RBFs are local linear models. The increased local expressiveness of linear models may significantly reduce the total number of local models and thus make the RBFs more efficient. Using linear models neither increases the scalability of the RBFs nor does it alter the MAR.

Due to the local representatives, RBFs allow a fine grained steering of the expressiveness as the local models can be added one by one and do not need to fit into some grid or other regular structure. The only costly aspect of this approximation structure is the global search for the locally relevant models. This becomes challenging if the number and position of the local models is adapted on-line as there is no fixed indexing to speed up the search. Therefore, in general here the RBF is assumed to have a MAR equal to one.

Characteristics: RBF

n : Number of inputs		
K : Number of local models		
Ellipsoid shape:	axis-parallel	unconditional
Memory demand	$\mathcal{O}(K \cdot n)$	$\mathcal{O}(K \cdot n^2)$
Linear parameters	$\mathcal{O}(K)$	$\mathcal{O}(K)$
Non-linear parameters	$\mathcal{O}(K \cdot n)$	$\mathcal{O}(K \cdot n^2)$
Evaluation effort		
Linear parameters	$\mathcal{O}(K)$	$\mathcal{O}(K)$
Feature vector	$\mathcal{O}(K \cdot n)$	$\mathcal{O}(K \cdot n)$
memory access	$\mathcal{O}(K \cdot n)$	$\mathcal{O}(K \cdot n^2)$
MAR Evaluation	1.0	1.0
Learning effort		
Linear parameters	$\mathcal{O}(K \cdot n)$	$\mathcal{O}(K \cdot n)$
Feature vector	$\mathcal{O}(K \cdot n)$	$\mathcal{O}(K \cdot n)$
memory access	$\mathcal{O}(K \cdot n)$	$\mathcal{O}(K \cdot n)$
MAR Learning	1.0	1.0

2.1.5 *Takagi-Sugeno-Kang Fuzzy Systems*

Conceptually, Takagi-Sugeno-Kang Fuzzy Systems (TSKFSs) formalize the linguistic fuzziness of if-then rules in order to describe the potentially complex behavior of a system by a set of simple local rules, [179, 193, 4, 126, 81]. Formally, these fuzzy systems segment the input space into potentially overlapping rules and define a local polynomial model for each rule similar to RBFs. Each fuzzy rule consists of an antecedent which defines the region in the input space where this rule is relevant and the consequence part describes the output of this rule by means of the polynomial model. The output of the whole fuzzy system comprising multiple such rules is defined as the interpolation of the local models. The common combinations of input segmentation and interpolation range from regular grid with B-Spline interpolation to arbitrarily placed nodes with Gaussian interpolation, i.e. a grid-based or RBF like approach.

TSKFSs ensure the desired continuity the same way as B-Splines do, i.e. by using an appropriate interpolation between the local models. In addition to that, they steer the local expressiveness by varying the degree of the local polynomial models. This way continuity and local expressiveness are decoupled and depending on the number of local models and their polynomial degree TSKFSs can behave like polynomials or GLTs. Hence, they share the approximation properties of one or another. Typically, TSKFSs with 0th or 1st polynomial degree are used for the local models in order to achieve a good local scalability while keeping the total amount of local models small due to their expressiveness and a suitable input segmentation and interpolation.

The input segmentation includes the position and span of the local models as non-linear parameters, while the coefficients of the local models are linear parameters. As TSKFSs are a generalization of GLTs, RBFs and Polynomials, to some degree, they show similar properties with respect to the adaptation of their expressiveness. The number of local models is the easiest way to adjust the expressiveness of the whole TSKFS. For grid-based antecedent placements this yields the same coarse grained steering of the number of models as for GLTs while arbitrarily placed antecedents allow for a more fine grained adjustment at the cost of explicitly maintaining their positions. An adaptation of the local model degrees is possible as well which follows the general behavior of polynomials in terms of granularity, except for the validity which is mainly preserved as each adapted polynomial model only affects its local region of the input space. This preserves the validity in other input regions the same way as in other local approximation structures.

In general, the fine grained expressiveness adjustments are all related to an explicit antecedent and monomial handling which fosters overall small fuzzy systems as it increases memory demands. The grid-based and thus, more memory efficient versions only allow for a coarse grained steering of the expressiveness. In essence, for small TSKFS one can afford a fine-grained expressiveness adaptation while for bigger or more complex models the adjustments become coarser. As TSKFSs are meant to be interpretable they are usually restricted to a small number of local models in order to be human readable, but the general concept is not inherently limited.

Characteristics: TSKFS

n : Number of inputs		
p : Local model degree		
K : Number of local models		
r : Nodes per input		
Input segmentation:	grid-based	RBF-like
Memory demand	$\mathcal{O}(r^n \cdot (p+1)^n)$	$\mathcal{O}(K \cdot (p+1)^n)$
Linear parameters	$\mathcal{O}(r^n \cdot (p+1)^n)$	$\mathcal{O}(K \cdot (p+1)^n)$
Non-linear parameters	$\mathcal{O}(r \cdot n)$	$\mathcal{O}(K \cdot n^2)$
Evaluation effort		
Linear parameters	$\mathcal{O}(2^n \cdot (p+1)^n)$	$\mathcal{O}(K \cdot (p+1)^n)$
Feature vector	$\mathcal{O}(2^n \cdot (p+1)^n)$	$\mathcal{O}(K \cdot (p+1)^n)$
memory access	$\mathcal{O}(2^n \cdot (p+1)^n)$	$\mathcal{O}(K \cdot (p+1)^n)$
MAR Evaluation	1.0	1.0
Learning effort		
Linear parameters	$\mathcal{O}(2^n \cdot (p+1)^n)$	$\mathcal{O}(K \cdot (p+1)^n)$
Feature vector	$\mathcal{O}(2^n \cdot (p+1)^n)$	$\mathcal{O}(K \cdot (p+1)^n)$
memory access	$\mathcal{O}(2^n \cdot (p+1)^n)$	$\mathcal{O}(K \cdot (p+1)^n)$
MAR Learning	1.0	1.0

2.1.6 *Multi Layer Perceptrons*

Multi Layer Perceptronss (MLPs) are motivated by nature and try to imitate the way the human brain processes information and adapts to it. The basic entities in MLPs are neurons which perform a non-linear transformation of their input. A single neuron has only limited expressiveness and they are arranged in layers in order to increase the overall expressiveness of the MLP, [111, 159, 80, 83, 160].

The input to a neuron is either a weighted sum of the input vector or the weighted sum of the output of a previous layer of neurons. This way, the input vector propagates through the MLP from the input layer to the output layer. All layers in between the input and output layers are called hidden layers as they are encapsulated and not directly accessible. The size of the input and output layer is determined by the application a MLP is used for, but the design of the hidden layer structure is less obvious and needs to fit the expected target complexity as well as data demands due to the resulting total number of parameters.

This kind of MLP is the standard Feed-Forward Neural Network (FFNN), recurrent neural networks allow loops in the processing path which introduces internal states to the network. This is not within the scope of this work as this kind of networks have fundamentally different properties compared to the other approximation structures reviewed here.

The scalability of the MLP is linear in the input dimension and linear in the size of each layer. All parameters except for output weights are non-linear parameters, which makes MLPs in general very hard to handle in on-line learning settings. The concept of a feature vector applies multiple times in MLPs as the output of each hidden layer is a feature vector and in general none of these feature vectors are sparse.

MLPs allow for different ways of on-line adaptation of their expressiveness. The existing layers can be enhanced by adding neurons to them. Further, the overall number layers can be increased. Both variants are likely to threaten the validity of the former hypothesis as most of the parameters are non-linear and interact with each other on a global scale. The granularity of steering the expressiveness depends on the actual structure of the MLP but scales linear in the wrapping layer sizes.

Characteristics: MLP

n : Number of inputs		
K : Number of hidden neurons		
Hidden layer size:	1	L
Memory demand	$\mathcal{O}(K \cdot n)$	$\mathcal{O}(K \cdot n + L \cdot K^2)$
Linear parameters	$\mathcal{O}(K)$	$\mathcal{O}(K)$
Non-linear parameters	$\mathcal{O}(K \cdot n)$	$\mathcal{O}(K \cdot n + L \cdot K^2)$
Evaluation effort		
Linear parameters	$\mathcal{O}(K)$	$\mathcal{O}(K)$
Feature vector	$\mathcal{O}(K \cdot n)$	$\mathcal{O}(K \cdot n + L \cdot K^2)$
memory access	$\mathcal{O}(K \cdot n)$	$\mathcal{O}(K \cdot n + L \cdot K^2)$
MAR Evaluation	1.0	1.0
Learning effort		
Linear parameters	$\mathcal{O}(K)$	$\mathcal{O}(K)$
Feature vector	$\mathcal{O}(K \cdot n)$	$\mathcal{O}(K \cdot n + L \cdot K^2)$
memory access	$\mathcal{O}(K \cdot n)$	$\mathcal{O}(K \cdot n + L \cdot K^2)$
MAR Learning	1.0	1.0

2.1.7 *Fourier Series*

In signal processing, the time-frequency transformation based on a discrete Fourier Series is a standard technique to decompose, analyze and process signals. It is used to approximate periodic signals as well as none-periodic ones which are decomposed into their Fourier Series and represented compactly as the basis functions of the Fourier Series form an orthonormal basis, [198, 22, 16, 106]. The course of the signal in the time domain is transformed into a set of amplitudes of discrete frequencies. This gives additional insights about the signal not directly visible in the time domain and allows to define additional features in order to separate signal and noise.

The only non-linear parameter of the Fourier Series is the order of the discrete wave spectrum. The coefficients of the sine and cosine terms are the linear parameters which are optimized by on-line learning. As the expansion of the Fourier Series to higher dimensions requires a tensor expansion, its scalability is exponential and the effective and total parameter ratio is one, just like for polynomials. All basis functions are globally supported, have a periodic shape and the overall approximation is smooth. Prior knowledge may help to exclude some discrete frequencies from the spectrum in order to form a more compact representation or to use pure sine or pure cosine terms when approximating odd or even target functions.

The Fourier Series is also similar to polynomials with respect to their expressiveness adaptation. A fine grained adjustment is only possible when adding single frequency components which are hard to identify as the frequency acts as a non-linear parameter to the overall approximation. Increasing the general considered frequency spectrum by one requires to add two parameters when handling a one dimensional input space. In general, this amount of parameters added by adjusting the expressiveness grows exponentially in input dimension and polynomially in the former considered frequency spectrum. Thus, the expressiveness adaptation of the Fourier Series is rather coarse and scales exponentially with the input dimension. Moreover, as all basis functions are globally supported, any change in the feature vector potentially renders the former knowledge in the parameter vector $\boldsymbol{\alpha}$ invalid.

Characteristics: Fourier Series

n : Number of inputs		
p : Number of Frequencies		
Spectrum:	only (co-)sine	full
Memory demand	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((2p+1)^n\right)$
Linear parameters	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((2p+1)^n\right)$
Non-linear parameters	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Evaluation effort		
Linear parameters	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((2p+1)^n\right)$
Feature vector	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((2p+1)^n\right)$
memory access	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((2p+1)^n\right)$
MAR Evaluation	1.0	1.0
Learning effort		
Linear parameters	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((2p+1)^n\right)$
Feature vector	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((2p+1)^n\right)$
memory access	$\mathcal{O}\left((p+1)^n\right)$	$\mathcal{O}\left((2p+1)^n\right)$
MAR Learning	1.0	1.0

2.1.8 *Wavelets*

Wavelets are similar to Fourier Series as they represent another kind of time-frequency transformation, [39, 134, 71, 101, 34]. They are restricted here to the case of discrete transformation as it is the one that forms a LIP-Approximation with a defined feature vector. In contrast to the Fourier Series, the discrete Wavelet transformation preserves information about time and frequency of the transformed signal, i.e. it contains information about when certain frequencies apply and what their amplitude is at that time.

From an approximation perspective, Wavelets are a more flexible counterpart to the Fourier Series as the continuity of the Wavelet Approximation is based on the chosen wavelet and inherits its properties. The wavelet approximation is still compact as the basis functions are required to form an orthonormal basis. Its scalability is exponential as the inherent concept of resolution in wavelets is based on hypercubes and regular input segmentation. The effective and total parameter ratio depends on the support properties of the wavelet used, but at least one parameter in each resolution layer is necessary.

In general, Wavelets and Fourier Series share the same properties with respect to expressiveness adaptation, but as Wavelets localize their parameters in both domains they can make use of spare representations for the feature and parameter vector. As the support of the wavelet parameters grows with increasing degree in the Daubechies family and the higher layer parameters are naturally global this sparsity argument is limited. But for the expressiveness adaptation these limitations are relatively mild, because increasing the expressiveness of a Wavelet approximation is done by adding finer wavelet layers. These finer layers are inherently more local than the former ones and thus fit to the sparsity argument. In this line, the Wavelets allow for a fine-grained adjustment of their expressiveness and as the added parameters express finer details of the already learned approximation, the validity of the former parameter vector remains intact.

Characteristics: Wavelets

n : Number of inputs

l : Number of layers

p : Daubechies order

Type:	Haar	Daubechies
Memory demand	$\mathcal{O}\left(2^{(l-1)n}\right)$	$\mathcal{O}\left(2^{(l-1)n}\right)$
Linear parameters	$\mathcal{O}\left(2^{(l-1)n}\right)$	$\mathcal{O}\left(2^{(l-1)n}\right)$
Non-linear parameters	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Evaluation effort		
Linear parameters	$\mathcal{O}(l)$	$\mathcal{O}(l(p)^n)$
Feature vector	$\mathcal{O}(l)$	$\mathcal{O}(l(p)^n)$
memory access	$\mathcal{O}(l)$	$\mathcal{O}(l(p)^n)$
MAR Evaluation	$l \cdot 2^{-(l-1)n}$	$l(p/2^{l-1})^n$
Learning effort		
Linear parameters	$\mathcal{O}(l)$	$\mathcal{O}(l(p)^n)$
Feature vector	$\mathcal{O}(l)$	$\mathcal{O}(l(p)^n)$
memory access	$\mathcal{O}(l)$	$\mathcal{O}(l(p)^n)$
MAR Learning	$l \cdot 2^{-(l-1)n}$	$l(p/2^{l-1})^n$

2.1.9 *Trees*

Regression and model trees merge the concepts of a rule base in TSKFSs and RBFs with the layer hierarchy of wavelets, [121, 142, 93, 12]. The tree structure of the rule set always covers the entire input space but uses only one input dimension in each node to define a split while having constant or linear models in the leafs of the tree to perform the local approximation. More complex splitting criteria are possible, but reduce the efficiency of the tree structure as they are computationally more demanding and require additional memory which may raise the scalability with respect to dimensionality for each node from constant to linear or even higher. The following discussion assumes the considered tree structures to be balanced as degenerated trees provide no benefit over linear search and therefore do not justify further mentioning.

The overall approximation of trees is discontinuous and separates the input space into piecewise smooth local models. A powerful local model in the leaf nodes may yield a coarse input segmentation and thus a nearly smooth global approximation, while constant local models always result in a discontinuous global approximation with potentially many input segments similar to GLTs.

Due to the strict input segmentation, all basis functions are locally supported and thus form a sparse feature vector. This yields a low effective to total parameter ratio. The tree structure has a very special impact on the scalability regarding dimensionality as this coupling becomes indirect. The effort for evaluating and learning a model tree grows logarithmically in the total number of local models. This fact is related to input dimensionality as the complexity of the target function potentially scales according to it. Hence, as long as the target complexity does not scale exponentially regarding input dimension, a tree representation may find a very cheap approximation in terms of memory and computational demands. This feature of trees is not further discussed here as it relates to dimensionality reduction capabilities inherent to tree representation. Unfortunately, this topic exceeds the focus of this work as mentioned in the introduction.

The splitting boundaries in the nodes are non-linear parameters and they are either fixed at design time due to prior knowledge or induced from statistics about the data stream. The local models provide the linear parameters for on-line learning and each local model is trained separately.

The growing of trees is a standard technique in data processing and in on-line learning the same concepts and problems occur. Axis parallel splits like in kd-trees allow for an easy construction strategy but may yield overly complex trees. Non-linear splits help to limit the size of the tree at the cost of a more complex construction strategy which may involve non-linear optimization tasks. In terms of granularity, trees allow for a fine-grained adjustment of their expressiveness as they apply the standard and mighty *divide and conquer* paradigm. This also supports the validity of the parameter vector as at most the validity of the model which is affected by the split is threatened, but e.g. for linear models this split is a seamless transition to a more fine grained representation preserving the validity of the local model.

The total number of local models is determined by the tree structure and grows exponentially in the depth of the tree. According to the potential growth of the target function complexity with increasing input dimensionality, the number of local models grows accordingly which results in an approximately linear growth of the depth. This relationship is partly illustrated in kd-trees as each input dimension is used for an axis-parallel split in a cyclic manner. Thus, the general connection between target function complexity and input dimension potentially renders the total number of local models to grow exponentially. This is no unique feature of the tree structure and again relates to the *curse-of-dimensionality*.

The power of the tree structure in this regard is to allow for a compact representation for less complex targets even and especially in high dimensions. This property equals the fine grained expressiveness steering of RBFs and makes the direct comparison to approaches with a different structure unintuitive. Therefore, here the tree depth is assumed to have a linear relation to the number of inputs as the considered trees are always balanced. This way the overall Memory Access Ratio of trees is assumed to behave like n/r^n where r is an auxiliary resolution parameter which links to grid-based structures. In effect, this reflects the overall sparse feature vector of the tree structure and its potential to build small and efficient representations even in high dimensional input spaces. In general, the Memory Access Ratio for balanced trees behaves like $\log(K)/K$ where K is the number of local models, but this representation does not allow for a direct comparison to other approximation structures reviewed here. Hence, assuming $K = r^n$ only represents the potentially exponential growth of the target complexity in higher dimensions, but does not affect properties of the tree structure with respect to Memory Access Ratio.

Characteristics: Trees

n : Number of inputs		
p : Tree depth		
Local model:	constant	linear
Memory demand	$\mathcal{O}(2^p)$	$\mathcal{O}(n \cdot 2^p)$
Linear parameters	$\mathcal{O}(2^p)$	$\mathcal{O}(n \cdot 2^p)$
Non-linear parameters	$\mathcal{O}(2^{p-1})$	$\mathcal{O}(2^{p-1})$
Evaluation effort		
Linear parameters	$\mathcal{O}(1)$	$\mathcal{O}(n)$
Feature vector	$\mathcal{O}(p)$	$\mathcal{O}(p \cdot n)$
memory access	$\mathcal{O}(p)$	$\mathcal{O}(p \cdot n)$
MAR Evaluation	$p/2^p$	$d/2^d$
Learning effort		
Linear parameters	$\mathcal{O}(1)$	$\mathcal{O}(n)$
Feature vector	$\mathcal{O}(p)$	$\mathcal{O}(p \cdot n)$
memory access	$\mathcal{O}(p)$	$\mathcal{O}(p \cdot n)$
MAR Learning	$p/2^p$	$p/2^p$

2.1.10 *Nearest Neighbors*

One of the most data centric approximation structures is the k-Nearest Neighbor (kNN) approach of storing past training samples and form a local model for evaluation based on the k nearest neighbors around the evaluation point, [200, 136, 10, 172]. For on-line regression on data streams the nearest neighbor approach faces the problem of a potentially infinite amount of data that needs to be stored. Thus, in addition to selecting the correct number of nearest neighbors and an appropriate interpolation scheme for the local model built from the nearest neighbors, a selection mechanism is necessary that decides whether to store or ignore an incoming sample as well as to keep or delete an already stored one.

The kNN approach scales linearly regarding dimensionality with respect to learning effort. The same is true for the total amount of parameters, which here is the total number of stored samples. For evaluation the situation is different, because in general the scalability of the interpolation schema for the local models applies. This scales at least linear with the number of nearest neighbors considered for each evaluation. Due to the selection mechanism for judging incoming and stored samples, the scalability of learning may be increased to a polynomial degree, e.g. by incorporating matrix multiplication or inversion.

The kNN approach automatically adapts to different data distributions and allows to steer the expressiveness in a fine grained manner on parameter-level. The strong data focus of this approach makes it prone to noise or on the other hand turns noise handling into a relatively expensive task. Noisy samples require a higher number of nearest neighbors to be considered in order to reduce the impact of additive noise, which directly increases the total sample demand accordingly. The actual nearest neighbor search is the main bottleneck of this approach as it scales linear in the total number of samples. Approximate kNN approaches can greatly speed up this search by indexing and hashing techniques, but this approximation directly affects the approximation properties of the whole kNN approach as an approximation structure and e.g. may introduce unforeseen discontinuities or other artifacts.

With respect to time-variance, the kNN approach needs to adapt to both a change in the instance distribution in order to provide the necessary local information where it is required and it needs to adapt to changes in the target function, which is relatively easy as adaptation in this approximation structure is reduced to storing new and deleting old samples. This kind of adaptation highlights a fundamental distinction between the approximation structures reviewed so far and kNN techniques. Although the concept of a feature vector is applicable to kNN, there is no concept of linear parameters as the only thing stored is a set of samples and the linear parameters of the local models are only calculated for evaluation and discarded afterwards. Moreover, the stored samples do not form linear parameters as their output value is fixed due to the stored label. Thus, the kNN approximation structure formally fits to the general model formulation in equation (1), but is used in a completely different way compared to polynomials, GLTs or MLPs. The kNN parameters are updated in a discrete way based on storing and discarding certain samples.

Characteristics: kNN

n : Number of inputs		
k : Number of neighbors		
S : Number of stored samples		
Neighbors:	one	k
Memory demand	$\mathcal{O}(n \cdot S)$	$\mathcal{O}(n \cdot S)$
Linear parameters	$\mathcal{O}(S)$	$\mathcal{O}(S)$
Non-linear parameters	$\mathcal{O}(n \cdot S)$	$\mathcal{O}(n \cdot S)$
Evaluation effort		
Linear parameters	$\mathcal{O}(1)$	$\mathcal{O}(k)$
Feature vector	$\mathcal{O}(S)$	$\mathcal{O}(S)$
memory access	$\mathcal{O}(S)$	$\mathcal{O}(S)$
MAR Evaluation	1.0	1.0
Learning effort		
Linear parameters	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Feature vector	$\mathcal{O}(1)$	$\mathcal{O}(1)$
memory access	$\mathcal{O}(1)$	$\mathcal{O}(1)$
MAR Learning	1.0	1.0

2.1.11 *Support Vector Regression*

Support Vector Regression (SVR) is another data centric approach which maintains a set of support vectors in order to form a global model by solving a constrained quadratic program, [7, 195, 175, 48]. The set of support vectors requires selection mechanisms similar to NN approaches, but in SVR the sample selection also needs to consider the kernel function for measuring the similarity of different samples. The shape and nature of this kernel function define the fundamental properties of the resulting approximation with respect to continuity and smoothness as well as the interpolation and extrapolation behavior of the SVR.

The evaluation of this approximation structure is based on measuring the similarity between the evaluation point and the support vectors in order to form a weighted interpolation of the heights of the support vectors. The weights of this weighted interpolation are obtained by solving the quadratic program. The similarity measure is kernel-based and relies on some distance metric in the feature space spanned by the kernel. For on-line learning an incremental variant of the SVR exists that redirects the data stream processing to incremental or decremental learning of the support vector set and to an update of the global model.

The SVR is the data centric global counterpart to kNN with local models. It possess no linear parameters due to its batch orientated origin and the quadratic programming. The overall approximation behavior is determined by the kernel function and allows to span the full spectrum from smooth to discontinuous functions. The SVR makes use of the kernel-trick and thus only implicitly uses feature vectors. Therefore, the concept of a sparse or dense feature vector does not apply. Nevertheless, the SVR model is global and all support vectors as well as weight parameters of the model are used for evaluation and learning.

The scalability of the SVR mainly depends on the number of support vectors used for the model and thus suffers the same issues as the NN approach with a better noise reduction due to the global scope of the model. The On-line SVR is even more demanding as different sets of support vectors and a number of statistics about the samples need to be tracked in order to achieve a behavior similar to the batch SVR. In general, the On-line SVR scale at least quadratic in the number of support vectors and dimensions, but since this approximation structure is inherently global its MAR equals one.

Characteristics: SVR

n : Number of inputs	
K : Number of support vectors	
Memory demand	$\mathcal{O}(n \cdot K)$
Linear parameters	$\mathcal{O}(K)$
Non-linear parameters	$\mathcal{O}(n \cdot K)$
Evaluation effort	
Linear parameters	$\mathcal{O}(K)$
Feature vector	$\mathcal{O}(K)$
memory access	$\mathcal{O}(K)$
MAR Evaluation	1.0
Learning effort	
Linear parameters	$\mathcal{O}(K)$
Feature vector	$\mathcal{O}(K)$
memory access	$\mathcal{O}(K)$
MAR Learning	1.0

2.1.12 *Summary*

Looking at all the approximation structures reviewed above shows that they cover most aspects of the considered coordinate system. The two main axes are the MAR and the continuity of the approximation structure. Figure 1 shows how the reviewed approaches populate these two axes. Most approaches are loosely placed according to a linear dependency between continuity and MAR. An exception here is the kNN approach which in general needs to look at every element of the data base to find the actual nearest neighbor but only provides a piecewise smooth output. The boundary of achieved approximation properties is highlighted by a dashed line. It includes 1st Order Simplicial B-Splines and GLTs with flat or smooth interpolation. This boundary leaves space for further possible approaches which allow for a higher continuity with reduced MAR. But it is not possible to achieve a non-trivial continuous output by looking at a single parameter for every evaluation. Thus, not every combination of continuity and MAR is realizable by learning linear parameters only.

A key concept in machine learning which complements the reviewed approximation structures is to use ensembles instead of single approximations. Ensemble techniques group many usually simple hypothesis representations in order to build one globally more complex representation which exceeds the capabilities of each single ensemble member. This approach is not reviewed here in line with the other approximation structures as it is applicable independently from a particular structure for the ensemble members. More importantly, using an ensemble neither changes the continuity properties of the overall approximation nor its MAR, because the ensemble inherits these properties from its members and only increases the total memory and computational demands compared to a single ensemble member. Nevertheless, each member in an ensemble usually is far less powerful with respect to approximation capabilities compared to a single approximation structure capable of representing the entire ensemble. Thus, ensembles can help to build compact representations for complex models and allow for a fine-grained expressiveness adaptation on ensemble member level.

Random Forests [21, 115, 169] are a popular example for highlighting this feature of ensembles and they do so by focusing on a limited set of samples and input features for building each ensemble member. Thus, at least indirectly they perform some kind of dimensionality reduction as each tree in the forest only considers a certain set of input dimensions which not necessarily covers all available input elements. This tackles the curse-of-dimensionality by reducing the sparsity of the learning samples. But as dimensionality reduction is not inside the scope of this thesis, these mechanisms are not further detailed. Nevertheless, ensemble methods provide valuable features for handling high-dimensional and complex data, but they do not interfere with the perspectives onto approximation structures in this work, i.e. scalability and MAR. They also provide some unique features for dealing with non-stationary target functions in on-line learning and thus, they are reviewed in section 2.3.3 from an adaptive learning perspective.

Another important aspect in on-line learning settings is the expressiveness adaptation of the different approximation structures with respect to granularity and scalability. The variety of reviewed approaches covers the full spectrum of possible combinations in these aspects, but there is a clear tendency towards local approximation structures for preserving the validity of the learned parameters when adapting the expressiveness. Wavelets are best in validity preservation and they bridge between local and global approximation structure as most of their basis functions are strictly locally supported and only a small fraction spans the whole input space or larger regions within it. The strength of the Wavelets is the possibility to combine the relation between individual parameters to local target function details with a sparse parameter representation. This yields a fine-grained adjustment of the overall model complexity while newly introduced parameters only act locally and therefore preserve the validity of the model in all other input regions. Unfortunately, Wavelets scale exponentially towards input dimension which also applies to their expressiveness adaptation.

A sparse representation of the parameter vector and the features always allows for a fine-grained expressiveness adaptation at the cost of increased memory demands. Hence, memory access in sparse representations is increased, too, and requires to store some index structures for efficient access. But in high dimensional input spaces the data distribution naturally tends to be sparse and therefore fosters a sparse representation by themselves.

In summary, the variety of approximation structures shows all desirable properties with respect to scalability, continuity and granularity but only spread across different approaches and not combined in one structure. The more specific goal for this thesis with respect to approximation structures is to define one which has a low MAR, offers user-defined continuity and allows for a fine-grained expressiveness adaptation which preserves the validity of the parameter vector. The envisioned approximation structure combines aspects of 1st Order Simplicial B-Spines in order to yield a good MAR and scalability, should employ the layer concept of Wavelets in order to preserve the parameter validity and needs to support a sparse representation fostering a fine-grained expressiveness adaptation.

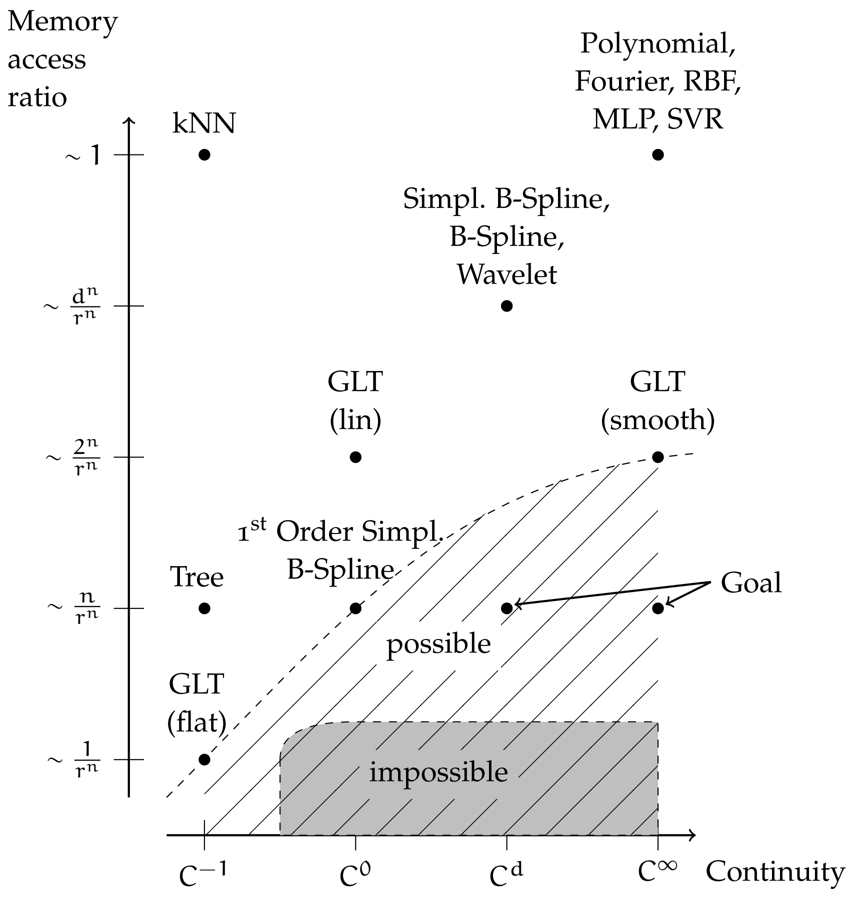


Figure 1: Overview of different approximation structures displayed as points in a coordinate system with respect to their MAR and continuity respectively. The reviewed approaches cover the top and left area of the coordinate system. The dashed line marks the boundary between the current approximation structures and the hatched area of possible further approaches. The gray colored area highlights mathematically impossible combinations of MAR and continuity. The goal of this thesis is to exceed the current boundary by developing a smooth and scalable approximation structure, thus heading for the dots the arrows are pointing at.

2.2 ON-LINE LEARNING ALGORITHMS

On-line learning algorithms adapt the linear parameters $\boldsymbol{\alpha}$ of an approximation structure in order to fit the target function f based on the sequence of training samples $(\mathbf{x}_t, \mathbf{y}_t)$. In this section the target function f is fixed in time and the on-line learning algorithms only need to reduce the noise in the samples in order to find the best approximation $\hat{f}^*$. The extension to the non-stationary or time-variant setting is handled in section 2.3. Further, the formal consideration is limited to scalar labels y_t as the extension to multiple dimensions can be achieved by applying the presented algorithms to each output individually.

The focus on linear parameter adaptation leads to a shift in the perspective of scalability of on-line learning algorithms in contrast to the scalability of approximation structures. For approximation structures the interesting scalability is the one towards the input dimensionality n while for on-line learning algorithms the natural figures to look at for scalability are the size and sparsity of the feature vector an approximation structure provides. Thus, the scalability towards the number of inputs is only indirectly covered. The sparsity as well as other properties of the approximation structure like locally supported basis functions and global continuity constraints are emphasized, because they support an easy to handle learning problem.

On-line learning algorithms are categorized as first order or second order algorithms depending on the kind of information they use to update the parameter vector $\boldsymbol{\alpha}$. First order algorithms only use the gradient information $e \cdot \boldsymbol{\phi}(\mathbf{x}_t)$ about the fitness landscape to update the parameters $\boldsymbol{\alpha}$, i.e. the first derivative of the fitness landscape. The gradient stems from the first derivative of the square loss in equation (3). Second order algorithms store additional information about the learning task in order to estimate second order information about the fitness landscape and thus perform more informed update steps.

A second axis for categorizing on-line learning algorithms is the kind of update they apply to the parameter vector. There are additive and multiplicative updates. Additive updates are used to coordinate many different basis functions in order to form one solution. Multiplicative updates favor the selection of one or a few basis functions from a huge set, i.e. they strive to identify a single expert that performs well for a given task. The idea of multiplicative update does not fit the approximation structures and the on-line learning setting stated here and thus, this topic is not further considered in this review. Especially, time-variance is a fundamental problem for multiplicative updates as parameters of zero weight need a special treatment to rise again.

The notation for on-line learning algorithms in this thesis follows the one in [26] as it provides a common frame for most on-line learning algorithms and allows for an easy comparison between them. The common structure for updates in this notation is given in equations (12) to (13). The update for linear parameters $\boldsymbol{\alpha}_{t+1}$ comprises the passive term $p_1 \cdot \boldsymbol{\alpha}_t$ and the aggressive term $a_1 \cdot (y_t - \hat{y}_t) \cdot S_t \boldsymbol{\phi}(\mathbf{x}_t)$. The two parameters p_1 and a_1 are the algorithm specific components. The second order information matrix S_t becomes the identity when dealing with first order on-line learning algorithms. The update of this second order information S_t follows the structure of the Sherman-Morrison-Woodbury formula for invertible matrices and has a passive and aggressive component as well with algorithm specific parameters p_2 and a_2 accordingly.

$$\tag{11}$$

$$vecalpha_{t+1} = p_1 \cdot \boldsymbol{\alpha}_t + a_1 \cdot (y_t - \hat{y}_t) \cdot S_t \boldsymbol{\phi}(\mathbf{x}_t) \tag{12}$$

$$S_{t+1} = p_2 \cdot S_t + a_2 \cdot S_t \boldsymbol{\phi}(\mathbf{x}_t)^T \boldsymbol{\phi}(\mathbf{x}_t) S_t \tag{13}$$

2.2.1 *First Order On-line Learning Algorithms*

First order on-line learning algorithms are easy to handle as they solely rely on the gradient $e\boldsymbol{\phi}(\mathbf{x}_t)$ and do not store additional information about the learning task at hand. They usually offer a limited number of additional hyper-parameters to adjust the learning behavior of the algorithm. This makes them attractive in terms of computation and memory demands as well as scalability. They treat every sample the same way regardless of the learning history which makes them naturally suitable for time-variant data streams as they will always adapt to a changing target function. The drawback of this always adaptive property is their poor noise reduction in comparison to second order algorithms.

The first on-line learning algorithms date back to the *perceptron* [158] and the Windrow-Hoff algorithm [197] which perform a direct gradient update with $p_1 = 1.0$ and $a_1 = 1.0$ or use a step size parameter λ yielding $a_1 = \lambda$. This general concept is also formulated earlier in [156] and has led to many extensions and generalization afterwards. A popular and more recent work on the family of on-line learning algorithms based on gradient descent is presented in [36] which presents a variety of normalized gradient descent algorithms. The general structure of normalized gradient descent algorithms is shown in equation (14). Due to the normalization, the parameter λ becomes a learning rate $\lambda \in [0, 1]$ rather than a step size but its ultimate effect is the same as it steers the aggressiveness of parameter updates.

$$\alpha_{t+1} = \alpha_t + \underbrace{\lambda}_{\text{Learning rate}} \quad \underbrace{\phi(x_t)e_t}_{\text{Gradient}} \quad \underbrace{\frac{1}{\phi(x_t)^T \phi(x_t)}}_{\text{Normalization}} \tag{14}$$

The learning rates or other hyper-parameters in first order learning are usually global ones and thus result in a constant memory demand and linear scalability towards the feature vector size. A set of parameter specific learning rates is possible, but contradicts the simplicity argument for choosing a first order algorithm with respect to prior knowledge demands as well as additional memory and thus memory access demands. If optimal learning rates for every single parameter or at least for certain blocks of parameters are necessary, detailed prior knowledge is required or the use of second order learning algorithms is indicated.

Other extensions include regularization terms in order to prevent overfitting. The well-known Tikhonov regularization [182] uses a L_2 regularization and is a standard tool in function approximation. In [203] a general framework for L_1 regularized optimization is presented. In general, the regularization allows to express prior knowledge about the target function which is mostly used to control the overall complexity of the approximation without restricting it in the first place by selecting an approximation structure with limited expressiveness. This is especially highlighted in the case of a L_1 regularization as it effectively limits the number of non-zero elements in the parameter vector $\boldsymbol{\alpha}$, while the L_2 regularization only limits its total weight. Thus, the regularization allows to express the expected complexity of the target in the chosen representation by limiting the weight or population of the parameter vector $\boldsymbol{\alpha}$.

The concept of regularization systematically links to former discussion about bias-variance-decomposition and sample efficiency. A complex approximation structure yields a high variance part in the approximation error, but also grants a high sample efficiency as the structure is capable to represent sample details due to its expressiveness. Regularization helps to limit the variance part contribution without limiting the capabilities of the approximation structure to represent detail information in principle. The regularization damps the information gain for processing a single sample, but as long as a sufficient amount of samples contributes to a certain solution the approximation structure will follow. This increases the sample demand, but this lack in sample efficiency is what helps to limit the variance part in the approximation error. In essence, regularization allows to trade sample efficiency for model complexity and therefore long term prediction accuracy. As a design component for on-line learning systems it allows to follow a *start big* approach for choosing the approximation structure. E.g. the frequency spectrum a Fourier Series needs to cover during operation may be hard to know at design time in detail, but a rough estimate covering a wide range may perform poorly due to its complexity. In such a case regularization may come in handy as it helps to find a solution with limited complexity and reasonable prediction performance without knowledge about the actual frequency domain.

Unfortunately, these useful properties of regularization only apply to global models to full extend as all of their parameters interact with each other and thus, the overall parameter weight is related to the target complexity. In local approximation structures the overall target complexity is already bounded due to the input segmentation defined by the local models the approximation structure comprises. In this setting regularization can still be reasonably applied to complex local models, but is not suited for guiding the model overall interaction as this affects fundamental properties of the approximation structure.

If regularization is not sufficient to guide the course of the learning process, constraints can be used to force the current approximation $\hat{f}_t$ to comply to certain properties like monotony, symmetry or boundedness. The work in [35] presents a framework to deal with such constrained optimization problems. Choosing a different approximation structure which enforces the desired properties by construction is a way to avoid the constrained optimization problem, but this may result in an approximation structure with mainly non-linear parameters or other undesirable properties which affect the learning behavior. Thus, constraining the optimization problem allows to choose the approximation structure more freely while ensuring the desired properties of the overall approximation. Following a set of constraints during learning in a strict manner is likely to affect the MAR for learning as the scope of parameters which need to be considered for processing each sample usually widens. Thus, constraining an on-line learning task tends to increase the MAR for learning without affecting the one for evaluation.

The formal interface of feature vectors and linear parameters for the on-line learning algorithms makes it possible to combine any learning algorithm with any kind of approximation structure. Although this is theoretically sound and supported by different convergence results in the case of linear regression, i.e. a linear mapping for the basis functions, the situation is different for non-linear basis functions. The actual kind and shape of non-linear basis functions define the hardness of the resulting learning problem. In particular, the condition of the approximation structure is of major concern as it defines the inherent error amplification in each parameter update, which is especially crucial in the beginning of a learning process as there is no knowledge supported by data that could identify a poor error amplification. In orthonormal approximation structures and local ones as well, the guidance of the gradient information towards the optimal parameter vector is good because the influence of different parameters in different situations is clearly separated in the gradient and allows to adjust the parameter vector $\boldsymbol{\alpha}$ accordingly. In feature spaces spanned by highly correlated basis functions this guidance is easily distracted by noise and less separative in general, which results in a poor tracking towards the optimal parameter vector.

Another aspect in this context is the magnitude of the basis functions as the gradient is directly affected by this property. This also relates to the error amplification due to inaccurate parameters which is inherently different from poorly conditioned parameter updates as the later one defines a property of the learning process while the former one applies to single states in this process. The simple gradient update in equation (12) with $p_1 = 1.0$ and $a_1 = 1.0$ scales according to the magnitude of the gradient. One approach to get rid of this impact of the approximation structure onto the gradient is the Incremental Risk Minimization Algorithm (IRMA) [27]. IRMA treats all approximation structures the same way as it takes into account the total change in the global mapping instead of the change in the parameter vector for the parameter update. Unfortunately, the IRMA approach scales quadratic in the total number of parameters of the approximation structure and is not able to make use of a sparse feature vector. This is due to the need of the inverse Gram-Matrix A^{-1} that links global mapping change and parameter vector change. For the definition of matrix A see equation (15). The IRMA parameters are $p_1 = 1.0$ and $a_1 = A^{-1}/(\sigma + \boldsymbol{\phi}(\mathbf{x}_t)^T A^{-1} \boldsymbol{\phi}(\mathbf{x}_t))$. The inverse Gram-matrix A^{-1} redirects the gradient and the stiffness parameter σ allows to steer the aggressiveness of the updates.

$$A_{i,j} = \int_X \phi_i(\mathbf{x})\phi_j(\mathbf{x})\, d\mathbf{x} \tag{15}$$

2.2.2 *Second Order On-line Learning*

Second order on-line learning algorithms store additional information about the learning task in order to improve their parameter adjustments over time and adapt to the data distribution at hand. This causes increased computational and memory demand as compared to first order algorithms and offers opportunities for a rich variety of algorithms that cover different classes of scalability. One central theme in second order learning is noise reduction or canceling which is a hard task for first order learning. In general, this is achieved by decaying parameter-specific learning rates that force a convergence of the parameter vector and thus yield a noise robust behavior. Different second order learning algorithms focus on certain aspects of this theme or link it to other aspects. Connections between time-variance and noise reduction are reviewed in section 2.3 as noise reduction and adaptation are conflicting goals and handling both at the same time requires another quality of learning.

The basic second order on-line learning algorithm is Recursive Least Squares (RLS) [17] which solely focuses on canceling the additive noise in the samples by reformulating the batch least squares problem into an iterative approach. In order to do so, RLS keeps track of the covariance of the incoming instances and weights the labels accordingly. The covariance of the sample is stored in a matrix S which results in a quadratic scalability towards the total number of parameters and as the scope of RLS is global it is not able to benefit from sparse feature vectors. The convergence properties of RLS are superior compared to first order learning algorithms in terms of convergence rate and robustness against noise.

Other recursive algorithms with subtle differences to RLS are reviewed in [91]. Worth mentioning here is the Recursive Instrumental Variables (RIV) version as it weakens the assumption in RLS of noise and instances being uncorrelated. The other variants in [91] pose no real alternatives to RLS in the on-line learning setting as they handle certain aspects of system identification rather than function approximation.

A second order algorithm similar to RLS but for classification is Adaptive Regularization Of Weights (AROW) [37] which interprets the second order information as a regularization matrix that measures the uncertainty about the parameters. This uncertainty measure is forced to decay due to the learning samples which results in an update close to the one of RLS. In [186] the AROW approach is extended to regression, i.e. Adaptive Regularization Of Weights for Regression (AROWR). An additional handling for non-stationary targets is also introduced in [186] which is further categorized and discussed in subsection 2.3.

Gaussian Herding (GH) [38] shares the covariance interpretation of the second order information of RLS and introduces additional constraints about the development of the shape of the resulting ellipsoid of preferred parameter adaptation directions. There are also variants for GH that project the second order information to a vector and thus allow for a linear scalability regarding the number of parameters and even allow to make use of a sparse feature vector.

Other second order algorithms with linear scalability are more similar to first order algorithms like the local variant of RLS, i.e. Local Recursive Least Squares (LRLS), for the learning of local models, [127, 122]. In the extreme case of locally constant models the LRLS becomes a mean filter with growing window or a gradient descent algorithm with decaying learning rate, respectively. The LRLS scales quadratic in the number of local model parameters but is linear with respect to the total number of models and thus, can exploit a sparse feature vector due to local model interpolation.

The issues of approximation structure specific learning behavior, e.g. condition, locality, basis function magnitude, apply to second order algorithms to some degree as well and thus motivated a second order variant of IRMA called Second order Incremental Risk Minimization Algorithm (SIRMA) both published in [27]. SIRMA stores the second order information in a set of matrices and scales cubic in the number of model parameters. It also links the noise reduction properties of other second order learning algorithms with the always adaptive property of first order algorithms. The parameter adaptation uses a stiffness concept that makes parameter less likely to be changed by increasing their stiffness parameter. This parameter is increased based on the instances x_t and thus acts like an inverse learning rate, but the stiffness is capped to some maximum value which allows for a little amount of constant adaptation.

The noise reduction capabilities of the algorithms reviewed above are limited to mean-free noise, i.e. they are not able to handle impulsive noise or outliers. One way to counteract on outliers is to use parameter adaptation which focuses on quantiles rather than statistical moments. Recursive Least M-Estimates in [205] are one way to realize such a behavior in on-line learning. The incoming samples are first categorized according to the prediction error they produce and rated accordingly. Samples yielding an unexpectedly high prediction error are fully ignored which results in an outlier-robust learning behavior. The issue of detecting outliers and their correct handling is getting even harder in time-variant settings, but this topic is not further considered in this work as adaptive learning algorithms inherently mitigate the influence of outliers.

2.3 ADAPTIVE LEARNING ALGORITHMS

Adaptive Learning algorithms are a special kind of on-line learning algorithms which tackle the problem of tracking time-variant target functions from noisy samples. Although time-variance arises naturally in on-line learning the resulting stability-plasticity dilemma requires to balance the noise reduction properties of second order learning and the steady adaptation of first order learning. The two categories of time variant behavior, i.e. drift and shift [184], are mirrored by two kinds of approaches for handling them, namely implicit and explicit approaches. Another quality of adaptive learning is introduced by ensemble methods which are inherently able to combine implicit and explicit aspects by updating or replacing ensemble members.

Implicit approaches focus on the always adaptive property of first order algorithms in order to keep track of a drifting target function. In contrast to that, explicit approaches handle shifts by treating them as singular events in the time line. When detecting such a shift event a proper reaction for adapting the current hypothesis is taken, e.g. a full or partial restart of an accompanying second order learning algorithm. These two categories are not strict and there are approaches that combine different detection and adaptation methods in order to cover all aspects of time-variance, e.g. [162] combines exponential forgetting and resting components. In [102] sample weighting and selection mechanisms are combined. These two examples highlight another aspect for categorizing adaptive learning algorithms. There are algorithms like [102] which consider a window of past and potentially weighted examples to form and update a hypothesis. On the other hand, approaches like the one in [162] rather keep track of the validity of the hypothesis by means of second order information. Explicit approaches behave fundamentally different with respect to these sample-based and hypothesis-based categories. Detecting a shift in a sample-based approach means to instantly discard all outdated samples and update the current hypothesis accordingly. Resting the second order information in a hypothesis-based approach has a different effect, as it does not directly change the related hypothesis but enforces its adaptation to future samples.

Ensemble approaches complete the picture of different methods for handling time-variance as they are free to take actions on different levels. The combination of ensemble members may change gradually or abruptly, the single ensemble members can be updated, reset or even get replaced. This allows to incorporate different time-variance handling mechanisms accounting for the full spectrum of time-variant behavior and even allows to explicitly handle reoccurrence of former behaviors by muting currently irrelevant members rather than erasing them. All of these adaptive learning concepts are rather forgetting mechanisms than learning algorithms. Thus, adaptive learning is inherently connected to forgetting as some part of the learned hypothesis get invalid over time and needs to be replaced by new information. The different kinds of forgetting mechanisms line up with the sample-based and hypothesis-based approaches. In general, implicit adaptive learning is related to a gradual forgetting either based on a windowing or exponential weighting method.

As mentioned above, the situation for explicit approaches is different because sample-based approaches indeed forget by discarding samples. Hypothesis-based approaches rather mark the current hypothesis as worth forgetting either partly or in total which does not immediately discard any information unless new samples actually contradict the current hypothesis. Ensemble algorithms offer a third category of neither using nor actually forgetting a particular hypothesis but just ignoring it in the combination of ensemble members. These general considerations are further detailed in the following sections and linked to particular approaches in the literature.

2.3.1 *Implicit Time-variance Handling*

Implicit adaptive learning algorithms stick to the always adaptive property of first order learning algorithms but strive for a better noise reduction. One of the easiest ways to do so is to make the gradient, a first order algorithm follows, more robust against noise by calculating it on a window of past learning samples. This windowing approach is used together with gradient descent in [19] with a focus similar to this work as it points out the benefits of the window approach compared to RLS in terms of computational and memory demand, which is true as long as the effort for queuing the samples is smaller than the memory demand of the total number of parameters introduced by RLS. The main design parameter of this approach is the window length, which essentially balances between noise reduction and time-variance adaptation. A short window favors adaptation while a long window helps to reduce noise. This basic windowing approach and common extensions are reviewed in [171, 72, 42].

A very similar approach is presented in [187] where the windowing is combined with RLS and uses an exponentially weighting of the samples in order to promote recent samples and limit the impact of older ones. But the main issue when dealing with windowing approaches is the window length, which has led to different approaches that introduce an adaptive window length [104]. A more general perspective in handling windows is introduced in [130] as here single examples from a set of representatives are discarded. This allows for a more flexible adaptation to reoccurring concepts and helps to reduce the total number of stored samples as newly arriving samples are only added on demand, i.e. if they are necessary to correctly represent the current hypothesis.

The same basic idea is present in [58] by taking into account novelty and similarity measures between the already stored and newly arriving samples. In the more general setting of random sampling this procedure is handled in [190]. Random sampling is closely related to windowing approaches as both head for a representative random subset of a potentially much larger set of samples and in on-line learning this larger set may even be infinite.

Adaptive windowing approaches are also possible without storing a sequence of samples. Combining the RLS algorithms with resetting mechanisms for the second order information allows to implicitly deal with windows of past samples while ensuring fixed and limited memory demand. Simple resetting mechanisms perform a reset of the covariance matrix at certain points in time [73, 138]. The time span between each reset may be constant yielding a periodic reset behavior or may follow a more complex user-defined schedule. A more data-centric approach is to reset the covariance matrix based on its spectral properties [138, 162], e.g. a lower bound for the smallest eigenvalue. Resetting the covariance matrix this way takes into account local hot spots in the data distribution and prevents the learning system from losing track of time-variant behavior in such regions.

RLS with forgetting introduces an exponential weighting scheme to forget old samples and thus allows to balance between the noise robust RLS behavior and a strong focus on recent samples like for first order learning algorithms. A general consideration of this topic is done in [120, 152, 151]. Different versions of the standard RLS algorithm with forgetting have gained much attention in system identification and signal processing in the past decades.

Variable forgetting factors are considered in [60, 183, 29, 112] in order to improve the tracking capabilities of the parameter estimation while avoiding a steadily growing covariance matrix. Stability issues introduced by a forgetting factor are handled in [147] and a robust version of RLS with forgetting is presented. As robustness and numerical stability are crucial for any application, there are further robust versions of RLS as presented in [204, 163, 9].

More elaborate forgetting schemes like the ones in [150, 185] introduce forgetting factors for individual parameters or groups of parameters. This is another means to avoid a steadily growing covariance matrix and thus enhances the stability of the parameter estimation. The concept of parameter-specific forgetting factors is automated using a Bayesian estimator in [174] which lifts the problem of forgetting factor tuning to choosing an appropriate prior for the Bayesian estimation.

In [192] the forgetting factor is adapted according to the magnitude of the squared prediction error. As the squared prediction error increases, the forgetting factor decreases and vice versa. This allows the estimation to balance its tracking and noise reduction capabilities on demand. The overall variety of RLS versions also fostered the analysis of convergence, performance and tracking properties like those handled in[55, 52, 76, 44].

2.3.2 *Explicit Time-variance Handling*

The goal of explicit approaches for time-variance handling is to detect changes in the processed data stream at discrete points in time [8]. The observed behavior in the resulting chunks of data between the identified points is assumed to be stationary. This allows to decouple the time-variance handling and the on-line parameter estimation. Hence, in principle even mechanisms from the field of anomaly, novelty or fault detection [69, 33, 32, 131, 132] are within the scope of this review, but the focus is on supervised on-line learning and especially regression. Another related field which is only indirectly covered here is *change point detection* in statistics [23]. A perspective most familiar with the one in this work is adaptive filtering [78]. The particular approaches reviewed next are meant to span and populate the different categories of explicit time-variance handling but they can hardly cover all individual approaches published in the rich literature for this topic.

The natural domain of time-variant effects for explicit approaches are shifts, i.e. abrupt changes in the behavior of the target function or the observed system. Window-based approaches like in [167, 103, 11] find the best split in a window of past samples based on statistics about the performance of models which are built from different parts of the window. Various statistical hypothesis tests can be applied in the detection mechanism, like Page-Hinkley test [146, 139, 2, 51], Bayes test [170], t-test [177], Kolmogorov-Smirnov-Test [133, 97, 194], etc. or the according concentration inequalities or combinations like in [109]. Which test to select depends on prior knowledge about the involved probability distributions, expected sample sizes and other application related issues. Despite the pure change detection, there is also work on quantifying the detected change in a meaningful way as in [100]. This gives valuable feedback in monitoring applications as it allows to separate small and large changes, but the resulting actions taken in explicit approaches are always the same.

In [66] the performance of an on-line learning system is observed in order to define warning and drifting states in time. Once the performance of the on-line learning system decreases to the warning level all following samples are stored in a window. At some later point in time the on-line learning system may cross the drifting performance level which leads to a retraining based on the windowed data which date back to the last warning level appearance. This helps to minimize the memory demand as the window is only maintained during the change detection and handling.

The focus of explicit approaches on detecting shifts in the data makes them less sensitive for drift detection as drifts induce only little change in each time step but over a long period of time. This problem is tackled in [6] by extending the detection method in [66] to a sequence aware version which not only considers the amount of performance drop but also the distribution of erroneous predictions on the time line.

When dealing with shifts, it is quite natural to look for reappearing behaviors as in [196]. Discovering the reappearance of a former behavior requires to store outdated models and choosing the correct one in case of a shift detection. This approach also bridges to ensemble methods as there is a set of stored models. These models are combined based on a *the winner takes it all* rule, but it is also possible to build a completely new model which might lead to discarding an old one following a least recently used scheme.

The tracking of multiple different models is even more efficient when using regression trees as shown in [93], because it is possible to replace only parts of the model. This allows to react to shifts at different scales by replacing the whole tree, a single node or only one leaf depending on the impact of the shift. Another feature of this regression tree which again links to ensemble methods is the so called *option tree*. This kind of tree may refuse to use a single splitting criterion when growing new nodes by keeping track of multiple nodes with different splitting criteria and independent subtrees related to them. This enables a compact representation and easy handling of different regression trees which only differ in certain aspects. With respect to the growing strategy, the option trees allow to start small while growing potentially overly big due to the option nodes, but each option node usually converges to a regular node in the long run yielding a standard tree structure.

2.3.3 *Ensemble Methods*

The general features of ensemble methods and why they are able to outperform single approximation approaches are illustrated in [43] and [107] provides a recent survey on different on-line learning ensemble techniques. Here the focus is on the potential of ensemble methods for handling time-variant behavior in data streams. The power of ensemble methods in handling time-variant behavior stems from the additional architectural layer of combining different models to one ensemble output. The two standard ensemble methods in machine learning, i.e. bagging and boosting, originate from classification and batch methods, but they have been adopted to data stream analysis as well [145, 144, 143].

BASIC CONCEPTS

The basic idea in bagging is to start with a single data set and to produce multiple versions of this original set by random resampling with replacement. The ensemble build from this variety of resampled data sets consists of members all trained using one individual version of the data sets. This way all ensemble members can be generated in parallel and they are using their own data. The on-line adaptation of bagging for data stream analysis distributes the incoming samples to all ensemble members according to a Poisson distribution. This simulates a random resampling as some ensemble members get no access to the current sample while others may receive it once or even multiple times. On-line bagging still allows to handle all ensemble members in parallel and independently from each other. In addition to that the data distribution is as independent as in batch operation because each ensemble member listens to the data stream according to its own Poisson distribution and needs no interaction or synchronization with its peers.

Boosting methods for batch processing build a sequence of models by weighting the individual samples in the data set according to the difficulty the former models face in predicting them, i.e. they are weighted according to the squared prediction error of the ensemble. Newly built models focus on the samples which are hard to predict and thus help to increase the overall prediction performance of the ensemble. The sample weights may be directly used in the learning algorithm which generates the model or may effect a resampling of the original data. The on-line version of boosting is similar to on-line bagging as it uses a Poisson distribution in order to simulate random resampling. However, the models in on-line boosting are still trained sequentially and the parameter of the Poisson distribution increases or decreases for each subsequently trained model according to the prediction performance of the currently trained model. This way, a sample correctly predicted by most of the models is less likely to be presented again, while incorrectly predicted samples are distributed more frequently as long as they appear to be hard to predict.

The main draw-back of the basic on-line bagging and boosting methods is their lack in handling time-variant behavior. There are several different approaches tackling this issue by introducing forgetting schemes similar to the methods review for implicit and explicit time-variance handling. A common feature of these approaches is to track the prediction performance of each ensemble member based on the most recent samples. The ensemble output is formed as the weighted combination of all ensemble members and poorly performing members are assigned a low weight while well performing ones are promoted by high weight. Another common issue in learning ensembles on-line is to guarantee the diversity of the ensemble members. On-line bagging and boosting contribute to this beneficial property by means of the Poisson distribution for sample selection, but there are further approaches for supporting ensemble diversity.

2.3.3.1 *Resetting and Replacement Schemes*

In [13] an ensemble of Hoeffding tree classifiers with limited sizes is proposed. The maximal size for the n-th ensemble member is 2^n and thus equally distributed on a logarithmic scale. Each tree may grow as it consumes the incoming samples but its maximal size in terms of number of nodes is fixed. Once a tree exceeds it maximal size it is either rigorously pruned or fully reset. As the frequency of this resetting mechanism scales inversely with the maximal size of the tree, the whole ensemble acts as a multi-scale sliding window approach. Therefore, it is inherently implicit with respect to its time-variance handling. The smaller trees adapt fast and they are able to keep track of changes while the larger trees perform better in stationary periods. This way, the ensemble ensures to contain at least some members suitable for the learning task at any time. The performance-based weight assignment is inversely proportional to the exponentially filtered squared prediction error.

The resetting mechanism for replacing ensemble members in [13] is time-based. A performance-based variant for random forests is presented in [161], where the worst performing ensemble members are regularly replaced by growing new trees. In general, [161] performs on-line bagging on an Extremely Randomized Forest [70], i.e. a set of test functions and thresholds in each splitting node is chosen randomly and on-line statistics about their performance refine the sets to single decisions. The performance-based replacement of ensemble members also takes into account the age of each ensemble member, as the candidates for replacement are selected accordingly. In a second step these candidates are randomly replaced and the probability of replacement increases according to their prediction error. In this scheme, older ensemble members are tested more frequently and among those even well performing ones are at risk of being deleted as the replacement test is stochastic in nature.

2.3.3.2 *Reoccuring Behavior detection*

The batch-incremental algorithm for classification in [53] handles an ensemble of equal members where each member is trained on a chunk of consecutive samples. Its most outstanding feature is the ability to detect and react on reoccurring behaviors, i.e. it may temporarily discard members from the ensemble by assigning it very little or no weight at all. In order to actually react to reappearing behaviors, the ensemble stores all the models it creates during learning. The authors explicitly recommend not to use any pruning of the ensemble member due age or performance in order to preserve the overall memory of the system for detecting reappearing behavior. A similar concept is proposed in [105] for segmentation and identification of time series. The approach keeps track of a sliding window in order to segment and classify based on the distribution of the windowed data. The classification in this approach is done in an unsupervised manner and thus, acts like a clustering algorithm. The fundamental distinction is the actual classification of each time series segment in contrast to the weighted combination in [53].

2.3.3.3 *Hybrid Approaches*

A more explicit time-variance handling for the ensemble members is proposed in [50]. The ensemble consists of Adaptive Model Rules (AMRules) and each such member uses a Page-Hinkely test on its individual prediction error in order to detect and delete outdated rules. The used test allows for an adjustable rate of change and thus is ignorant to slow drifts, but detects significant shifts. The approach exemplarily demonstrates the power of ensemble methods in handling different kinds of time-variant effects by delegating them to individual parts of the ensemble architecture. Here, the replacement of ensemble members takes care of adapting to shifts and the adaptation of each ensemble member as well as the ensemble combination rule allow to keep track of drifts. A similar concept is presented in [25] with the explicit focus on handling different kinds of time-variant behavior. Also very similar is [176] as it follows the same general structure, but measures the performance on a sliding window instead of using an exponential filter.

The replacement strategy in [50] already allows to detect changes in single rules and is able to act accordingly precise with respect to the knowledge which needs to be preserved as it is unaffected by time-variant behavior. This fine-grained information forgetting is also used in [93] for regression trees and accompanied by Option Trees which allow to efficiently handle ensembles of similar trees by tracking multiple splitting options and the corresponding subtrees in each splitting node.

The ensemble size is a crucial design parameter as long as the ensemble is not allowed to steadily grow like in [53]. The issue of adapting the ensemble size on-line is handled in [153]. The addition of new ensemble members is mapped to a statistical test which guarantees the overall accuracy improvement when including new members up to some confidence level. The same statistical argument applies for the removal of certain members and together these tests make up the automatic size adjustment for the ensemble. Due to the computational costs for evaluating the involved test statistics, they are checked regularly, but not in every single learning step. This computational burden of evaluating the most interesting statistics is a fundamental issue in on-line learning and also present in e.g. [93] with respect to the growing and replacement strategy. The overall effect of this regular checking is a latency introduced in the adaptation mechanisms of the algorithm which usually can be tuned to be negligible by increasing the computational demands or relying on prior knowledge about the expected rate of change with respect to sample generation frequency.

Another important aspect in maintaining the members of an ensemble is to preserve their dissimilarity. This aspect is easy to handle while creating new ensemble members, but it is difficult to ensure a certain degree of dissimilarity while training existing ensemble members. Every single ensemble member adapts to the same overall target function and thus all members inherently tend to become similar to each other during the learning process.

The whole field of ensemble methods is much broader than the thin slice of approaches related to time-variance handling which are reviewed here. Unfortunately, this interesting part of machine learning cannot be considered any further in this work, because it only touches the scope of this thesis due to its relation to learning in non-stationary environments.

2.4 ON-LINE AND INCREMENTAL LEARNING SYSTEMS

The methods and approaches reviewed so far already cover most of the building blocks of an on-line learning system, which essentially comprises an approximation structure and an on-line learning algorithm. The combination of these parts to a complete learning systems is reviewed here focusing on the mechanisms used to handle the non-linear parameter adaptation necessary in many approximation structures. Especially, the mechanisms to increase and decrease the expressiveness of the used approximation structure is concerned as these features are essential in non-stationary environments. The reviewed approaches are grouped according to the approximation structure they belong to and roughly follow the ordering in section 2.1. The selected literature barely covers the full variety of papers in the field, but points out the basic strategies associated with different approximation structures.

2.4.1 *Rigid Structure Approaches*

All approximation structures which suffer from the *curse of dimensionality* inherently form more or less rigid structures. This includes polynomials, grid-based look-up tables, B-Splines, Fourier Series and Wavelets as they either inherently make use of a grid for approximation or follow the tensor expansion. Nevertheless, all of these approximation structures contribute to the field of on-line learning systems and therefore are at least briefly mentioned here.

Although the approximation structures listed above lack in scalability, they contribute to low dimensional problems with limited resources in terms of computational and memory demands as in adaptive filter using polynomials [96] or B-splines [164]. Using adaptive filters as a standard tool for a wide range of problems with limited prior knowledge requires to use a flexible internal approximation structure. Polynomials provide such a structure and are fast to evaluate and efficient to store at the same time. Nevertheless, the polynomial degree as the main design parameter is hard to define.

A soft pruning of a polynomial model is achieved in [30] using Least Absolute Shrinkage and Selection Operator (LASSO) (compare [181]) regularization. This regularization forces the parameter vector to be sparse and thus, the most important basis functions are selected in order to build the approximation. This is similar to the weighted combination in ensembles and the LASSO approach has been further developed to handle groups of parameters [202] and sparse groups of parameters [64]. This regularization facilitates the choice of the polynomial degree as a design parameter because the system inherently aims at minimizing its expressiveness while balancing this goal against its prediction performance.

In [140] the great impact of grid-based look-up tables in the context of combustion engines is reviewed and linked to their usage as non-linear models and controllers. The merit of using grid-based approaches here is twofold. On the one hand, the evaluation of such systems is fast and learning is simplified due to a-priori fixed and optimized non-linear parameters β, i.e. the input segmentation contributes to reduce the learning problem complexity.

The time-frequency transformation approaches of Fourier Series [106] and Wavelets [18] suffer from the same issues as the other highly structured approaches like polynomials, GLTs and Spline, namely the *curse of dimensionality*. Thus, despite the fact that they offer powerful and unique features for approximation, their expressiveness adaptation capabilities are limited.

2.4.2 *RBF and Fuzzy Approaches*

RBF approaches and also Fuzzy Systems which make use of them in their antecedents offer a great variety of expressiveness adaptation methods. One system that needs to be mentioned here for its pioneering work is the Adaptive Neural Fuzzy Inference System (ANFIS) [94]. ANFIS has a fixed number of rules whose antecedents and consequent parts are updated simultaneously using gradient descent. This learning scheme highlights the strong connection between RBF, Fuzzy and neural network approaches, but it is only a historical starting point for the approaches considered here, as ANFIS does not change its overall expressiveness over time. The adaptation of the antecedents rather shifts the fixed amount of expressiveness locally on demand like polynomial approaches do. This kind of expressiveness adaptation is necessary for handling time-variant behavior, but it is not sufficient as it does not allow to react to overfitting or biased approximations in case of a changing target complexity. Therefore, the focus here is on techniques which change the overall expressiveness over time.

In general, these approaches decouple the handling of non-linear parameters of the RBF, i.e. position and span of the local representations, and the linear parameters of the local models. The non-linear parameters are estimated based on unsupervised learning techniques like clustering and density estimation. On a conceptual level, RBF networks are dealing with questions like when to add a new local model, how to merge two similar models and how to detect obsolete models in order to remove them. The removal of a model may be due to its age and long inactivity or due to low prediction performance. In the latter case the model is rather replaced by a new model or split into two than actually being removed.

The easiest way of deciding when to create a new local model is by testing the contribution of all other models to the current instance x_t. If no model is activated by more than a threshold parameter, a new model is created at the instance x_t with standard parameters for its shape. This technique is used in approaches like [98, 165, 99, 5, 3, 189, 128] with variations according to the used RBF kernels and the update of their non-linear parameters. In [201] the test for creating new local models also requires a sufficiently large prediction error for the current example.

The issue of scalability towards input dimensionality is highlighted in [189] in a perspective similar to the one in this work. It makes use of dimensionality reduction methods in order to build simple local models and even proposes an efficient indexing of the local models for handling trajectory based learning tasks. The other approaches differ in the way how they represent and treat the receptive fields of the local models. These representation issues lead to different further approaches like [155, 125] which more or less focus on dimensionality reduction and compact representations in order to keep the overall complexity of the approximation low. Merging two models requires them to have sufficient overlap with respect to their receptive fields and they need to be sufficiently similar with respect to their output.

A structured view on merging of local models is presented in [124]. Merging two models raises the issue of shaping the common receptive field of the involved local models according to the amount of samples each single model represents. The removal of local models is not only related to overfitting and complexity issues, but especially to forgetting in non-stationary environments as in [123]. These forgetting mechanisms comprise features based on age, contribution and performance of local models and include the concepts reviewed in adaptive learning algorithms and ensemble methods. Altogether, the merging and pruning or forgetting mechanisms help to prevent the on-line learning system from overfitting and support its ability to handle non-stationary environments. The most important design aspect of these kinds of systems is how they generate new models and what user-defined parameters, e.g. a contribution threshold, this growing strategy provides.

In general, a growing strategy which is eager for building new models may yield to complex merging and forgetting tasks and an overall increased computational and memory demand. On the other hand, a restrictive growing strategy inherently produces more biased approximations which usually threatens prediction performance. This tradeoff becomes even harder to balance in high dimensions due to the distribution aspect of the *curse of dimensionality* as the available samples become exponentially less dense and tends to populate only the boundary of some high-dimensional manifold.

2.4.3 *Neural Network Approaches*

The issues about high-dimensional problems are uniquely tackled by Extreme Learning Machines (ELM) [89, 90], which are a special variant of single layer MLPs with randomly initialized input weights. The random initialization of these non-linear parameters is fixed and thus excluded from learning. The resulting random features in the hidden layer are kernelized and based on random one-dimensional projections of the inputs. The on-line variants of ELMs [114, 87, 157, 116] stick to the random and fixed initialization of the input weights and adapt the linear output parameters using standard on-line learning algorithms like gradient descent and recursive least squares. The random feature generation is a statistically supported countermeasure for the *curse of dimensionality*, but it only yields the chance to find an appropriate set of features, which allows to learn well performing output weights. This raises the same conflict between prediction performance and overall computational and memory demands as for any other on-line learning system, but with a random sampling perspective onto the feature vector. The principle of *the more, the better* does not apply here due to universal approximation properties or in the sense of a Taylor Series approximation, but rather due to an increased probability of hitting valuable features. So, there is no guaranteed monotone increase in performance when adding more features, but it is likely. The random features also help to mitigate overfitting issues. As there is no systematic way of generating new features, there is little chance to hallucinate any structure in the learned samples due to noise. However, overfitting due to low sampling density in the beginning of the learning is still possible. One way to deal with this is to increase the number of features on-line as in [75], see [57] for batch operation. Another option is to make use of ensembles as in [110] with all related issues like performance and contribution tracking. A remarkable feature of ensembles of ELMs is the diversity of their members. Due to the random feature generation they rely on, this property nearly comes for free, even without on-line bagging or boosting. This beneficial property for an ensemble can even be pushed further by conditioning the probability distribution of the newly generated features based on the existing ones in the ensemble, which would yield some kind of random *tabu search*. For further readings about ELMs refer to [88, 28, 45, 86, 46].

2.4.4 *Tree-based Approaches*

The growing of trees is a standard technique which is shared by both batch and on-line learning. The popular Hoeffding trees [47] are made for classification with enhancements towards incremental forests in [67], but have already been adopted to regression as well, [92, 2, 93]. The growing mechanisms in trees are a second mean to tackle the *curse of dimensionality* as every splitting node only considers a binary decision. The resulting tree takes a limited sequence of these binary decisions in order to select one model to answer a query. This yields an information compression or reduction similar to the random features of the ELM but carries much more structure due to the hierarchy of the decisions. The corresponding models are located in this hierarchy of decisions, but they do not comply to a hierarchy themselves. When growing the tree, a model is replaced by a splitting node whose children are new models which are initialized using the former model. Thus, the models are refined with respect to the tree structure, but as the split model is removed from the tree, they do not carry the same hierarchy as the splitting decisions do. The split criterion is the fundamental design issue in growing model trees, similar to the local model generation in evolving fuzzy systems.

The unique feature of the tree structure is its hierarchy which allows to build statistics about the best local split for every model and to check whether a split actually improves the prediction performance. Tracking these statistics in high dimensions is one of the most challenging parts in learning a model tree on-line. According to the standard kd-tree structure, each splitting node needs to select an input dimension to use for the split and the actual splitting point in that input dimension. The necessary statistics to choose a valuable splitting criterion require a sufficiently large amount of samples and are computationally cumbersome. This is why they are evaluated only once in a dozen of learning steps or even less often, see [93]. Choosing a wrong splitting criterion either requires to prune the tree as the mistake becomes obvious or the resulting tree grows unnecessarily big, which increases sample, memory and computational demands. Thus, there is reason not to rush into splitting decisions, but this also slows down the growing of the tree and fosters biased approximations during the growing period. A calm growing strategy is a good countermeasure against overfitting as it is not likely for the involved statistics to be fooled by noise. The pruning of models in a tree roughly follows the same line as in evolving fuzzy systems. Replacements can be made at any stage of the tree and thus at any model granularity, while the merging of models is limited to siblings.

2.4.5 *Nearest Neighbor Approaches*

The data base management in nearest neighbor approaches is similar to member handling in ensembles. The fundamental questions are whether to include a new sample and how to detect obsolete samples in the data base. While the age of the samples is easy to track, their individual performance is less obvious to measure. The diversity of the data base is even more severe in nearest neighbor methods as for ensembles, as it represents the current hypothesis. In [172] these issues are grouped to three indicators for the usefulness of a sample, namely spatial relevance, temporal relevance and consistency. Together, these indicators keep the data base up to date with diverse and representative samples. The unique feature of nearest neighbor approaches is their ability to flexibly build local models which can even be selected using cross-validation. In [15] this *winner-takes-all* selection is further developed into a performance-based weighted average of local linear models. There are also approaches which deploy non-linear local models as in [95] for a soft sensor application.

The approximation power of nearest neighbor approaches stems from their flexibility to choose both the number of considered neighbors and the complexity of the local model and they are able to validate both on a second set of samples independent from the one for training the local model. In this regard, nearest neighbor techniques link on-line and batch processing by selecting a small subset of the streaming data for applying a standard or adopted batch processing method. But this flexibility in answering queries is only a true strength when operating on an actually representative data base.

2.4.6 *Support Vector Approaches*

While nearest neighbor approaches and on-line support vector regression share many aspects due to their sample-based learning nature, the situation for integrating samples in on-line SVR is more complex due to the underlying non-linear constrained optimization problem. The bookkeeping procedure in on-line SVR approaches like [129] is based on the incremental learning algorithm for classification in [31]. It keeps track of three different sets of vectors. The margin support vector set contains all vectors on the margin of the current approximation, the error support vector set gathers all vectors exceeding the margin and all vectors within the margin build the remaining set of ignored vectors. So, every new sample needs to be delivered to one of these sets. This may trigger additional movements between the sets depending on its actual target set in order to meet the Karush-Kuhn-Tucker condition. The resulting on-line learning behavior is similar to RLS with respect to robustness and noise reduction, thus for dealing with time-variant behavior additional relearning mechanics are required like in [118]. This relearning is triggered by a constantly decreasing prediction performance and replaces the whole model by a new one which is trained based on the most recent samples. There are also SVR-based ensembles methods [119] which focus on time-variant behavior and make use of a weighted combination with performance based weight estimation.

3 APPROACH

The approach this thesis presents in order to achieve the goal formulated in section 1.4 features a layered approximation structure by means of a multi resolution approximation. Each layer consists of a symmetric simplicial input segmentation equipped with a smooth interpolation scheme and an adaptive learning algorithm. The different layers of the approximation structure form a refinement hierarchy which relates this approach to multi resolution analysis methods like Fourier Series and Wavelets, but the presented approach here introduces a learning architecture which adapts the refinement layers one after another in an incremental fashion. Thus, each sample the resulting on-line learning system consumes is processed in an incremental manner with respect to the data stream it originates from and the information granularity extracted by learning. This chapter introduces the single parts of the approach and shows how they interfere in order to provide a full on-line learning system capable of dealing with evolving data streams while scaling linearly in regards to input dimensionality.

3.1 SCALABLE AND SMOOTH INTERPOLATION ON SYMMETRIC SIMPLEX-STRUCTURES

The overview of different approximation structures shows that a GLT with a smooth interpolation (GLT smooth) limits the scalability of current smooth approximation structures. On the other hand, 1st Order Simplicial B-Splines offer the best scalability for a continuous approximation. The approximation structure presented in this section combines both aspects in order to be scalable and smooth at the same time. This is achieved by combining a symmetric simplicial input segmentation with a special interpolation scheme which allows for a smooth transition between the heights associated with the vertexes of each simplex.

3.1.1 *Symmetric Simplicial Input Segmentation*

The input segmentation used here as a basis for the approximation structure is a symmetric variant of the *Freudenthal Triangulation* on a regular grid which dates back to [62] and has been used and even been independently rediscovered by many others, e.g. [108, 137]. The *Freudenthal Triangulation* is useful as it essentially defines a subdivision of an n-dimensional hypercube into $n!$ simplexes and the hy-

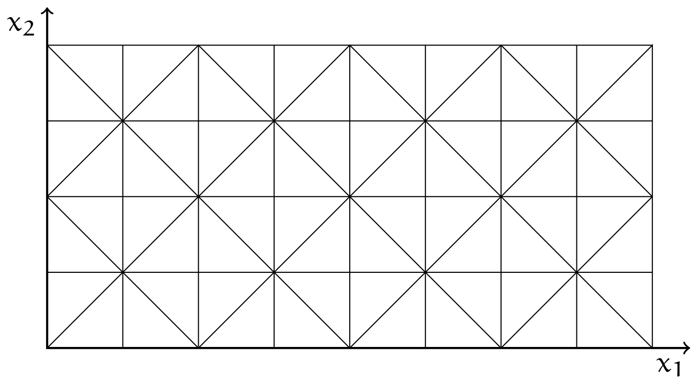


Figure 2: Schematic view of the symmetric simplicial input segmentation on a 8-by-4 grid in two dimensions.

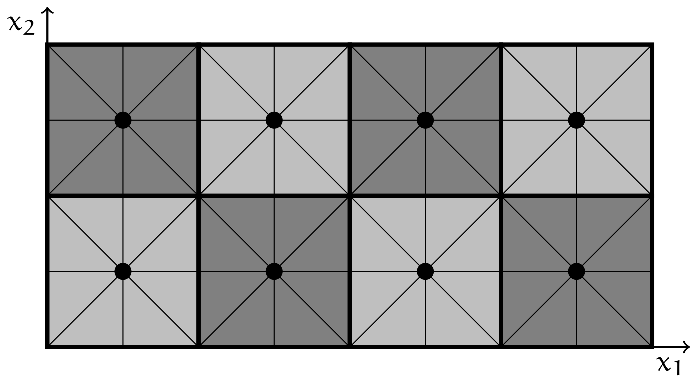


Figure 3: The symmetric simplicial input segmentation with highlighted hypercubic complexes and base nodes. The simplexes belonging to one complex are framed by a thick line and share the same gray tone. Each complex has a central base node marked by a dot. The complexes form a coarse hypercubical 4-b-2 gird over the underlying simplicial 8-by-4 grid.

percube is the only known regular geometric structure for covering the $\mathbb{R}^n$ with $n > 4$. Hence, in combination the simplicial subdivision of a regular grid in $\mathbb{R}^n$ yields a structured triangulation of $\mathbb{R}^n$ with simplexes of equal hyper-volume and similar shape. A schematic presentation of this triangulation in two dimensions is shown in Figure 2.

The main benefit of a symmetric triangulation is the possibility to restrict the considerations for finding a reasonable interpolation scheme to a single hypercube as all of the interpolation properties defined there extend directly to the whole grid by using appropriate symmetry operators, i.e. reflections at hypercube boundaries. This perspective naturally results in hypercubic complexes centered around one base node as shown in Figure 3.

The hypercubic complex structure allows for an easy indexing of the complexes and the simplexes within by relating them to their common base node. The base nodes are indexed by their grid coordinates. The simplexes are indexed using the grid nodes that belong to the simplex. These form a shortest path from the center of a complex to its outer vertexes along the edges of the hypercubes. This path can be uniquely defined by the integer sequence of the input dimensions on which the points on the path differ and the direction of the path. Figure 4 illustrates the indexing of the simplexes inside each complex and highlights the connection between the nodes that define the simplex and the integer sequence for indexing it. For all further considerations the hypercube with strictly positive indexes is used as it allows the same insights like every other hypercube due to the symmetry of the triangulation. Moreover, it is useful to look at three equivalent representations for the simplexes in the considered hypercube as illustrated in Figure 5 for the two dimensional case, namely:

1. Drawing in a coordinate system
2. Path in the edge-graph of a hypercube
3. Ordering of coordinate values

The first and visual representation is best for building intuition about the geometry in the symmetric simplicial triangulation, but it is limited to low dimensional cases. The second, graph-based representation allows a more topology orientated perspective onto the triangulation and slightly exceeds the dimensionality limitations of the drawing representation. Only the last and completely formal representation allows to handle issues in high dimensional spaces. Nevertheless, each representation has its own merits to demonstrate certain aspects of the interpolation considered in this section.

3.1.2 *Smooth Interpolation on Symmetric Simplicial Structures*

In one dimension there are many smooth interpolation methods for interpolating between two neighboring points, but expanding them to higher dimensions using the tensor product of many one dimensional interpolation functions leads to an exponential growth of the computational and memory demands for operating such an interpolation. The simplicial input segmentation introduced in section 3.1.1 offers the opportunity for an interpolation scheme which scales linearly regarding input dimensionality. The 1st Order Simplicial B-Splines define such an interpolation scheme as they linearly blend between the heights of the nodes of the simplex. In a n-dimensional space $\mathbb{R}^n$ there are only (n+1) nodes which define the simplex. This yields a simple simplex-related indexing of the basis functions ϕ_i that carry the interpolation scheme as shown in Figure 6. The indexing is easy to grasp using the path representation as every basis function of each simplex is indexed by their distance to the base node. Since every basis function ϕ_i belongs to a node, the position $p(i) \in [0, 1]^n$ of each such node is indexed accordingly.

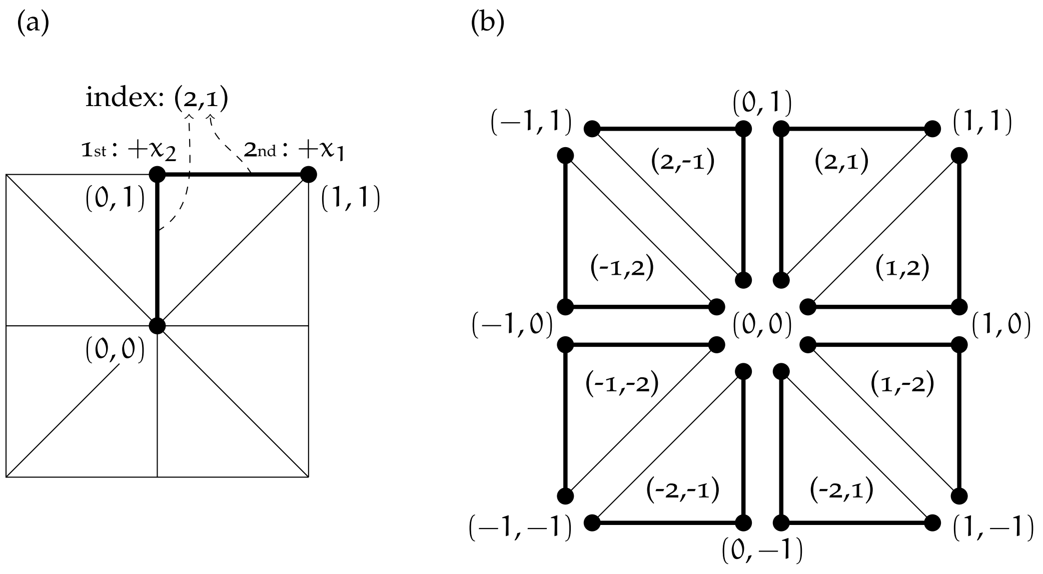


Figure 4: Illustration of the simplex indexing relative to the common base node centered at the origin $(0, 0)$ in the local coordinate system. Part (a) of the figure shows how the path marked in bold relates to the indexing. Each edge passed along the path follows one coordinate direction and the index gathers these signed directions ordered along the path. Part (b) of the figure shows all simplexes of a two dimensional hypercubic complex and their corresponding indexes.

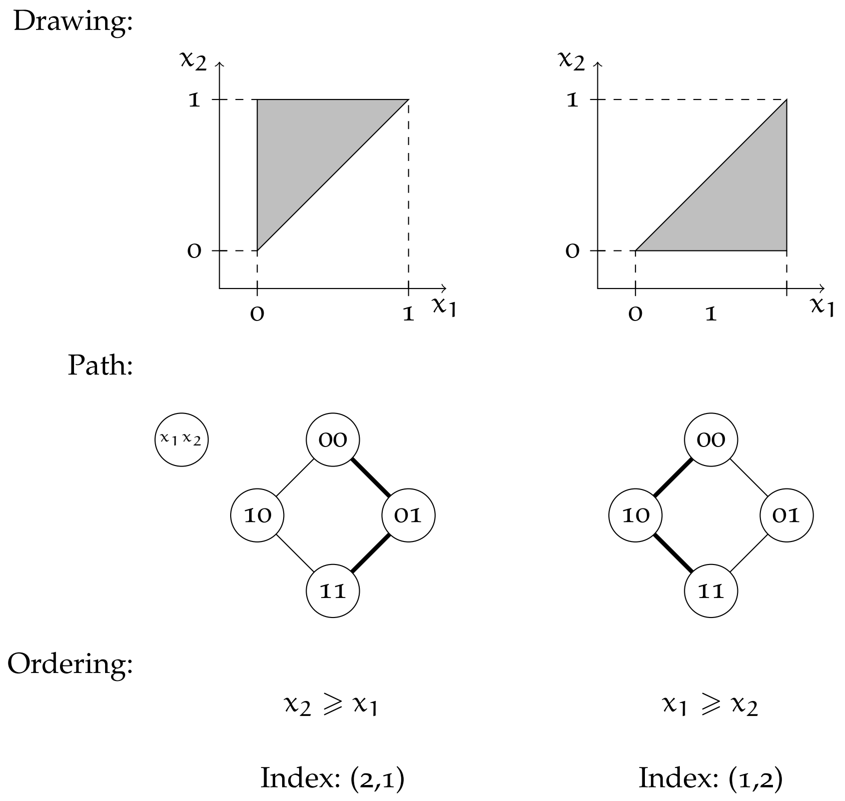


Figure 5: Here, the three representations of a simplex used in this work are illustrated for each kind of simplex in the two dimensional case. The scope is limited to the hypercube with strictly positive indexing without loss of generality. The different representations of the two simplexes indexed by (2,1) and (1,2) are ordered from top to bottom and show each simplex as a drawing, a path in a graph and an ordering of coordinate values.

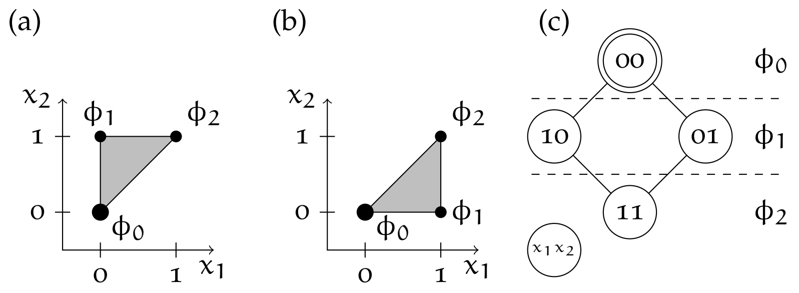


Figure 6: Illustration of the simplex-related indexing of the basis functions ϕ_i in the two graphical representations. The basis function ϕ_0 always belongs to the base node. The index i of the other basis functions ϕ_i equals the L_1- or Manhattan distance to the base node. In the graph representation in part (c), each kind of basis function ϕ_i belongs to one layer of the graph. The different layers of the graph are separated by dashed lines and each layer groups nodes of equal distance to the base node which is marked by a double circle.

The main drawback of the 1st Order Simplicial B-Spline approximation is its continuous but not smooth interpolation on a global scale as potentially every transition from one simplex to a neighboring one is not differentiable. This yields a widely sharp-edged approximation, especially in high dimensions. In order to obtain a smooth interpolation scheme, that still scales linearly with the dimension, the structure of the simplicial input segmentation needs to be considered to ensure a smooth transition between neighboring simplexes.

The development of this interpolation starts with the two dimensional case in the following subsection as it allows to visualize important properties and keeps the mathematical formulation simple. The insights of the two dimensional case are lifted to the general case of n-dimensional inputs and yield the Simplicial Tensor Product described in subsection 3.1.2.1 as well as the Simplicial Norm-Based Aggregation developed in subsection 3.1.2.2.

2-Dimensional Case

A smooth interpolation scheme defines basis functions ϕ_i for every node in the grid which weight the height of each corresponding node to form the approximation. These basis functions ϕ_i need to form a partition of unity over the whole input space, i.e. at each point in the input space they sum up to one, $\sum_i \phi_i(x) = 1 \forall x \in X \subset \mathbb{R}^n$. Moreover, the value of the basis functions ϕ_i needs to equal one at the node they belong to, it needs to vanish outside the simplexes that touch this node and all derivatives up to order d at the node and the boundary of the touching simplexes need to vanish, too. In order to meet these requirements it is important to look at the geometry of the support of each basis function ϕ_i.

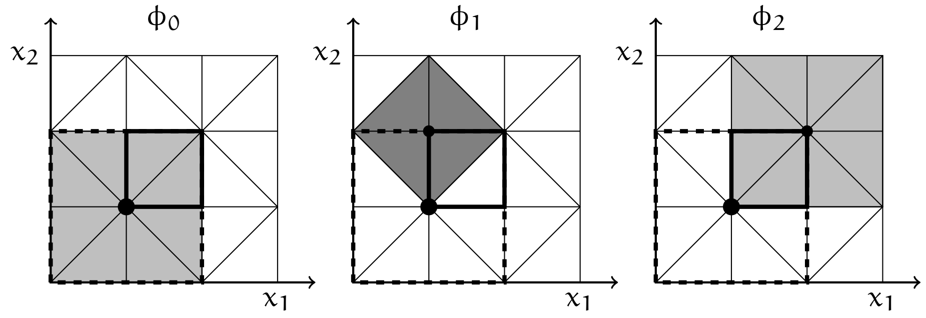


Figure 7: The figure shows the shape of the support of three basis functions of 2D Simplex highlighted in gray shades. The base node to which these basis functions refer is marked by a bold dot, the corresponding hypercubical complex is indicated by a dashed line and the considered hypercube is framed in bold. A smaller dot centered in the shaded areas marks the node the basis function belongs to.

Figure 7 shows the shape of the three basis functions of one simplex in two dimensions and highlights the hypercube for further consideration by a bold solid line. Over the whole grid, all basis functions show a squared support, two with axis-aligned squares and one with a diagonal-aligned square. Within the considered hypercube the shapes are actually different as the diagonal-aligned square becomes a triangle.

Without loss of generality, the considered hypercube is assumed to be the unit square $[0, 1]^2$. For a specific and easy example in two dimensions, it is further assumed to head for a global continuity of C^1, i.e. a continuously differentiable approximation. A one dimensional basis function yielding a globally C^1 behavior is given in equation (16). This function equals one at the origin with a vanishing slope and vanishes at one with a vanishing slope. The goal is to lift the properties of this simple polynomial function $g(x)$ to the two dimensional case while fitting the shape of each individual basis function $phi_i(x_1, x_2)$.

$$g(x) = 2x^3 - 3x^2 + 1 \tag{16}$$

In order to achieve this goal it is necessary to formalize the requirements for the basis functions ϕ_i from above into boundary conditions in the unit square. For the basis function ϕ_0 that belongs to the base node this is done in equations (17) to (20).

$$\phi_0(0,0) = 1 \tag{17}$$

$$\phi_0(1,x_2) = \phi_0(x_1,1) = 0 \tag{18}$$

$$\left.\frac{\partial\phi_0}{\partial x_1}\right|_{x_1=0} = \left.\frac{\partial\phi_0}{\partial x_1}\right|_{x_1=1} = 0 \tag{19}$$

$$\left.\frac{\partial\phi_0}{\partial x_2}\right|_{x_2=0} = \left.\frac{\partial\phi_0}{\partial x_2}\right|_{x_2=1} = 0 \tag{20}$$

This kind of basis function is easy to handle as the square shape of its support fits to the geometry of the tensor expansion and allows the standard product of one dimensional basis functions in order to form the two dimensional extension, see equation (21).

$$\phi_0(x_1,x_2) = g(x_1)g(x_2) \tag{21}$$

Equally simple to handle is the basis function ϕ_2. It shares the same shape of the support as the basis function ϕ_0 and has similar boundary conditions as shown in equations (22) to (25).

$$\phi_2(1,1) = 1 \tag{22}$$

$$\phi_2(0,x_2) = \phi_2(x_1,0) = 0 \tag{23}$$

$$\left.\frac{\partial\phi_2}{\partial x_1}\right|_{x_1=0} = \left.\frac{\partial\phi_2}{\partial x_1}\right|_{x_1=1} = 0 \tag{24}$$

$$\left.\frac{\partial\phi_2}{\partial x_2}\right|_{x_2=0} = \left.\frac{\partial\phi_2}{\partial x_2}\right|_{x_2=1} = 0 \tag{25}$$

The resulting tensor product for this basis function is shown in equation (26). It is very similar to the basis function ϕ_0 but uses mirrored coordinate variables.

$$\phi_0(x_1,x_2) = g(1-x_1)g(1-x_2) \tag{26}$$

The appropriate extension for the basis function ϕ_1 with a triangularly shaped support is less obvious as it needs to fit to the non-axis-parallel constraints shown in equations (27) to (30). The approach to meet these constraints is geometrically motivated as the problem stems from the geometry of the support of ϕ_1. The aim is to form an interpolation scheme similar to the tensor expansion $\phi(x_1)\phi(x_2)$ by projecting its resulting square shape to a triangle. This projection can be done by either mapping the $x_2 = 0$ edge to the main diagonal or the $x_1 = 1$ edge. The schematics of these two projections are illustrated in Figure 8 and they yield the two interpolation formulas $\phi((1-x_1)/(1-x_2))\phi(x_2)$ and $\phi(1-x_1)\phi(x_2/x_1)$. None of both formulas meets the requirements as they either fail to ensure a vanishing derivative along x_1 at $x_1 = 0$ or along x_2 at $x_2 = 1$.

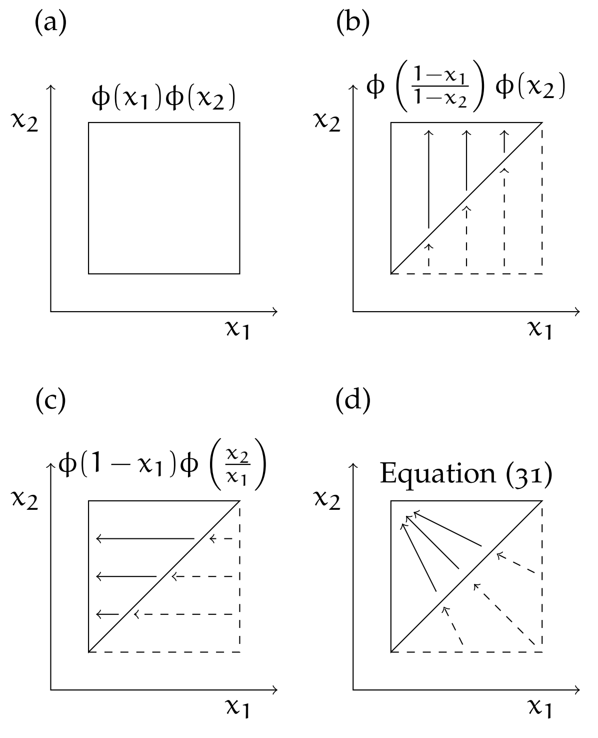

(a)
$\phi(x_1)\phi(x_2)$
x_2
x_1
(b)
$\phi\left(\frac{1-x_1}{1-x_2}\right)\phi(x_2)$
x_2
x_1
(c)
$\phi(1-x_1)\phi\left(\frac{x_2}{x_1}\right)$
x_2
x_1
(d)
Equation (31)
x_2
x_1

Figure 8: Part (a) of the figure shows the regular tensor product and the corresponding square geometry. Parts (b) to (d) illustrate the principle behavior of different projections from the square to a triangle. The arrows show the direction of the projection. In parts (b) and (c) the projection is axis parallel. The solid arrows also point to the edge that does not fit the boundary conditions required for a globally smooth interpolation. Part (d) shows the change of the projection direction depending on the position inside the square. This varying projection direction allows for a globally smooth interpolation.

$$\phi_1(0,1) = 1 \tag{27}$$

$$\phi_1|_{x_1=x_2} = 0 \tag{28}$$

$$\left.\frac{\partial\phi_{tri}}{\partial x_1}\right|_{x_1=0} = \left.\frac{\partial\phi_{tri}}{\partial x_1}\right|_{x_1=x_2} = 0 \tag{29}$$

$$\left.\frac{\partial\phi_{tri}}{\partial x_2}\right|_{x_2=1} = \left.\frac{\partial\phi_{tri}}{\partial x_2}\right|_{x_1=x_2} = 0 \tag{30}$$

Combining both projections as shown in equation (31) yields a blending from one projection to the other smoothed by the one dimensional interpolation function, here $g(x)$. This way, each projection is active only for the one part of the boundary where the corresponding derivative actually vanishes. Due to the reuse of the interpolation function $g(x)$ in the blending scheme of ϕ_0, the basis function inherits the smoothness properties of the one dimensional function $g(x)$. The formula for ϕ_1 is not defined in $(0,1)$ but can be smoothly continued by $\phi_1(0,1) = 1$. Thus, it lifts the properties of the one dimensional basis function to the two dimensional case, like the tensor product, and fits the triangular shape of the support of the basis function.

$$\begin{aligned} \phi_1(x_1,x_2) &= \frac{g(x_1)}{g(x_1)+g(1-x_2)} \cdot g(x_1) g\left(\frac{1-x_2}{1-x_1}\right) \\ &+ \underbrace{\frac{g(1-x_2)}{g(x_1)+g(1-x_2)}}_{\text{Blending}} \cdot \underbrace{g\left(\frac{x_1}{x_2}\right) g(1-x_2)}_{\text{Projection}} \end{aligned} \tag{31}$$

Unlike the tensor product, the interpolation in equation (31) involves a sum over multiple products and thus, it is not obvious if or how it scales to higher dimensions. But this blending and projection structure is essential for the envisioned interpolation scheme and builds the backbone for the multidimensional generalization. The simple two dimensional example allows to illustrate certain aspects graphically and already carries all concepts necessary for the general n-dimensional case. However, this general situation requires a more formal consideration as described in the following subsection.

3.1.2.1 *Simplicial Tensor Product Aggregation*

The interpolation scheme in equation (31) is based on the knowledge about the shape of the basis function ϕ_1, which is easy to grasp visually in two dimensions. For the general n-dimensional case, it is important to develop a formal representation for the shape of the basis functions. The support of each basis function is formed by the union of the simplexes touching the node this basis function belongs to. Again, it is sufficient to look at a single hypercube and consider the simplexes in there as the rest of the geometry is formed by symmetry operations.

Fortunately, there are only $\lceil (n+1)/2 \rceil$ different shapes for the support of the basis functions ϕ_i in n dimensions. The layer perspective onto the hypercubical edge-graph in Figure 4 for the simplex indexing already shows that all basis functions in the same layer share the same shape and index. The shapes are symmetric with respect to the middle of the graph, i.e. ϕ_0 always has the same shape as ϕ_n, ϕ_1 the same as ϕ_{n-1} and so on. This is due to the symmetry of the triangulation.

The simplexes belonging to the support of a basis function ϕ_i are the ones that touch its corresponding node. In the path representation these are all paths that touch or pass this node. This perspective is shown in Figure 9 for a four dimensional hypercube and the node 1001 as an illustrating example. The corresponding orderings of the path highlighted in the figure are depicted in equations (32) to (35).

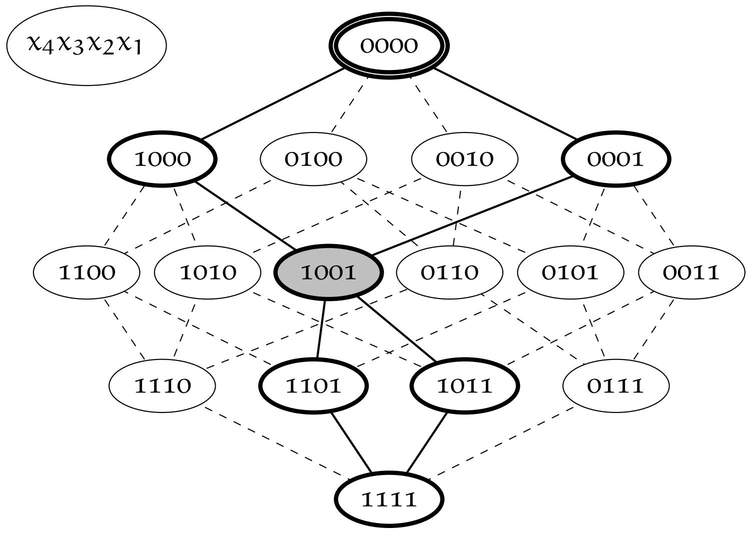


Figure 9: This graph represents the edge connections of a four dimensional hypercube. One node is picked in order to highlight all paths passing through this node. The picked node 1001 is marked in gray and by a bold ellipse. All paths passing this node are marked by a bold lines; All other connections are represented by dashed lines. Other nodes visited by the highlighted paths are marked in bold as well.

$$x_4 \geqslant x_1 \geqslant x_3 \geqslant x_2 \tag{32}$$

$$x_4 \geqslant x_1 \geqslant x_2 \geqslant x_3 \tag{33}$$

$$x_1 \geqslant x_4 \geqslant x_3 \geqslant x_2 \tag{34}$$

$$x_1 \geqslant x_4 \geqslant x_2 \geqslant x_3 \tag{35}$$

The shape of the support of the basis function that belongs to the node 1001 is defined by the union of the simplexes defined in equations (32) to (35) and due to the inner structure of this set of inequalities their union can be condensed into one inequality as shown in equation (36).

$$\min(x_1, x_4) \geqslant \max(x_2, x_3) \tag{36}$$

The inequality in equation (36) directly applies for higher dimension as it essentially states that the minimum of one group of coordinate values needs to be greater or equal to the remaining ones. One of these groups may be empty which yields the unconstrained case which corresponds to the square geometry. The most remarkable thing about the geometry of the different basis functions ϕ_i is their generalized triangular shape inside the hypercube. This generalized triangle is rectangular with hypercubical cathetes and the dimensions of the hypercubical cathetus sum up to the input dimensionality. The generalization of equation (36) to n-dimensions is shown in equation (37), where A_i is the basis function specific index set which contains all coordinate indexes with non-zero node position values, i.e. $A_i = \{k \in \{1, ..., n\} | p(i)_k = 1\}$.

$$\min\{x_k | k \in A_i\} \geqslant \max\{x_k | k \notin A_i\} \tag{37}$$

The above geometrical consideration focuses the whole hypercube. In order to define the basis functions ϕ_0, ϕ_1, ..., ϕ_n in n-dimensions the situation is even simpler as it is sufficient to focus on the single simplex the current evaluation point x_t belongs to. Within this simplex it is only important to determine which facet of the simplex is constrained and what constraints to apply there. Every facet of this simplex leading to another simplex within the support of the same basis function poses no boundary conditions as it is no boundary of the support. Facets of the simplex that belong to the boundary of the support can either belong to the hypercubical shape or a triangular one. The part of the boundary which belongs to a hypercubical shape allows for a tensor product, while the triangular shaped part of the support requires a blending similar to equation (31), but there is at most one such facet for every simplex. Hence, it is sufficient to find this facet in order to respect the full geometry of the basis function while focusing on one simplex.

Fortunately, this facet is easy to find in the ordering representation of a simplex. In general, the constraints imposed by every ordering $\pi \in S(n)$ has the structure shown in equation (38) and each ordering π uniquely defines a simplex $\Delta_\pi \subset [0, 1]^n$.

$$x_{\pi(1)} \geqslant x_{\pi(2)} \geqslant ... \geqslant x_{\pi(n)} \tag{38}$$

As observed in the two dimensional example, the basis functions $\phi_0 = \prod_{j=1}^{n} g(x_j)$ and $\phi_n = \prod_{j=1}^{n} g(1 - x_j)$ have square support without any triangular constraints and follow the pure tensor expansion. For every other basis function ϕ_i with $i = 2, ..., n - 1$ the min-max-structure in equation (36) reduces to a single inequality for each simplex Δ_π as shown in equation (39) and the resulting generalized interpolation scheme in equation (40).

$$x_{\pi(i)} \geqslant x_{\pi(i+1)} \tag{39}$$

$$\begin{aligned} &\phi_i(x) \qquad\qquad\qquad\qquad\qquad (40)\\ = \;& \frac{g(x_{\pi(i)})}{g(x_{\pi(i)}) + g(1 - x_{\pi(i+1)})} \prod_{j=1}^{i} g(x_{\pi(j)}) \prod_{j=i+1}^{n} g(\frac{1 - x_{\pi(j)}}{1 - x_{\pi(i)}}) \\ + \;& \frac{g(1 - x_{\pi(i+1)})}{g(x_{\pi(i)}) + g(1 - x_{\pi(i+1)})} \prod_{j=1}^{i} g(\frac{x_{\pi(j)}}{x_{\pi(i+1)}}) \prod_{j=i+1}^{n} g(1 - x_{\pi(j)}) \end{aligned}$$

This *Simplicial Tensor Product* interpolation scheme is one central result in this thesis as it allows to lift the properties of one dimensional interpolation functions like $g(x)$ to the n-dimensional case in a symmetric simplicial input segmentation. It scales linearly with respect to the number of parameters necessary for its evaluation due to the geometry of the simplex it refers to. The only drawback of this interpolation is the computational demand for calculating all basis functions ϕ_i as this demand scales quadratic in the number of input dimensions.

Another issue of the Simplicial Tensor Product is the partition of unity as this constraint is not fulfilled globally in dimensions greater than one, $n > 1$. One direct countermeasure is to normalize the basis functions ϕ_i as in equation (41).

$$\phi_{norm,i}(\mathbf{x}) \quad = \quad \frac{\phi_i(\mathbf{x})}{\sum_{j=0}^{n} \phi_j(\mathbf{x})} \tag{41}$$

The reason for this lack of partition of unity is the use of the tensor product to handle the unconstrained input dimension. The square geometry of this tensor product conflicts with the overall generalized triangular shape of the basis functions and covers some regions in the input space only incompletely. The normalization fills these regions but makes the formal analysis of the resulting approximation cumbersome. Nevertheless, the interpolation scheme in equation (40) has its merits as the tensor product potentially allows to define a desired continuity along every single input dimension separately. But this requires profound prior knowledge. Another way to get the partition of unity is to replace the tensor product by a more appropriate aggregation as worked out in the following subsection.

3.1.2.2 *Simplicial Norm-Based Aggregation*

The tensor product is one way to aggregate the unconstrained input dimensions in each of the two groups of indexes in equation (37) and it fails to form a partition of unity as the value of the corresponding basis functions ϕ_i rapidly decays along diagonal direction while following the desired interpolation $g(x)$ along each coordinate axis. Radial Basis Functions do not distinguish between different directions as they reduce multiple input dimensions to one scalar value using an appropriate similarity measure. Sophisticated similarity measures may treat every input dimension in its own right, but in general the reduction idea is able to be ignorant of directions and thus mitigate the shortcomings of the tensor product.

The goal here is to define a similarity measure, i.e. a reduction, that respects the geometry of the already know basis function support and allows for a smooth interpolation by lifting the properties of a one dimensional function $g(x)$ to the n-dimensional case. The basis for this development is a different representation of equation (37) as shown in equation (42). It highlights the triangular constraint rather than the ordering of the coordinate values and the right side of the inequality provides a direct distance measure between the center of the basis function ϕ_i and an evaluation point $\mathbf{x}$.

$$1 \geqslant \max\{(1-x_k) : k \in A_i\} + \max\{x_k : k \notin A_i\} \tag{42}$$

For a more compact and thus clear representation, it is useful to define individual vectors that contain the parts of the complete evaluation point $\mathbf{x}$ which belong to A_i, i.e. $a_i = (\nu_i \in \mathbf{x} : i \in A_i)$, and those which are not in A_i, i.e. $\overline{a}_i = (\nu_i \in x : i \notin A_i)$. This notation allows for the norm-based representation of the generalized triangular constraint shown in equation (43). Like in the two dimensional case, this inequality is related to the structure of the interpolation scheme defined in equation (31) which comprises of blending and projecting.

$$1 \geqslant ||1 - a_i||_\infty + ||\overline{a}_i||_\infty \tag{43}$$

Here, the two norms $||1 - a_i||_\infty$ and $||\overline{a}_i||_\infty$ correspond to the single input variable x_1 and x_2 from the two dimensional example. But the infinity norm is not suited for smooth interpolation schemes as it simply reproduces the square shape that is related to the unconstrained input coordinates. This square shape results in edged contour lines of the corresponding basis functions ϕ_i, but a smooth interpolation scheme requires smooth contour lines of its basis functions.

Hence, the contour lines of the basis functions $\boldsymbol{\phi}$ need to be smooth, respect the non-smooth boundary and form a smooth transition from the boundary to the center. One way to meet these requirements is to use the variable norm $||\,.\,||_\nu$ defined in equation (44). This norm yields the Euclidean norm if the maximum norm of $\mathbf{x}$ equals zero and smoothly transits to the maximum norm as the maximum norm of x approaches one. If the maximum norm of $\mathbf{x}$ equals one, the corresponding basis function has no influence at all and the overall approximation is well defined.

$$||\mathbf{x}||_\nu = ||\mathbf{x}||_{\frac{2}{(1-||x||_\infty)^2}} \tag{44}$$

The contour lines this variable norm produces follow the well-known shapes of circles in different L_p norms. They start with the actual euclidean circle and approach the square shape of the maximum norm. Using this variable norm to aggregate the unconstrained coordinate values yields the interpolation scheme defined in equations (45) to (47).

$$\begin{aligned}
\phi_0 &= g(\|\mathbf{x}\|_\nu) && (45)\\
\phi_n &= g(\|\mathbf{1}-\mathbf{x}\|_\nu) && (46)\\
&\text{for} \quad i = 2, \dots, n-1 : \\
\phi_i(\mathbf{x}) &= (1-\gamma)\cdot g\left(\|\overline{a}_i\|_\nu\right) g\left(\frac{\|1-a_i\|_\nu}{1-\|\overline{a}_i\|_\nu}\right) && (47)\\
&\quad +\gamma \cdot g\left(\|1-a_i\|_\nu\right) g\left(\frac{\|\overline{a}_i\|_\nu}{1-\|1-a_i\|_\nu}\right)\\
\gamma &= \frac{g(1-\|1-a_i\|_\nu)}{g(1-\|\overline{a}_i\|_\nu)+g(1-\|1-a_i\|_\nu)}
\end{aligned}$$

This Simplicial Norm-based Aggregation uses a variable norm to define smooth contour lines and respects the geometry of the support of the basis functions. It yields a partition of unity and produces the same principle interpolation behavior along coordinate axes and diagonals. This behavior is defined by the underlying interpolation function $g(x)$ whose properties are lifted to the n-dimensional case in a uniform way compared to the shaped approximation of the Simplicial Tensor Product Expansion. But this norm based approach does not allow to treat individual input dimensions differently with respect to their continuity. It is computationally more demanding than the Simplicial Tensor Product Aggregation as the norm calculation of arbitrary L_p-norm is costly and not as well supported by hardware in common CPUs as pure multiplication. It also scales quadratic in the computational demand with respect to the number of input dimensions. Thus, both interpolation schemes are very similar to each other with respect to lifting one-dimensional functions to high-dimensional simplicial grids, but differ in central aspects like partition of unity and computational demands.

3.2 SIMPLICIAL REFINEMENT LAYERS

The approximation structure developed in section 3.1 builds upon a symmetric simplicial input segmentation which is equipped with two different interpolation schemes, the Simplicial Tensor Product Aggregation and the Simplicial Norm-Based Aggregation. Both interpolation schemes lift the properties of a chosen one dimensional interpolation function $g(x)$ to the n-dimensional case. In terms of approximation properties this structure suffers from all issues the GLT is prone to, despite the exponential scalability as it offers a linear scaling for the memory access and a quadratic scaling for the computational demand.

The problem of approximation artifacts in high resolution grids due to the forced vanishing slopes at the grid nodes and the bound-

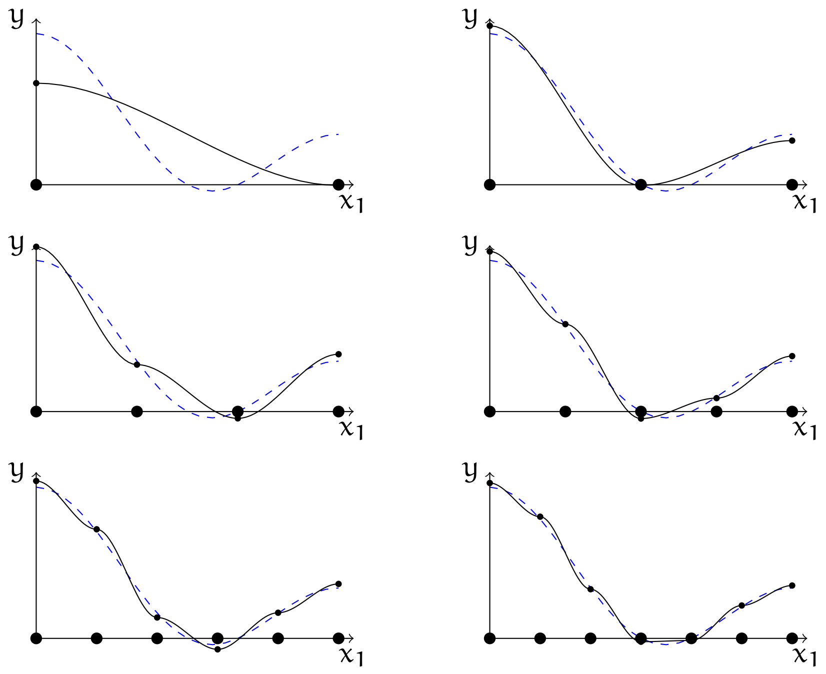


Figure 10: Illustration of the approximation artifacts due to the vanishing slope at grid nodes, here in one dimension. The interpolation uses the polynomial function $g(x)$ from equation (16). The grid resolution increases from two nodes in the top left plot to seven nodes in the bottom right one.

ary of the support of each basis function is even more severe, because it even affects the simplex boundaries and the input segmentation divides each hypercube into $n!$ simplexes. In Figure 10 the problem of approximation artifacts is illustrated in the one dimensional case for different grid resolutions. For low resolution grids the approximation is biased but in general smooth. By increasing the grid resolution, the approximation gets less biased, but starts to oscillate around the target function while forming a stair-like overall shape. This stair-like behavior does not affect the formal smoothness properties of the approximation structure, but contradicts the original purpose of building smooth approximations as the learned derivatives become less and less meaningful. An option to get back to reasonable derivatives is to introduce local linear models, i.e. preventing the derivatives from vanishing at nodes and boundaries. Although this approach is feasible in general, it is not further considered here as it affects the MAR of the whole system and would increase the MAR to quadratic scalability which contradicts the goal of this thesis.

The pesky approximation artifacts for high resolution grids and the poor approximation capabilities of low resolution grids give rise to the idea of a multi resolution approach in order to combine the strengths of both while canceling their shortcomings. This requires a cooperation of different layers in the multi resolution approach such

$$\begin{tikzcd}
\text{Layer } l_0: & l_{0,t-1} \arrow[r, "{(x_t, y_t)}"] & l_{0,t} \arrow[dr, dashed] & \\
& & & \delta_0 = y_t - l_{0,t}(x_t) \arrow[dl, dashed] \\
\text{Layer } l_1: & l_{1,t-1} \arrow[r, "{(x_t, \delta_0)}"] & l_{1,t} \arrow[dr, dashed] & \\
& & & \delta_1 = \delta_0 - l_{1,t}(x_t) \arrow[dl, dashed] \\
\vdots & & \vdots & \vdots \\
& & & \delta_{r-1} = \delta_{r-2} - l_{r-1,t}(x_t) \arrow[dl, dashed] \\
\text{Layer } l_r: & l_{r,t-1} \arrow[r, "{(x_t, \delta_{r-1})}"] & l_{r,t} &
\end{tikzcd}$$

Figure 11: Illustration of the proposed learning architecture for the refinement layers l_r and the resulting information propagation across different layers. The learning starts by updating the base layer l_0 and calculating the residual δ_0. All subsequent layers l_r use the residual δ_{r-1} of their preceding layer l_{r-1} as learning target. The actual learning of each layer l_r is indicated by solid lines and the information propagation across the layers is highlighted using dashed lines.

that the higher layers mitigate the approximation artifacts of the lower ones, while the lower layers account for the missing approximation capabilities of the higher ones.

This section develops the layer structure which embeds the simplicial approximation from section 3.1 in order to reduce its approximation artifacts while preserving its scalability properties. The development starts in the following subsection 3.2.1 by introducing the learning architecture for handling the different layers. It proceeds in subsection 3.2.2 by introducing refinement strategies which contribute to the artifact mitigation and finally handles the information propagation across the layers in subsection 3.2.3. The learning architecture perspective here is focused on the additive composition of simplicial approximation structures, for a more general perspective onto additive models please refer to [79].

3.2.1 *Learning Architecture*

The basic idea of cooperation between layers of different resolution is present in Wavelets. Fourier Series and Tschebyschow Polynomials employ similar refinement approaches, but do not provide a local approximation in high resolution components. In all of these approximation structures, the cooperation of different resolution components is widely transparent to on-line learning algorithms as the corresponding basis functions in each structure are orthonormal to each other. Nevertheless, even in Wavelets an explicit coordination between the different layers is essential as changes in a higher layer have a wide or even global impact onto the approximation. This can easily destroy already learned knowledge and thus, delay or even threaten the convergence of learning at all. Moreover, the interpolation schemes defined in this thesis for the symmetrical simplicial input segmentation in general do not form an orthonormal basis and thus require an explicit handling of the cooperation across multiple layers to some degree, because a non-orthonormal set of basis functions yields suboptimally conditioned parameter updates. This requires explicit handling by means of a learning architecture in order to avoid poor parameter updates during learning due to error amplification and fatal forgetting.

In general, the envisioned layer structure is linear in the parameters like the underlying simplicial approximation structure. Therefore, the different layers conceptually only group certain basis functions in order to form reasonable abstractions. The overall approximation structure of the layer architecture is depicted in equation (48) with each layer l_r having its own set of basis functions $\phi_{r,i}$ and parameters $\alpha_{r,i}$, distinguished by the layer index r.

$$L(\mathbf{x}) = \sum_{r=0}^{k_{max}} l_r(\mathbf{x}) \tag{48}$$

The learning architecture for this layer structure needs to respect and ensure the refinement property of consecutive approximation layers. In order to do so, it follows a top-down approach for the information propagation across subsequent layers. The base layer l_0 is the only one to receive the actual learning samples $(\mathbf{x}_t, y_t)$. All subsequent layers learn the residuals of their predecessor. The whole learning architecture is illustrated in Figure 11. The main benefit of this kind of learning architecture is a forced decoupling in terms of detail approximation between the different layers. Subsequent layers are trained to eliminate the flaws of their predecessors no matter whether they are caused by a biased approximation due to low expressiveness or by an approximation artifact. This ensures a differential encoding of the target function over all layers whose combination fits to the additive layer structure in equation (48).

The fundamental issue of this learning architecture originates from the samples $(\mathbf{x}_t, \delta_r)$ which interface subsequent refinement layers $l_{r>1}$. The target functions represented by these samples are inherently nonstationary as any change in a higher layer affects the residuals a subsequent layer produces. Thus, the corresponding residual target functions interfacing subsequent layers are inherently time-variant. This results in a strictly sequential convergence of the single layers, which drastically increases the necessary amount of training data and thus delays the overall convergence. Even worse is the expected poor approximation quality while converging to a stationary solution due to rare meaningful samples in the lower layers. All in all, the additive layer structure and the corresponding learning architecture are superior with respect to artifact mitigation as well as layer cooperation and also provide useful insides about the decomposition of the target function, but they also render the learning widely infeasible as the actual cooperation of the refinement layers is hindered by time-variant residual targets which continuously render the already learned knowledge invalid as the output of a preceding layer changes.

All of this makes an appropriate information propagation about the adaptation of each layer inevitable. This way, the target functions for the non-base layers become stationary again and all layers converge concurrently. Hence, one additional requirement for the refinement strategy of the layer architecture is to support an information propagation that allows for a decoupling in terms of approximation details while keeping the target functions of the lower levels stationary.

Unfortunately, this additional requirement for the refinement strategy conflicts with the goal of mitigating the approximation artifacts by using a layer architecture. For a good mitigation of the approximation artifacts all nodes across all layers need to have unique positions as this way no vanishing slope of a single node has a strong influence on the resulting approximation. The information propagation is best supported by aligned grids which share as many node positions as possible because the alignment yields a strong connection between nodes in subsequent layers and thus fosters an easy and efficient adjustment propagation across the layers, i.e. all nodes which share the same position as their corresponding nodes in the preceding layer only need to refer to this single node.

As the conflict between these two requirements is fundamental, two different refinement strategies are presented next. An aligned one which fosters information propagation across subsequent layers and a shifted one which focuses on mitigating the approximation artifacts of each layer in the overall approximation. These two refinement strategies give rise to individual approximation structures and further allow to combine them as a third option.

3.2.2 *Refinement Strategies*

A refinement strategy for the layered learning architecture has to meet three main requirements. These start by an increased expressiveness in subsequent layers, cover the information propagation across different layers as well as the mitigation of approximation artifacts and even touch the Memory Access Ratio as the refinement strategy should induce as little additional memory access as possible. In terms of a general target function decomposition into different detail levels, each layer should provide its expressiveness equally distributed across the whole input space.

The refinement strategies presented here perform a doubling of the grid resolution in subsequent layers. This is a compromise between a reasonable decomposition into sufficiently dissimilar approximation parts and the total number of layers. A more slowly increasing grid resolution would yield a finer decomposition spectrum at the cost of maintaining, evaluating and learning additional layers. A faster growth of the expressiveness, e.g. by tripling the grid resolution in each layer, would reduce the total number of layers which are necessary to approximate a certain functions with a given accuracy. But it would also skip certain decomposition components and thus, yield a less informative approximation spectrum. A reasonably scaled spectral decomposition yields strong features for detecting irrelevant parts of the approximation and helps to separate signal from noise.

Doubling the grid resolution affects all input dimensions equally and in a global manner. This assumes an equal relevance and span of all input variables x_i. The span of each input variable x_i is either given by prior knowledge or the input is normalized according to statistics about its mean and standard deviation. There are further options for handling dissimilar inputs with unknown span by exploiting the geometry of the refinement layers, but these are not considered here in order to focus on the refinement strategy.

The central refinement strategy introduced here combines two simplicial input segmentations which are shifted relatively to each other. This combined refinement strategy is described in subsection 3.2.2.3 and decomposes into an aligned and a shifted part which are introduced separately in the subsequent subsections. The fundamentals of these two refinement strategies directly apply to the combined one. Further, each partial refinement strategy allows for on-line learning systems on its own. The aligned and shifted refinement strategies are simpler than the combine one which comprises both. This allows the partial strategies to trade restricted expressiveness for reduced computational and memory demands.

All refinement strategies are related to different information propagations and encodings, respectively. They share the same principles, but differ with respect to effort and effectiveness. Subsection 3.2.3 presents the information propagation and encoding for refinement strategies which are introduced in the following subsections.

3.2.2.1 *Aligned Refinement Strategy*

The aligned refinement strategy presented in this subsection focuses on the information propagation across subsequent layers. Its

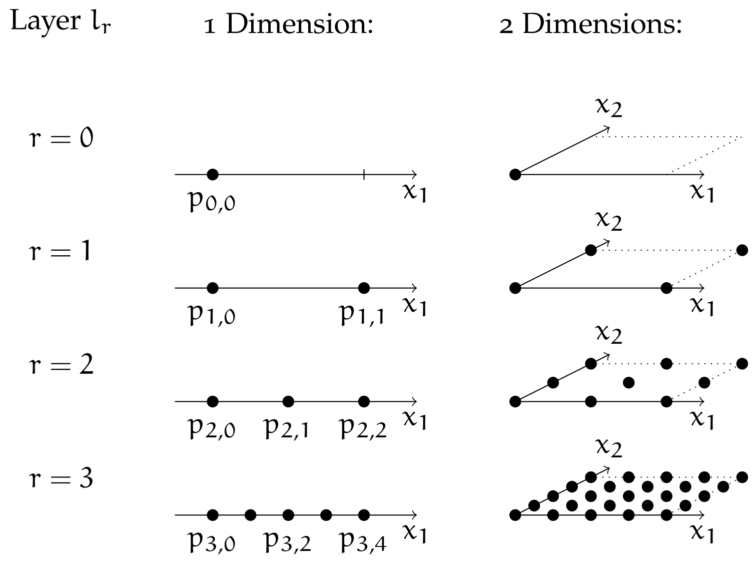


Figure 12: Illustration of the principle structure of the aligned refinement strategy and the resulting layers l_r in dimensions one and two. The nodes in each layer form a regular grid. Every node from an upper layer is replicated in the lower layers and is accompanied by additional nodes for building a grid with doubled resolution. The indexing of the node positions $p_{r,j}$ is shown exemplarily in the one dimensional case.

structure is regular in order to minimize the memory access and to yield a uniform distribution of the approximation capabilities. The focus on a proper information propagation fosters stationary residual target functions and thus, the data efficiency of the whole layer structure as an on-line learning system. The resulting aligned refinement structure for on-line learning follows the corresponding strategy in order to build refinement layers during the learning progress.

The basic principle of the aligned refinement strategy is illustrated in Figure 12. It starts with a single node for the base layer l_0 and continues with a first refinement layer l_1 which covers the entire input space with a single hypercube and the corresponding simplicial segmentation. Each subsequent refinement layer $l_2, \ldots, l_r$ doubles the grid resolution of its predecessor by replicating the nodes of the preceding layer and adding new ones centered between the nodes of the former one in order to fill the finer regular grid. This way, most positions of nodes in subsequent layers coincide and the different grids across the layers are aligned to each other.

The refinement strategy assumes the input space $X \in \mathbb{R}^n$ to be a hyperrectangle. In one dimension, i.e. $n = 1$, this is an interval $X = [x_{min}, x_{max}]$ and in higher dimensions this generalizes to $X = \times_{i=1}^{n} [x_{min,i}, x_{max,i}]$. Without loss of generality the input space here is considered to be a hypercube in order to define the refinement strategy in a principle manner. The one dimensional position $p_{r,j}$ of each

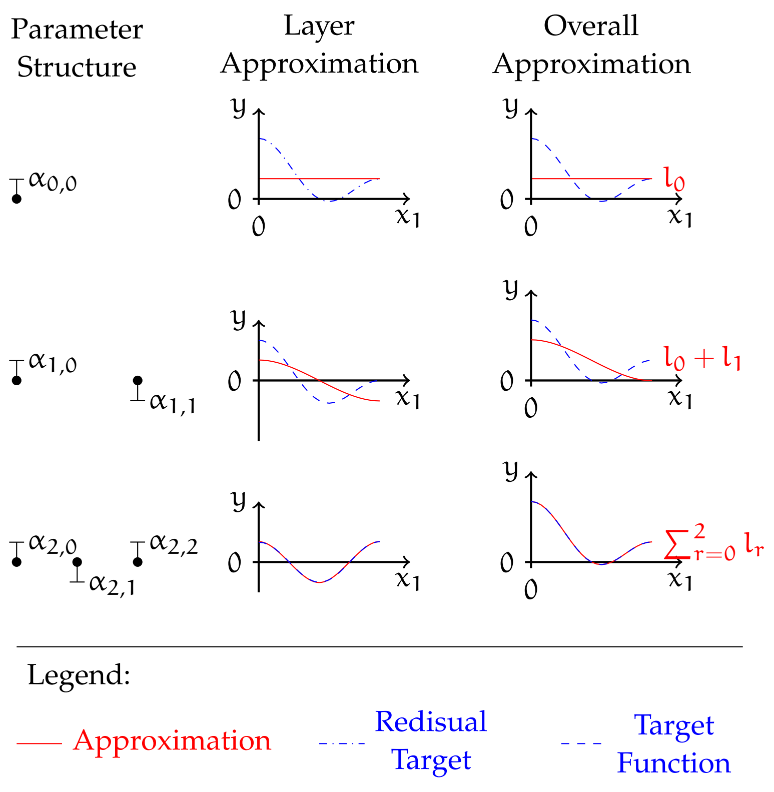


Figure 13: Illustration of the relation between parameter structure, layer approximation and overall approximation in one dimension for a non-linear target function in the aligned refinement structure. The parameter structure is shown in the left column by attaching the parameter heights to the aligned refinement structure. The corresponding approximation in each layer is plotted in the middle column. Here, the blue dash-dotted line marks the layer specific residual target function due to the learning architecture and the red solid line shows the approximation of each of these targets. The right column shows the resulting overall approximation. The blue dashed lines mark the target function and the red solid lines the approximation.

node in every layer $l_r, r > 1$ is formalized in equation (49). The extension to higher dimensions uses a multi-index $p_{r,\iota}$ with $\iota = (j_1, ..., j_n)$, i.e. one index for each input, which encodes the corresponding position along this dimension.

$$p_{r,j} = x_{min} + (x_{max} - x_{min})\frac{j}{2^{r-1}} \quad 0 \leqslant j \leqslant 2^{r-1} \tag{49}$$

The most important feature of the newly added nodes in each refinement layer is their connection to the simplicial input segmentation as every added node coincides with the center of an edge in the coarser segmentation. This strengthens the connection between subsequent layers and features the information propagation across them, as every node only relates to the adjustments of its one or two corresponding predecessors. The locations of these predecessors are perfectly aligned which allows for an easy and effective information propagation of the local adjustments.

The regularity of the aligned refinement strategy yields the desired equally distributed expressiveness and favors little memory access as all information about the node placement of the lower layers is encoded in the first refinement layer. The only drawback of the layer alignment is the similarity of different layers in terms of approximation properties and especially approximation artifacts. These artifacts appear at the center and boundary of every basis function ϕ_i and cannot be mitigated at points where all node positions and boundaries coincide, i.e. at the boundary of the input space. Thus, the aligned refinement strategy is best suited for approximating target functions with a vanishing slope at the boundary of the input space.

Figure 13 shows the resulting approximation of this refinement strategy in combination with the simple exemplary interpolation function from equation (16). The figure also highlights the relation between parameter structure, layer approximation and the overall approximation. This shows how a non-linear target function is decomposed into different approximation parts which follow the differential encoding scheme across the layers imposed by the learning architecture.

The situation shown in Figure 13 is the stationary best approximation of the non-linear target function. Hence, it shows the goal for learning when using this refinement strategy. But the residual target functions plotted in the middle column are not stationary during learning without the correct information propagation across subsequent layers. Thus, the central goal of the envisioned information propagation is to realize these residuals as stationary target functions. The necessary efficiency for keeping the memory access low forces the information propagation to comply with the learning architecture and the refinement strategy. This information propagation is introduced in subsection 3.2.3.

3.2.2.2 *Shifted Refinement Strategy*

The aligned refinement strategy is suited for representing functions with vanishing slope at the boundary of the input space. Thus, one goal of the shifted refinement strategy is to form a counterpart to the aligned refinement strategy with non-vanishing slope at the boundary. This allows to easily represent target functions which match this property and thus complements the class of target functions of the aligned refinement strategy. It shares the same regularity as the aligned refinement structure for memory access efficiency and uniform expressiveness, but is less focused on information propagation.

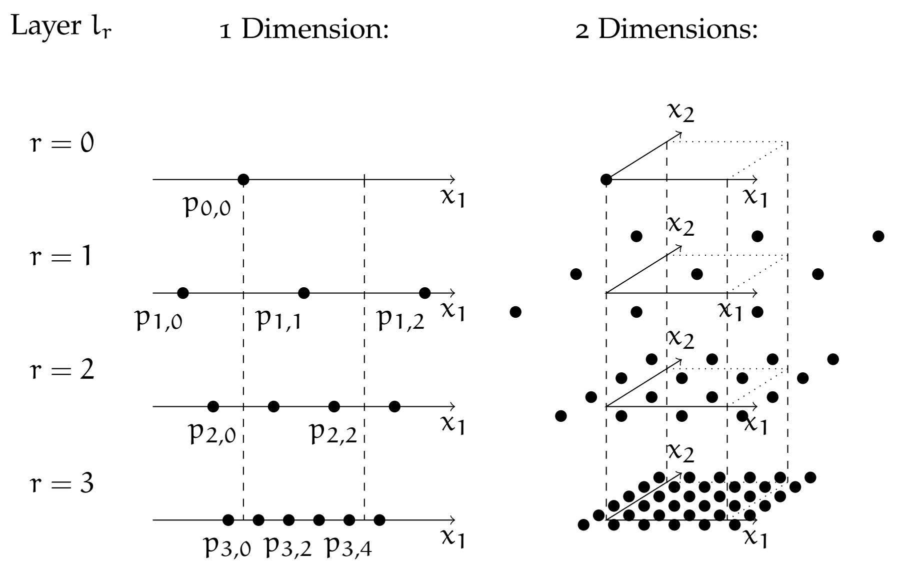


Figure 14: Illustration of the principle structure of the shifted refinement strategy and the resulting layers l_r in dimensions one and two. The nodes in each layer form a regular grid. Every node in each layer has a unique position in the layer architecture. The grids of subsequent layers are shifted relatively to each other by exponentially decaying amounts. The indexing of the node positions $p_{r,j}$ is shown exemplarily in the one dimensional case.

This yields a less stationary behavior of the residual target functions and lowers the data efficiency as a cost for the flat-spot-free overall approximation.

The mitigation of the approximation artifacts in each layer requires to define unique positions for all nodes in all layers. This way, the approximation artifacts of the nodes in any layer are balanced by all others. Preceding layers support the mitigation indirectly by smoothly filtering out the coarser behavior of the target function. This results in a reasonable initialization for the learning in the considered layer. Subsequent layers directly compensate the approximation artifacts as they are part of the residual target they deal with.

The principle structure of the shifted refinement structure is illustrated in Figure 14 for layers in dimensions one and two. The base layer consists of a single node placed at the origin of the input space. The first refinement layer covers the whole input space with a hypercubical complex of 2^n hypercubes grouped around a base node centered in the input space. This base node is the only node of the first refinement layer lying inside the input space, all other nodes are placed outside as each single hypercube of the complex fits the size of the whole input space. The subsequent refinement layers partly exceed the input space as well and are shifted relatively to each other by a displacement that is halved in each subsequent layer. The positions $p_{r,j}$ of the nodes in one dimension are defined in equations (50)

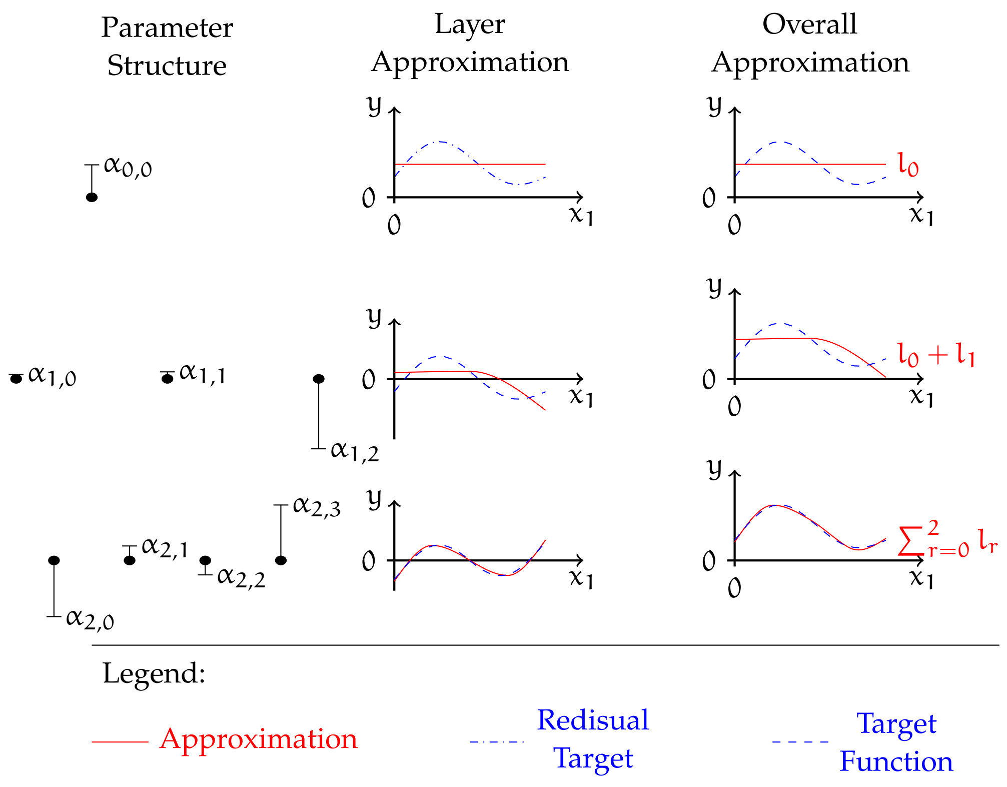


Figure 15: Illustration of the relation between parameter structure, layer approximation and overall approximation in one dimension for a non-linear target function in the shifted refinement structure. The parameter structure is shown in the left column by attaching the parameter heights to the shifted refinement structure. The corresponding approximation in each layer is plotted in the middle column with blue dash-dotted lines marking the layer specific residual target function and the red solid lines showing the approximation of these targets. The right column shows the resulting overall approximation. Here the blue dashed lines mark the target function and the red solid lines the approximation.

to (51) and the multi-dimensional extension is the same as for the aligned refinement strategy.

$$p_{0,0} = x_{min} \tag{50}$$

$$p_{r,j} = x_{min} - \frac{x_{max} - x_{min}}{2^r} + j\frac{x_{max} - x_{min}}{2^{r-1}} \tag{51}$$

The refinement layer grids, which exceed the input space, allow for a unique positioning of every node in the whole layer architecture while providing the necessary uniformly distributed expressiveness. These unique positions foster the non-zero slope at the boundary of the input space and avoid the flat-spot artifacts of the aligned refinement strategy. But as no node in subsequent layers is placed with respect to any input segmentation in other layers, there is no one-to-one or one-to-two correspondence between the nodes in a coarser and finer layer. This limits the quality and effectiveness of an information propagation as relations between the nodes in subsequent layers are more complex and handling them in an easy way ignores certain relation aspects. The resulting approximation and layer encoding of the shifted refinement strategy is shown in Figure 15.

3.2.2.3 *Combined Refinement Strategy*

The combined refinement strategy uses two different input segmentations in each refinement layer. One follows the aligned refinement strategy and the other one the shifted refinement strategy. The two resulting grids in each layer are shifted relatively to each other. This way, the approximation capabilities of both underlying refinement strategies are combined and their restrictions towards boundary conditions are canceled.

The whole combined refinement strategy separates into an aligned and a shifted part. Only in terms of information propagation for evaluation and learning a unified view is necessary. The single parts are the refinement strategies introduced above and they only share a common base layer with a single node. All subsequent combined refinement layers are a combination of one aligned and one shifted refinement layer. Figure 16 illustrates the principle structure of the refinement layers and highlights the aligned and shifted parts using points and crosses, respectively, to mark the node positions in one dimension. The extension to higher dimensions follows the standard tensor-based grid construction as indicated for the aligned and shifted refinement strategy.

The parameters α_A and α_S, the indexing p_A and p_S and the basis functions ϕ_A and ϕ_S are handled separately for the two parts of this refinement strategy. This allows to directly reuse the corresponding elements of the simple refinement strategies. In order to let the basis functions of the combined refinement strategy form a partition of

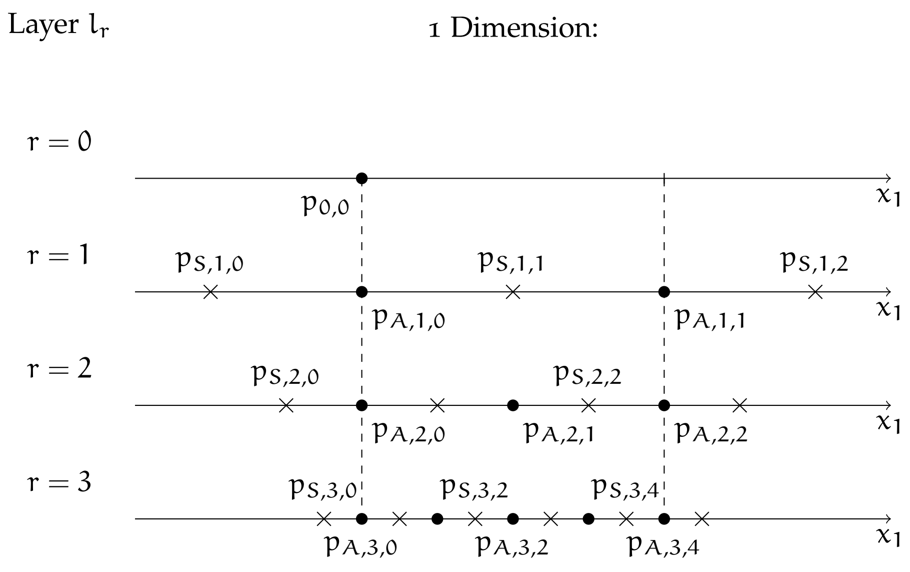


Figure 16: Illustration of the principle structure of the combined refinement strategy and the resulting layers l_r in one dimension. The base layer contains only one node. Each refinement layer consists of two regular grids, an aligned one and a shifted one. Nodes belonging to the aligned grid are marked as points and the ones belonging to the shifted grid by crosses. The indexing of the node positions separates into aligned nodes $p_{A,r,j}$ and the shifted node $p_{S,r,j}$, respectively, as shown exemplarily for some nodes in different layers.

unity, their basis functions are calculated according to equation (52). The reuse of the indexing relates the aligned node positions p_A to equation 49 while the shifted nodes p_S are governed by equation 51. The parameters and basis functions of the individual parts are indexed accordingly. Of course, this doubles the effort in terms of computational and memory demands compared to the simple refinement strategies, but the scalability is the same, especially with respect to Memory Access Ratio.

$$\phi(x) = \frac{\phi_A(x) + \phi_S(x)}{2} \tag{52}$$

By doubling the effort, this refinement strategy is not only able to adequately represent target functions which fit to the aligned or the shifted refinement strategy, but also the full spectrum in between. This turns every single refinement layer into a more powerful approximation, but the different layers remain sufficiently dissimilar with respect to approximation capabilities as the approximation capability of every layer is effected the same way and the learning architecture takes care to fully exploit the capabilities of each layer while passing the not representable details to subsequent layers. The increased expressiveness of every single layer also increases the necessary amount of samples in order to adequately learn the parameters. Thus, the combined refinement strategy trades reduced prior knowledge demands about principle target function properties for an increased learning sample demand.

The whole structure of the combined refinement strategy is similar to the Fourier Series. Both share a single global mean parameter, the sine terms of the Fourier Series correspond to the shifted part and the cosine terms to the aligned part. The strongest difference is the expressiveness in each refinement layer and the growth of the expressiveness in subsequent layers. Each refinement layer has a higher expressiveness compared to the corresponding frequency in the Fourier Series. The expressiveness of the layers grows exponentially while the frequencies in the Fourier Series grow linearly. This exponential growth of the expressiveness is present in Wavelets, as well. Thus, the combined refinement mixes elements of both approximation structures which are based on time-frequency transformations and yields a linearly scalable third variant.

3.2.3 *Information Propagation and Encoding*

The necessary information to propagate along subsequent layers is the adjustment of the higher layer in order to make the target functions of the subsequent layers stationary. A direct adaptation of all nodes in every layer would result in an exponential growth in the number of handled parameters with respect to the layer depth and thus, would yield a Memory Access Ratio of one. An efficient way of propagating the information is to represent the parameters $\boldsymbol{\alpha}$ in such a way that the adjustments in preceding layers become transparent for evaluation as well as learning, i.e. the learning architecture remains unaffected by the representation. In effect, the information propagates through subsequent layers in an indirect way.

Although it is necessary to propagate the adjustment information along subsequent refinement layers, the goal of the information propagation is not to fully compensate this adjustment. A match between the change of the approximation in a coarser layer and the induced adjustment in the finer layer would mean to restrict the expressiveness of the whole layer architecture to the capabilities of the bottom layer. This would render the original motivation to use a layer architecture in order to mitigate the approximation artifacts of high resolution grids invalid. Thus, the information propagation aims at balancing artifact mitigation and adjustment propagation in order to yield widely stationary residual target functions interfacing subsequent refinement layers which still allow for cooperation across all layers with respect to approximation capabilities.

The additive layer architecture in equation (48) states a differential encoding with respect to the refinement layers as every layer defines its approximation details relatively to its predecessor. This way the parameters in the refinement layer are usually globally mean-free and decay in magnitude from top to bottom. However, a direct representation of these parameters as in the left column in Figure 13 yields the problems mentioned above.

In order to achieve the desired indirect information propagation it is necessary to represent the parameters of the refinement layers as differences. This way, all necessary information is propagated on demand and the additional effort is minimal. The parameters α_i in each refinement layer are encoded as the difference between the parameter representation ω_i and its corresponding reference value ρ_i in the preceding layer as formalized in equation (53).

$$\alpha_i = \omega_i - \rho_i \tag{53}$$

The parameter representation $\boldsymbol{\omega}$ and the resulting information propagation are visualized exemplarily in Figure 17 for the aligned refinement strategy. The values of the parameters α_i in the left column are the same as in Figure 13 and thus, directly correspond to the approximation of each single layer. The parameter representations ω_i in the middle column of Figure 53 rather correspond to the overall approximation shown in the right column of Figure 13, but there is no one-to-one correspondence between the parameter representations in each layer and the according overall approximation because the overall approximation is influenced by the layer structure and the parameter representations.

In terms of Memory Access Ratio the parameter difference encoding is optimal as it only implies connections between parameters already used for evaluating each layer and thus only requires additional access to this very same set of parameters. This reuse of already considered parameters is optimally supported by modern caching technologies. All in all, the amount of information propagated across subsequent layers comprises the residual value due to the learning architecture and the parameter representations $\omega_{r,i}$ of all nodes responsible for the current evaluation or learning step.

This way, all parameter adjustments due to learning can be done locally and the changed information is propagated indirectly and on demand to the nodes requiring it. The adaptation of the parameters α_i due to learning only affects them indirectly. The learning adjusts the parameter representations ρ_i which are the only stored values. Thus, the parameter representation ω_i is fully transparent to the learning algorithm although it gets directly adjusted. In each learning or evaluation step, every parameter $\alpha_{r,i}$ is only calculated once and instantly discarded after completing the learning or evaluation.

These two differential encodings, one over the refinement layers and one for all parameters inside each refinement layer, balance each other, but they intentionally do not fully annihilate. The indirect information propagation in the difference encoding of the parameter representations adjusts the finer layer in order to compensate the change in the coarser layer. This adjustment ignores the non-linear interpolation and therefore fosters the cooperation of all layers with respect to the non-linear shape of the target function.

In addition to that, the induced adjustment onto the finer layer is not suited to perfectly fit the change of the coarser layer as the approximation capabilities of the finer layer are limited. The mismatch between the change in the coarser layer and the induced adjustment of the finer layer gives rise to a change in the target function for the finer layer and thus, yields a time-variant behavior of this target function as long as the coarser layer has not converged to a static function. This matches the main goal of the information propagation to balance this mismatch and the resulting non-stationary target function behavior against the cooperation across the refinement layers in terms of approximation artifact mitigation. This balancing is further limited as the information propagation needs to define a fixed propagation scheme which only makes use of the active nodes in order to preserve the Memory Access Ratio of the whole approach.

The differential encoding of the parameters α_i applies to all refinement strategies the same way. Only the calculation of the reference values ρ_i is specific for the individual refinement strategies and thus, defines the actual information propagation. These information propagations and reference values calculations are described in the following paragraphs.

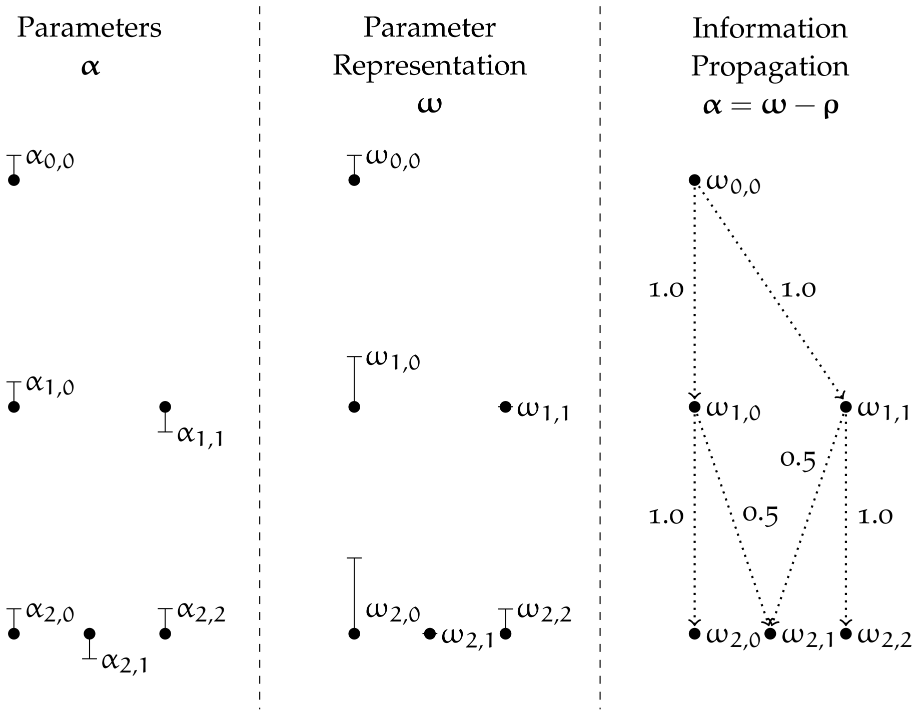


Figure 17: The three columns of this figure show from left to right the parameters α of the aligned refinement strategy from the example in Figure 13, the representation ω of these parameters in differential encoding and the path of the information propagation in a weighted directed graph. The reference value ρ for each parameter representation $\omega_{r,i}$ is the weighted sum of its predecessors in this propagation graph.

3.2.3.1 *Aligned Refinement Strategy*

Information propagation in the aligned refinement strategy exploits the strong coupling between subsequent layers. The reference value ρ_i for the nodes in a finer layer, that share the same position as the corresponding ones in the coarser layer, is just the parameter representation ω_i of the corresponding node in the coarser layer. An example for this kind of relation is the parameter representation $\omega_{1,1}$ in Figure 17. All other nodes in the finer layer relate to the center of an edge and each edge connects two nodes irrespective of the input dimension. Therefore, the canonical reference value for these kind of nodes is the mean of the parameter representation values of the nodes the corresponding edge connects. The parameter representation $\alpha_{2,2}$ in Figure 17 is an example for such an edge-centered node and it receives its reference value from the mean of the parameter representations $\omega_{1,1}$ and $\omega_{1,2}$.

In general, the calculation of reference values in the aligned refinement strategy follows equation (54). Nodes with an even index share the same position as their corresponding node in the preceding layer and map to the same value using floor and ceil operation. Odd indexes map to the two corresponding nodes in the preceding layer and are averaged. This simple indexing directly applies to multidimensional inputs by using multi-indexes and handling each single index according to equation (54).

$$\rho_{r,j} = \frac{\omega_{r-1,\lfloor j/2 \rfloor} + \omega_{r-1,\lceil j/2 \rceil}}{2} \tag{54}$$

Although the formula in equation (54) is correct for the common cases, it does not cover special cases where one of the referred predecessors is missing, e.g. the parameter representation $\omega_{1,1}$ from Figure 17 would refer to $\omega_{0,0}$ and $\omega_{0,1}$, but the latter does not exist. Whenever one of the referred predecessors is missing, the remaining one becomes the reference value. This also ensures that the weighting of the predecessor forms a partition of unity.

3.2.3.2 *Shifted Refinement Strategy*

The shifted refinement strategy allows for an easier information propagation compared to the aligned one. Every node has one unique predecessor it relates to and its reference value $\boldsymbol{\rho}$ is the parameter representation $\boldsymbol{\omega}$ of this predecessor. Figure 18 illustrates the parameter representation and information propagation for the shifted refinement strategy by reusing the example from Figure 15. Here the information flow forms a tree with the base layer node as root. All nodes in the subsequent refinement layers build the nodes and leafs of this tree. The calculation of the reference value $\rho_{r,j}$ is formalized in equation (55).

$$\rho_{r,j} = \omega_{r-1,\lfloor (j+1)/2 \rfloor} \tag{55}$$

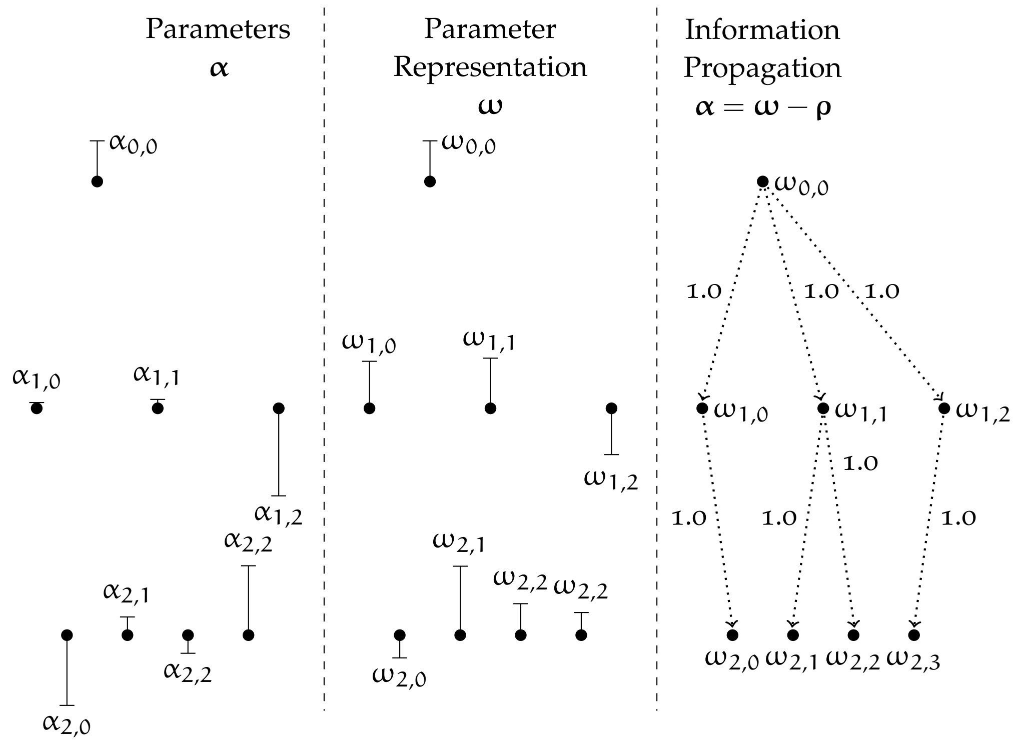


Figure 18: The three columns of this figure show from left to right the parameters α of the shifted refinement strategy from the example in Figure 15, the representation of these parameters ω in difference encoding and the path of the information propagation in a weighted directed graph. The reference value ρ for each parameter representation $\omega_{r,i}$ is its predecessor in this propagation graph.

The extension to multidimensional inputs is the same as for the information propagation in the aligned refinement strategy. Due to the simplicity of this information propagation, equation (55) covers all possible cases and no explicit handling of special cases is necessary.

This information propagation suffers from the restriction to only make use of nodes which are surely activated together in subsequent layer, i.e. the support of a basis function which corresponds to a node in a finer layer has to be a subset of the support of the corresponding node in the coarser layer. In the shifted refinement strategy the supports are not aligned to each other and thus a change in the coarser layer affects nodes in the finer layer which do not receive information about that change. An example for this case is the parameter representation $\omega_{2,0}$ in Figure 18. The layer output at the corresponding node position is influenced by the parameter representation $\omega_{1,1}$, but there is no information propagation between them as they are not surely activated concurrently. This limits the quality of the information propagation in terms of a minimal mismatch between the change in the coarser layer and the induced adjustment in the finer layer.

The concept of sure concurrent activation is of special importance for the combined refinement strategy as the information propagation there needs to handle nodes from different grids which are shifted relatively to each other.

COMBINED REFINEMENT STRATEGY

The information propagation in the combined refinement strategy uses parts of the aligned and shifted ones. Similar to the refinement strategy, the corresponding information propagation splits into two parts for the aligned nodes and the shifted ones. The reference values for the aligned nodes $\rho_{A,r,j}$ are composed of two parts. The first one is the reference value from the aligned refinement strategy in equation (54) including the handling of the special cases not directly covered by this formula, here referred to as $I_{A,r,j}$ for highlighting the internal information propagation inside the aligned part of the combined refinement strategy. The second part $C_{S,r,j}$ contains the cross-grid information. It has a similar structure but relates to the shifted grid in the preceding layer with the same special cases handling. The combination of both parts is formalized in equation (56).

$$\begin{aligned} \rho_{A,r,j} &= \frac{1}{2}\left(I_{A,r,j} + C_{S,r,j}\right) \\ I_{A,r,j} &= \frac{\omega_{A,r-1,\lfloor j/2 \rfloor} + \omega_{A,r-1,\lceil j/2 \rceil}}{2} \\ C_{S,r,j} &= \frac{\omega_{S,r-1,\lfloor (j+1)/2 \rfloor} + \omega_{S,r-1,\lceil (j+1)/2 \rceil}}{2} \end{aligned} \tag{56}$$

The structure of the reference values $\rho_{S,r,j}$ in the combined refinement structure follows the same principles as for the aligned ones. The grid-

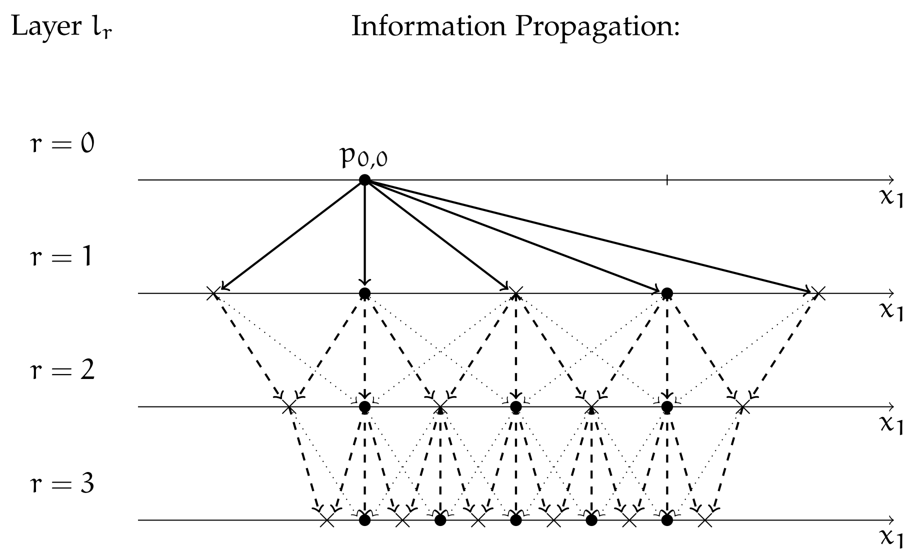


Figure 19: Principle representation of the information propagation in the combined refinement strategy. As the combined strategy comprises an aligned and a shifted part, the corresponding nodes are marked by dots and crosses, respectively. The actual information propagation by means of reference values is indicated by dashed and dotted arrows which represent weighing factors of 0.5 and 0.25, respectively. The base node is shared by all nodes in the first refinement layer. In general, each aligned node in a combined refinement layer gains information from three nodes in the preceding layer while the shifted ones only connect to two such nodes.

internal information propagation is accompanied by a cross-grid term as formalized in equation (57).

$$\begin{aligned} \rho_{S,r,j} &= \frac{1}{2}\left(I_{S,r,j} + C_{A,r,j}\right) \\ I_{S,r,j} &= \omega_{A,r-1,\lfloor (j+1)/2 \rfloor} \\ C_{A,r,j} &= \omega_{S,r-1,\lfloor (j)/2 \rfloor} \end{aligned} \tag{57}$$

The whole structure of this information propagation for the combined refinement strategy is visualized in Figure 19. The weight of the different edges is encoded in the line style. All nodes in the first refinement layer only relate to the common base node. Nodes in the subsequent layers relate to two or three different nodes from a preceding layer.

3.3 LOCAL ADAPTIVE LEARNING

The learning architecture and the information propagation introduced in the previous sections form a divide and conquer scheme for learning the whole layer architecture as they allow for a separate treatment of every single layer l_r. The only restriction for choosing an on-line learning algorithm for the single layers is its potential to handle time-variant target functions as the principally time-variant residual target functions the learning architecture imposes are not rendered fully stationary by the introduced information propagation in order to foster the cooperation of the layers. Thus, the desired on-line learning algorithm needs to combine great noise reduction with slight time-variance handling while scaling linearly in the number of handled parameters and fitting to the sparsity of the feature vector.

The great variety of adaptive learning algorithms reviewed in section 2.3 offers many candidates meeting one or two of the requirements listed above. The preferred approach used here is the local variant of RLS or AROWR, respectively, combined with the window estimation technique from [166] in order to determine the forgetting horizon. The resulting adaptive learning algorithm from this combination of concepts is detailed in the next paragraph. It is fundamentally devoted to time-variance adaptation and thus, in contrast to other adaptive learning algorithms, needs a mechanism to ensure a sufficient noise reduction, which is introduced in the subsequent paragraph.

3.3.0.1 *Variance-based Local Recursive Least Squares*

The LRLS update is a parameter-specific exponential smoothing filter with growing window size. As the development of the parameter vector $\boldsymbol{\alpha}$ is focused in this section, the indexing used to distinguish different versions of the parameter vector in subsequent time steps is $\boldsymbol{\alpha}_t$ and $\alpha_{t,i}$ refers to the i-th component of the parameter vector in time step t. A layer-specific indexing is omitted here, as all layers are learned subsequently and independently from each other with respect to the used on-line learning algorithm.

The parameter deviation $\mathbf{d}_t$ for the parameter vector $\boldsymbol{\alpha}_t$ given the current sample $(\mathbf{x}_t, y_t)$ is a normalized gradient as shown in equation (58). The update for the parameter vector $\boldsymbol{\alpha}_t$ using this deviation $\mathbf{d}_t$ is depict in equation (59) with $\boldsymbol{\gamma}_t$ being the time variant vector of parameter-specific learning rates.

$$\begin{aligned}
\mathbf{d}_t &= \left(y_t - \boldsymbol{\alpha}_{t-1}^T \boldsymbol{\phi}(\mathbf{x}_t)\right) \frac{\boldsymbol{\phi}(\mathbf{x}_t)}{\boldsymbol{\phi}^T(\mathbf{x}_t)\boldsymbol{\phi}(\mathbf{x}_t)} && (58)\\
\alpha_{t+1,i} &= (1-\gamma_{t,i}) \cdot \alpha_{t,i} + (\gamma_{t,i}) \cdot d_{t,i} && (59)\\
\gamma_{t+1,i} &= \frac{1}{\frac{1}{\gamma_{t,i}} + \phi_i(\mathbf{x}_t)} && (60)
\end{aligned}$$

The key factor in this update scheme is the learning rate vector $\boldsymbol{\gamma}_t$. A value of $\gamma_{t,i} = 1$ forces instant total forgetting by fully adapting to the current target $d_{t,i}$. A trade-off between noise reduction and time-variance adaptation can be achieved by a low constant value of $0 < \gamma_{t,i} < 1$. If the value of $\gamma_{t,i}$ decays over time as shown in equation (60), the envisioned LRLS behavior is achieved. The learning rate decays according to the activation of the local model represented by $\phi_i(\mathbf{x}_t)$. Each single learning rate $\gamma_{t,i}$ represents the inverse forgetting horizon $\tau_{t,i}$ of the local exponential filter as depict in equation (61).

$$\tau_{t,i} = \frac{1}{\gamma_{t,i}} \qquad (61)$$

VARIANCE-BASED WINDOW ADAPTATION

The update equations (58) to (60) represent the basic LRLS behavior. They perform parameter-specific noise reduction but are unable to track time-variant behavior. The incorporation of the windowing scheme from [166] modifies the update of the learning rate $\gamma_{t,i}$ in order to adjust the forgetting horizon and for limiting it to the currently relevant data.

The modification of the forgetting horizon is based on two statistics about the gradient information in $d_{t,i}$, namely its mean $\overline{d}_{t,i}$ and the mean of its squared value $\overline{d^2}_{t,i}$. The updates for these statistics are shown in equations (62) and (63), respectively.

$$\begin{aligned}
\overline{d}_{t+1,i} &= (1-\gamma_{t,i}) \cdot \overline{d}_{t,i} + \gamma_{t,i} \cdot d_{t,i} && (62)\\
\overline{d^2}_{t+1,i} &= (1-\gamma_{t,i}) \cdot \overline{d^2}_{t,i} + \gamma_{t,i} \cdot d_{t,i}^2 && (63)
\end{aligned}$$

In the stationary case, the mean of the gradient $\overline{\mathbf{d}}_t$ approaches zero and the mean of its square $\overline{\mathbf{d}^2}_t$ reflects the noise level. In this case a shortening of the forgetting horizon $\boldsymbol{\tau}_t$ is not necessary and it may grow as put in equation (60). A time-variant behavior of the target function causes a non-zero mean $\overline{\mathbf{d}}_t$ of the gradient $\mathbf{d}_t$ and the magnitude of this mean $\overline{\mathbf{d}}_t$ in relation to the mean of the squared gradient $\overline{\mathbf{d}^2}_t$ measures the impact of the time-variance. This yields the update for the forgetting horizon $\boldsymbol{\tau}_t$ in equation (64).

$$\tau_{t+1,i} = \min\left\{\left(1 - \frac{(\overline{d}_{t,i})^2}{\overline{d^2}_{t,i}}\right) \cdot \tau_{t,i} + \phi_i(x_t), 1\right\} \tag{64}$$

This adaptation of the forgetting horizon $\boldsymbol{\tau}_t$ yields the regular LRLS behavior as long as $\overline{\mathbf{d}}_t$ is small. The forgetting horizon rapidly shrinks as the squared value of the mean is similar to the mean of the squares $(\overline{\mathbf{d}}_t)^2 \approx \overline{\mathbf{d}^2}_t$, e.g. due to a shift in the target function. Each forgetting horizon $\tau_{t,i}$ is limited from below by $\tau_{t,i} \geqslant 1$ as indicated by the min-operator in equation (64). This ensures plausible learning rates and rests the learning rate $\gamma_{t+1,i}$ to one in case of a shift detection indicated by $(\overline{d}_t)^2 = \overline{d^2}_t$.

This kind of shift detection and the resulting total forgetting is useful for a fast and appropriate tracking of time-variant target functions, but the algorithm cannot recover from this one step forgetting horizon $\tau_{t,i} = 1$ as the statistics which determine the shrinking of the forgetting horizon are estimated using the same horizon. This even prevents starting the algorithm from scratch with initial values $\tau_{t,i} = 1$. The necessary shift recovery for this Variance-based Local Recursive Least Squares (VL-RLS) algorithm is presented in the following paragraph.

3.3.0.2 *Cold-Start and Shift Recovery*

The adaptation of the forgetting horizon $\boldsymbol{\tau}_t$ needs careful handling as it directly effects the learning rate $\boldsymbol{\gamma}_t$ and thus the overall behavior of the learning algorithm. If a shift occurs and the fraction $(\overline{d}_t)^2/\overline{d^2}_t$ equals one it is reasonable to reduce the forgetting horizon and adapt to the new data. But this fraction may reach a value of one independent of any time-variant event due to noise especially in the early stages of learning when the forgetting horizon is short by default. Thus, a suitable shift recovery mechanism not only has to enable the estimation of meaningful statistics $\overline{d}_t$ and $\overline{d^2}_t$ but also needs to prevent the algorithm from rapidly reentering the shift detection in case of noisy learning samples.

An easy way to achieve the desired shift recovery is to introduce a minimal forgetting horizon $\tau_{min} > 1$ and to disable the shrinking of each forgetting horizon $\tau_{t,i}$ as long as it is smaller than the minimal one, i.e. $\tau_{t,i} < \tau_{min}$. This ensures the noise canceling LRLS behavior until the minimal forgetting horizon is reached and allows time-variance adaptation afterwards. The minimal forgetting horizon can be defined globally τ_{min}, for each layer $\tau_{min,k}$ or for each parameter $\tau_{min,i}$ depending on the available prior knowledge about the expected time horizons. Nevertheless, this design parameter may have a significant influence on the learning behavior of the algorithm if a parameter-specific tuning is necessary but the prior knowledge only allows to set a global parameter, especially in high dimensions with potentially many parameters.

An adaptive shift recovery reduces the prior knowledge demands by estimating parameter-specific minimal forgetting horizons. The simple estimation scheme presented here starts by initializing each minimal forgetting horizon $\tau_{min,i} = 2$. As long as the current forgetting horizon is smaller than the minimal forgetting horizon $\tau_{t,i} < \tau_{min,i}$, the shrinking of the forgetting horizon is deactivated. Concurrently, as shown in equation (65), the accumulated activation $c_{t,i}$ is tracked in order to rate the current forgetting horizon.

$$c_{t+1,i} = c_{t,i} + \phi_i(x_t) \tag{65}$$

If a shift is detected before the accumulated activation $c_{t,i}$ has reached the double minimal forgetting horizon $\tau_{min,i}$, this detection is handled as a noise related shift but not as a false positive error. Like in a regular shift detection, it results in a reset of the forgetting horizon $\tau_{t,i} = 1$ and the accumulated activation $c_{t,i} = 0$. In addition to that, the minimal forgetting horizon $\tau_{min,i}$ is doubled in order to make the learning behavior more robust against noise. Any shift detection that happens beyond the doubled minimal forgetting horizon is handled normally without altering the minimal forgetting horizon. This way, the shift recovery has minimal influence on the global learning while ensuring a reasonable robustness against noise and allows starting the algorithm from scratch without prior knowledge about the necessary statistics and time horizons.

The complete on-line learning algorithm combining the VL-RLS and this adaptive shift recovery is shown as pseudo-code in Algorithm 1. In this algorithm, every parameter α_i is accompanied by statistics about the gradient and the forgetting horizon. All in all, there are five additional values which need to be stored together with each parameter α_i. They increase the total memory access but do not change the complexity class of the underlying approximation structure. By discarding some adaptive features of the algorithm the number of additional values can be reduces, but this replaces computational and memory demands by prior knowledge demands.

This adaptive learning algorithm completes the layer architecture to form a full on-line learning system capable of learning incrementally from data streams. The central design parameter in this learning system is the number of layers used for the approximation. Due to the multi resolution approximation, there are strong features embedded in the approach which allow an adaptation of the layer resolution. This layer adaptation is presented in the next section.

3.4 ADAPTIVE SIMPLICIAL MULTI RESOLUTION APPROXIMATION

The on-line learning system and its components developed in the previous sections already provide valuable properties for handling incremental learning problems on data streams. The central design parameter of the layer architecture is its depth, i.e. the number of layers. This parameter defines the learning behavior of the AS-MRA and thus requires tuning in order to optimize the performance of the incremental learning system. This tuning can be part of designing the incremental learning system or it can be estimated on-line. This section introduces two approaches for estimating the optimal layer depth on-line following different principles.

The introduced growing strategy focuses on minimal memory demands and precise control of the overall expressiveness of the AS-MRA in a fine grained manner. In contrast to that, the accordingly proposed ensemble evaluation approach uses a layer structure of fixed depth, but handles the evaluations of subsequent layers as individual ensemble members in order to choose the best performing one for the overall output. This way, the ensemble approach virtually decouples the layer structure for learning and evaluation as the layer depth for learning is fixed all the time and therefore guarantees a certain information exploitation of the incoming learning samples. Selecting the layer depth for evaluation based on the prediction performance - like in ensembles - allows to balance bias and variance components of the prediction error in order to optimize the overall performance.

3.4.1 *Growing Strategy*

The growing strategy represents a start small approach and allows for a fine grained expressiveness adjustment of the AS-MRA. The goal of the growing strategy is to shape the layer depth locally in order to accurately fit the target function while avoiding unnecessary memory demands. This section introduces the mechanisms and parameters of the growing strategy for building new nodes and layers on-line in an adaptive manner. This strategy allows to replace the layer depth as a global design parameter by more application related entities. These entities are meant to incorporate different kinds of prior knowledge without being mandatory. Thus, they allow for a trade-off between prior knowledge and learning sample demands.

The goal is to develop a set of optional user-defined parameters which allow to steer the behavior of the on-line learning system in a way similar to standard non-linear black-box optimization tools. These tools provide parameters to define the desired quality of the result in term of accuracy and resolution. Both measures can be defined either as absolute or relative values and mark stopping criteria for the overall optimization. Other parameters allow to restrict the effort in terms of iterations and function evaluations and thus, limit the runtime of the optimization. The same ideas apply to the layered approximation structure in on-line learning. But instead of limiting the computational effort, the focus in on-line learning is to limit the memory demand.

A growing strategy for the layered approximation structure is necessary in order to flexibly adapt to the data and target at hand while sticking to the constraints induced by the user-defined parameters. The growing strategy needs to allow for learning from scratch while building as many nodes and layers as necessary in order to meet the defined quality constraints and as little nodes and layers as possible in order to keep the memory effort low. The overall memory demand can easily be limited by defining a maximum layer depth and maximum number of nodes. Identifying the least important nodes when hitting the node limit is a non-trivial task within a global optimization scope and is not part of this work as it connects to central issues in compression techniques.

The quality and memory constraints follow a strict hierarchy as stopping criteria for the growing strategy. But all of them are dominated by a simple passing criterion, i.e. the minimal layer depth. Building a refinement structure from scratch with a restricted growing strategy means to start with a coarse input segmentation which refines as more data becomes available. This way, the information of each learning sample is handled as coarse as the input segmentation is at that time and details about the location of the sample are lost. Using a minimal layer depth ensures to handle the local information of each learning sample at least in the defined resolution. This yields a better sample exploitation compared to a restricted growing strategy at the cost of higher computational and memory demands in the beginning of the learning process.

The first and most natural stopping criterion for the growing strategy is the avoidance of nodes with vanishing parameters. Those only carry information about the location of the learning samples and do not alter the actual approximation. This ensures that in each learning step the number of layers at the most increases by one as any newly built layer fully adapts to the learning sample at hand, i.e. all subsequent residuals vanish. All other stopping criteria and the learning aspects they refer to are handled in the following paragraphs ordered according to their priority.

MEMORY LIMITATION

The memory limitation is one of the most fundamental user-defined restrictions. It is easy to test by counting the nodes and equally easy to guarantee by stopping the growing of new nodes and layers accordingly. Limiting the total number of nodes yields the most precise limit for the overall memory demand of the whole layered architecture. But this limit alone is not suitable for guiding the growing strategy towards a reasonable solution as the nodes would populate the input space according to the data density rather than approximation demands.

A less precise way of limiting the memory demand is to define a maximum layer depth. This also limits the maximal computational demands and the response time of the learning system for both evaluation and learning. The memory limit induced by the maximum layer depth is very rough as the number of parameters grows exponentially in the layer depth. Limiting the maximal layer depth automatically limits the input resolution as well, because the input resolution is doubled in each successive refinement layer, i.e. it scales exponentially with the layer depth.

Combining these two limitations allows to precisely control the memory demand of the layered approximation structure and the computational effort for evaluation and learning. With respect to learning behavior, this combination still guides the growing strategy based on the data distribution, but with the limited input resolution. Thus, input regions with high sample density not necessarily dominate the whole learning system.

When defining no memory limitations at all the learning system potentially makes use of all available memory and only stops building new nodes as the remaining memory is not sufficient for a new node and all tracked statistics accompanying it.

RESOLUTION LIMITATION

The maximal layer depth already acts like a resolution limitation as long as the span of the input space is fixed and known beforehand. If this prior knowledge is not available, the estimated span of the input space and the resulting input segmentation may change during the learning process. The handling of unknown input spans is based on input normalization using the 3σ formula in equation (66).

$$x_{norm,i} = \frac{x_i - \overline{x}_i}{3\sigma_{x_i}} \tag{66}$$

The involved statistics for normalization are estimated incrementally following a similar procedure as the overall on-line learning, but without an adaptive time horizon as in the adaptive learning algorithm. Therefore, there is no direct connection between input resolution and layer depth as long as the normalization statistics are yet to converge. Once, the mean and standard deviation have converged, the layer depth uniquely maps to a corresponding input resolution.

Combining the input normalization with the regular input segmentation in the refinement layers yields an equal relative input resolution for all inputs. This is usually inappropriate as not all inputs require the same resolution, but this issue is not further detailed here as there are many other options for handling dissimilar inputs and the whole topic inherently links to dimensionality reduction as the dissimilarity between different inputs is not limited to affecting their span but also their relevance. So, for the sake of simplicity and a focused discussion of the core concepts of the introduced on-line learning system, the handling of dissimilar inputs is only considered by normalizing all inputs.

All in all, the resolution limitation allows to steer the layer growing in cases of lacking prior knowledge about input spans or if those are time-variant. It is widely transparent to the input span estimation and only connects to the maximal layer depth as the normalization statistics converge.

ACCURACY LIMITATION

Conceptually, the accuracy limitation is the most powerful parameter for the user to steer the learning behavior as it defines a goal state of reasonable approximation quality. Unfortunately, the actual approximation quality and accuracy of an on-line learning system is generally unknown and can only be estimated by statistical means, especially when dealing with time-variance. This necessity to estimate the current accuracy of the learning system turns the corresponding limitation into an approximate one. Hence, the accuracy estimation is inherently delayed and requires a reasonable amount of learning samples before yielding sufficiently accurate estimations.

The accuracy estimation $\hat{a}_r$ reuses the statistics already tracked by the VL-RLS adaptive learning algorithm. The variance estimation of the parameters is mapped to an approximation error in order to estimate the accuracy. This way, the accuracy estimation is performed on the same time horizon $\boldsymbol{\tau}$ as the rest of the learning and thus, automatically adapts to time-variance. The validity of the accuracy estimation varies depending on the time horizon used for its estimation.

The influence of noise in the learning samples is not explicitly handled in this procedure, because the effect of noise on the approximation accuracy is indistinguishable from insufficient approximation capabilities. This becomes obvious in the bias-variance decomposition of the expected mean squared approximation error $\mathbb{E}\left[(e)^2\right]$ using the standard additive error model for the label generation, compare equations (69) to (72). When tracking these terms in learning curves over the number of learning samples, the variance part in this decomposition vanishes for a fixed approximation structure and an infinite amount of training data, but the bias and noise parts remain. Thus, when limiting the desired accuracy using an absolute value, the effect of noise needs to be considered, too.

$$\begin{aligned}
e &= y - \hat{f}(\mathbf{x}) && (67)\\
y &= f(\mathbf{x}) + \epsilon && (68)\\
\mathbb{E}\left[(e)^2\right] &= \mathbb{E}\left[(e)^2\right] - \mathbb{E}\left[e\right]^2 + \mathbb{E}\left[e\right]^2 && (69)\\
&= \mathrm{Var}(e) + \mathrm{Bias}(e)^2 && (70)\\
&= \mathrm{Var}(\hat{f}(x) - f(x) + \epsilon) + \mathrm{Bias}(\hat{f}(x) - f(x) + \epsilon)^2 && (71)\\
&= \mathrm{Var}(\hat{f}(x) - f(x)) + \mathrm{Var}(\epsilon) + \mathrm{Bias}(\hat{f}(x) - f(x))^2 && (72)
\end{aligned}$$

A remarkable feature of the layer approximation structure is its ability to support relative accuracy limits by comparing subsequent refinement layers. This procedure does not fully cancel the influence of noise to the accuracy estimation, but it allows to define the desired accuracy relatively to the noise level. The noise level is the same in all layers as they all receive shifted versions of the original learning sample. The bias component shrinks in subsequent layers as the expressiveness grows exponentially. Hence, a reasonable accuracy estimation $\hat{a}_r$ in all layers l_r allows to steer the growing strategy according to the bias reduction relative to the accuracy estimation.

$$\begin{aligned}
\frac{\hat{a}_r}{\hat{a}_{r-1}} &= \frac{b_r^2 + \sigma^2}{b_{r-1}^2 + \sigma^2} < (1 - \delta) && (73)\\
&\Leftrightarrow \frac{b_r^2 + \sigma^2}{b_{r-1}^2 + \sigma^2} - \frac{b_{r-1}^2 + \sigma^2}{b_{r-1}^2 + \sigma^2} < -\delta && (74)\\
&\Leftrightarrow \frac{b_{r-1}^2 - b_r^2}{\hat{a}_{r-1}} > \delta && (75)
\end{aligned}$$

The fundamental drawback of the absolute and relative accuracy limitation is their relation to the approximate accuracy estimation. This limits all theoretically sound observations with respect to their validity in any application as the accuracy estimation is affected by the actual approximation bias b_r^2, the sample noise σ^2 and the time horizon τ. The time horizons in different layers are usually not the same and decay in subsequent layers according to the sample density and time-variant effects. This makes the accuracy estimations in lower layers appear worse compared to the ones in higher layers. For the absolute accuracy limitation this potentially leads to unnecessarily built nodes in lower layers, which are detected as obsolete as the time horizon allows for a more realistic accuracy estimation. The relative accuracy limitation is affected the other way round as an increased accuracy estimation in a lower layer potentially cancels the effects of the reduced bias and thus, would prevent the growing strategy from building new nodes. This different behavior of the absolute and relative accuracy limitation allows to purposely steer the tendency of the growing strategy of being liberal or restrictive.

3.4.1.1 *Overfitting Detection*

All of the user-defined parameters introduced above are optional ones, leaving the available amount of memory as the only hard constraint. Another natural constraint for the growing strategy is not to build nodes with vanishing parameters α_i. But these two alone are not suited for a proper guidance of the growing strategy without any further restrictions as the approximation error is unlikely to vanish on the long run.

The layer architecture provides many valuable features for an overfitting detection which does not involve user interaction. Overfitting in this layer architecture means to use too many layers in order to approximate the target function. Thus, the lower layers all just try to learn the noise as there are no more function details left to approximate for them. So, in the limit of infinite learning samples these lower layers all have zero mean and their accuracy estimation $\hat{a}_r$ equals the noise level σ^2. For any sufficiently large finite amount of learning samples their accuracy estimations are at least similar. In case of a small amount of learning samples overfitting is hard to detect as the necessary statistics are not meaningful.

The actual distribution of the accuracy estimations over the different layers depends on the whole learning process, but this distribution tends to be flat tailed. The overfitting detection limits the length of the flat-spot to three layers by comparing its mean and variance to the rest of the distribution. Any distribution which increases in the last three layers is treated as overfitting as well.

Overfitting in this layer architecture is only a problem for on-line applications. Any data analysis task with a fixed amount of learning samples is free to use more layers than necessary as the sparsely populated layers can easily be ignored in the off-line analysis. In continuously running applications overfitting reduces the performance and wastes computational and memory resources. This simple overfitting detection is meant to complement the optional parameters in situations where their stopping criteria are not met, but a further refinement is not indicated. It is also an additional learning guidance in case of missing user-defined parameters and therefore, builds some kind of inherent limitation for the memory and computational demands with respect to the learning behavior.

PARAMETER OVERVIEW

All the parameters introduced in the above paragraphs connect to certain aspects of the learning process in order to incorporate prior knowledge accordingly. Although there are no mandatory parameters, it is important to keep in mind the overfitting detection as it always has the potential to affect the learning behavior. All optional parameters mark either passing or stopping criteria. The following list reviews them ordered according to their priority.

1. Maximal number of nodes
2. Maximum layer depth
3. Minimal layer depth
4. Maximal input resolution
5. Maximal absolute accuracy
6. Minimal relative accuracy gain

These six parameters allow to steer the behavior of the growing strategy and thus, of the whole learning system in a way similar to standard non-linear optimization tools. The actual behavior depends on the data at hand and the given accuracy limits may never be reached. But once they are hit, the learning converges to a fixed structure while every parameter heads to its optimal value with respect to all the data accumulated in its corresponding time horizon.

3.4.2 *Ensemble Evaluation*

The learning architecture of the AS-MRA decomposes the target function into a sequence of independent additive refinement components. This sequence allows to easily evaluate a given AS-MRA learning system at different layers. The ensemble evaluation makes use of this additive decoupling of an AS-MRA approximation in order to define independent layer depths for learning and evaluation. The layer depth for learning is fixed and represents a mandatory design parameter. The evaluation employs an adaptive layer depth by selecting the best performing layer of the fixed structure as the overall output of the AS-MRA. The performance is measured by estimating the mean squared prediction error of the output of all layers individually. This way, the learning is always performed in a fixed and predefined structure which ensures a certain information extraction from each learning sample. The evaluation treats the outputs of each subsequent refinement layer as originating from individual ensemble members and selects the best performing member for defining the overall ensemble output. This allows to have the overall output governed by layers which are sufficiently populated by learning samples in order to yield reasonable generalizations. The AS-MRA defines no actual ensemble as the different layers are not independent from each other due to the differential parameter encoding and the target function decomposition, but this decomposition also allows to treat the resulting subsequent refinement outputs as being independent.

The performance estimation is performed layer-wise with a single global parameter for each layer and uses the same time horizon as the root node of the corresponding AS-MRA. This improves the robustness of the ensemble evaluation towards noise while preserving its ability to adapt to non-stationary target functions in general.

The memory limitation parameters introduced in the growing strategy also apply for the ensemble evaluation. The layer depth and node count are controlled the same way as in the growing strategy. Here, the layer depth is a fundamental and mandatory design parameter for using the ensemble evaluation while the node limitation remains optional.

The introduction of a mandatory design parameter by the ensemble evaluation requires no fine grained tuning with respect to prediction performance in the targeted application. The ensemble evaluation automatically optimizes the prediction performance. Thus, guidelines for choosing the layer depth are simpler to state for the ensemble evaluation as this technique decouples the learning behavior and the layer design as long as the layer depth is sufficiently large. This allows to set the layer depth according to computational and memory demands rather than target function properties and setting the layer depth too large does not harm the prediction performance. Although there is no actual expressiveness adaptation in using the ensemble evaluation, this strategy follows a start big approach in contrast to the growing strategy described in the previous section.

Algorithm 1 : Variance-based local recursive least squares with adaptive shift recovery.

```
input  : Sequence of learning samples $(\mathbf{x}_t, y_t)$
output : Adjusted parameters $\boldsymbol{\alpha}_t$
Initialize:
  $\boldsymbol{\alpha} = 0$,   $\boldsymbol{\tau} = 1$,   $\mathbf{c} = 0$,
  $\overline{\mathbf{d}} = 0$,   $\overline{\mathbf{d}^2} = 0$,   $\boldsymbol{\tau}_{min} = 2$;
for $t = 1, 2, 3, \ldots$ do
    Receive learning sample $(\mathbf{x}_t, y_t)$
    Calculate activation $\boldsymbol{\phi}(\mathbf{x}_t)$
    Calculate gradient $\mathbf{d}$
    Initialize parameter adjustment $\Delta\boldsymbol{\alpha} = 0$
    for $i = 0, 1, \ldots, n$ do
        $\gamma_i \leftarrow 1/\tau_i$;
        $\overline{d}_i \leftarrow (1 - \gamma_i)\overline{d}_i + \gamma_i d_i$;
        $\overline{d^2}_i \leftarrow (1 - \gamma_i)\overline{d^2}_i + \gamma_i d_i^2$;
        $\Delta\alpha_i \leftarrow \gamma_i d_i$
        if $\tau_i < \tau_{min,i}$ then
            $\tau_i \leftarrow \tau_i + \phi_i(\mathbf{x}_t)$
        else
            $\tau_i \leftarrow \left(1 - \frac{(\overline{d}_i)^2}{\overline{d^2}_i}\right)\tau_i + \phi_i(\mathbf{x}_t)$
            if $\tau_i < 1$ then
                $\tau_i \leftarrow 1.0$
            end
            if $\left((\overline{d}_i)^2 == \overline{d^2}_i\right)$ then
                if $(c_i < 2 \cdot \tau_{min,i})$ then
                    $\tau_{min,i} \leftarrow 2 \cdot \tau_{min,i}$
                end
                $c_i \leftarrow 0.0$
            end
        end
        $c_i \leftarrow c_i + \phi_i(\mathbf{x}_t)$
    end
    $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}$
end
```

4 INVESTIGATIONS

The introduced AS-MRA requires investigations concerning the different parts it consists of and the learning challenges it is designed for. These two categories shape the structure of the investigations and the first part of it focuses on the individual parts of the AS-MRA structure like input segmentation, interpolation and layer structure. The learning architecture links to the behavior of the AS-MRA as an on-line learning system in different settings and covers issues like prediction performance, time variance tracking and noise reduction. Most of these aspects apply to all investigations as any particular learning behavior can be mapped to the AS-MRA parts responsible for it, but it allows to start the investigations on simple components in easy to grasp learning settings while heading for more complex and aggregated measures with respect to the learning behavior. The third and last part of the investigations covers the proposed expressiveness adaptation strategies and the effects of the user-defined parameters these strategies introduce.

The approximation structure and the learning algorithm used in AS-MRA allow for a reasonable comparison to other state of the art methods of their kind using common experimental settings. The expressiveness adaptation strategies are designed for the layer architecture and thus only allow for a comparison to other approaches which involve a similar concept. But as these strategies focus on non-linear parameter estimation, they are inherently bound to a specific approximation structure. Hence, the results of the corresponding comparisons do not provide the same clarity and precision of insights about the approach as the comparison of the other parts because effects introduced by the strategy cannot be isolated.

Despite these algorithm related considerations, there are also task related issues like input space dimensionality, target function complexity, time-variance and noise, which complete the spectrum of the investigations with respect to the requirements defined in section 1.3. The only requirement not handled explicitly in the investigations is the usability as this aspect is completely different in nature and widely handled in the design of the approach by avoiding mandatory parameters.

The whole set of investigation issues groups into algorithm related and task related ones. Covering all of these aspects and all of their possible interactions exhaustively would extend the volume of this work in an inappropriate way, thus the focus is on the key aspects and the necessary interactions which are related to general theoretical aspects as well as to severe practical issues like the stability-plasticity dilemma, bias-variance decomposition and target function complexity.

The experiments performed to investigate the different aspects of the introduced learning system and to compare it to other approaches start by looking at easy to grasp measures of performance and proceed to more elaborate and complex ones, which condense the essence of multiple experiments in a structured way. The different metrics also give rise to different perspectives onto the learning process as they highlight certain properties of the learning system or how one of these properties connects to the task at hand.

4.1 PERFORMANCE MEASURES

This section introduces the four basic performance measures used in the investigations and how they relate to different aspects of learning. The ultimate measure to rate on-line learning systems is their prediction performance with respect to a particular application as the expected gain in applying on-line learning is to improve the overall performance of the application while reducing the design effort. This prediction performance gives rise to the cumulative squared prediction error or CL as formalized in general in equation (2) and more precisely using the square loss in equation (76). The CL essentially sums up the squared prediction errors $(y_t - \hat{f}_t(x_t))^2$ over time which yields a reasonable measure for the overall prediction performance of the on-line learning system. As mentioned in the introduction 1.2, there are alternatives to using the squared prediction error, but the general concept of the CL of summing up prediction errors is universally applicable.

$$\mathrm{CL}(T) = \sum_{t=0}^{T} (y_t - \hat{f}_t(x_t))^2 \tag{76}$$

While the CL is good at reflecting the actual performance of the learning system in the application, it only indirectly measures the ability of the learning system to approximate the target function f as the labels y_t may be corrupted by noise. Measuring the Cumulative Target Loss (CTL) is only possible in experimental settings with defined targets. Compared to the CL, the CTL better reflects the ability of a learning system to generalize its approximation to the target function independent of noisy labels. The formal definition of the CTL is given in equations (77).

$$\mathrm{CTL}(T) = \sum_{t=0}^{T} (f(x_t) - \hat{f}_t(x_t))^2 \tag{77}$$

When focusing on the generalization properties in experimental settings, the CTL keeps track of the whole learning history, condensed into one measure which is sparse in time and space as it only handles the instances used for learning. A more global view onto the generalization properties of an on-line learning system is obtained by using a different set of instances $G_X = \{x_{g,1}, ..., x_{g,N_G}\} \subset X$ in order to estimate the Ground truth Loss (GL) of the current approximation $\hat{f}_t$ as shown in equation (78). The GL calculates the mean squared error between the target function and the current approximation on a set of test instances G_X for every time step t. Depending on the distribution of the instance set G_X, the GL allows for a global analysis of the approximation quality independent of the learning sample distribution.

$$GL(t) = \frac{1}{N_G} \sum_{i=1}^{N_G} (f(\mathbf{x}_{g,i}) - \hat{f}_t(\mathbf{x}_{g,i}))^2 \tag{78}$$

The GL is only measurable in experimental settings with given target functions. A more widely applicable measure is the Data Loss (DL) which concentrates on the ability of the learning system to memorize learning samples as formalized in equation (79). Although the DL does not provide the same insights about the generalization properties of the learning system as the GL, it still yields valuable information. A vanishing DL combined with a linearly growing CL is a strong indicator for overfitting, as the learning system simply stores all samples without generalizing to the target function. On the other hand, a huge DL indicates a lack of expressiveness of the learning system in order to approximate the target function. The DL automatically measures the prediction performance of the learning system with respect to the instance distribution and its applicability is limited to stationary targets as for a time-variant target past learning samples become invalid and thus, would bias the performance measure.

$$DL(T) = \frac{1}{T} \sum_{t=0}^{T} (y_t - \hat{f}_T(\mathbf{x}_t))^2 \tag{79}$$

These four kinds of measures allow to estimate the fundamental properties of on-line learning systems, i.e. prediction performance and generalization power. Other aspects like convergence rate and time-variance handling are indirectly covered by observing these measures over time. The only kind of measure that is used in all experiments as it is always applicable and meaningful is the CL. To some extent, it also reflects the aspects the other measures focus on, but not in such a pure manner. The connections between the different measures are highlighted in an exemplary learning task where the cosine function is to be learned using an aligned layer architecture of depth three. The results of this initial experiment are shown in Figure 20.

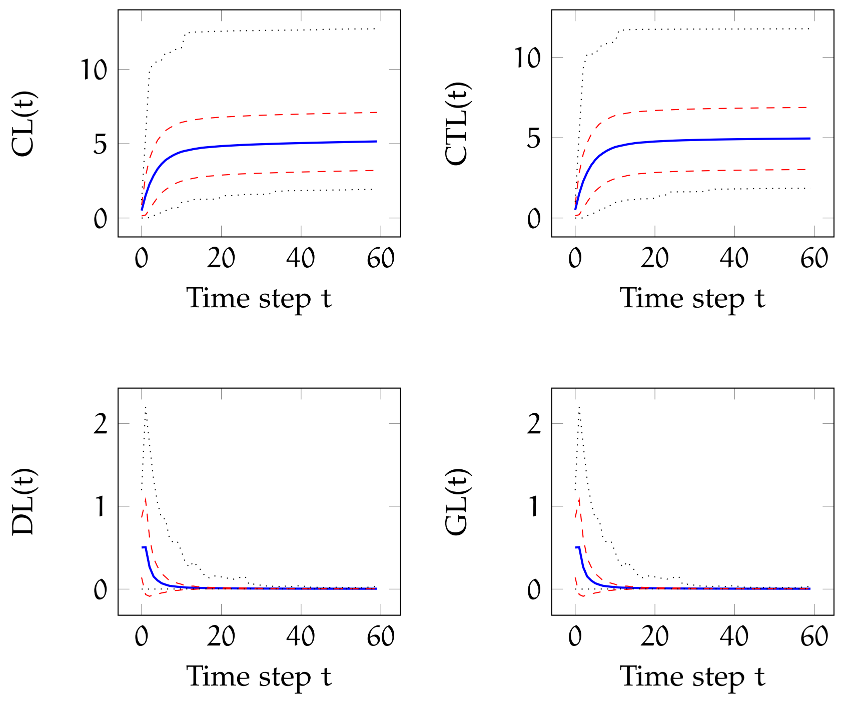


Figure 20: The four plots show the progress of the different performance measures in an exemplary learning task. The task is to learn the cosine function using an aligned AS-MRA with fixed depth of three layers. One trial of the experiment spans 60 samples whose instances are drawn according to a uniform distribution and there are 1301 trail repetitions. The results are presented by plotting the course of each measure using its mean value (blue solid line) with standard deviation (dashed red line) and the respective minimum and maximum (black dotted line) of all repetitions in every time step.

With respect to their nature, all four considered measures show a similar course over time. The mean values of the two cumulative measures rapidly saturate at a height about five while the other two measures decay to zero by the same time. Thus, the mean value preserves nearly the same information about the learning behavior in every measure, but the distribution of the performance is better reflected by the cumulative ones, because the non-cumulative measures and their whole distribution across repeated experiments with different random seeds converge to zero. While this is encouraging with respect to the convergence properties of the learning process as such, it is less informative in terms of learning process observation. The variety of different learning behavior due to random experimental settings is not reflected in the final distribution DL(60) and GL(60). So, all of the measures capture the behavior of the learning process and its convergence, but only the cumulative measures reflect the full variety of the behavior over the whole course of learning. Thus, the following experiments will only use the cumulative measures.

4.2 general setting

Although the experiments documented in this chapter investigate a variety of on-line learning aspects, there are some common elements which are shared across all experiments if not stated explicitly in the detailed description. These elements form the general setting for experiments and are described here. Following up the findings in section 4.1 all results are reported as mean CL over 1301 experiment trials. Additional statistical figures about the trials are presented on demand. The parameter vector α of all approximation structures used in any learning setting is initialized to zero, i.e. $\alpha_{t=0} = 0$.

The instance x for learning and target noise ϵ are both drawn according to a uniform distribution. The domains for such distributions are highly experiment specific and therefore not covered here, but the target noise is always mean-free. When comparing different on-line learning systems all of them receive the exact same learning samples in all trails and all trails are fully independent of each other.

If not stated otherwise the considered target functions are stationary. Experiments on time-variant targets are explicitly stated as such. The number of samples in each experimental trail is highly specific for each individual setting and in general is set to suffice a convergence of the on-line learning system towards a stable approximation. Most of the considered target functions tend to be mean-free. Therefore, the initial parameter vector $\alpha_{t=0} = 0$ comprises some kind of prior knowledge about the target, but only to the least possible extend without explicitly treating the complex topic of prior knowledge incorporation.

4.3 approximation structure properties

The first section of investigations is about general approximation properties of the layer architecture and the simplicial input segmentation it stems from. The experiments here are restricted to one dimensional problems in order to allow for a plain visualization and to connect the learning behavior to the performance measures and vice versa. The intuition stemming from these simple experiments forms the foundation to grasp the abstraction condensed in the more complex experimental settings.

The layer architecture comes in three different variants due to the refinement strategies and the resulting input segmentations. These variants are compared to similar model-based approximation structures from the review in section 2.1, namely polynomials, B-Splines, GLTs and Fourier Series. Polynomials, B-Splines and GLTs are examples for global, regional and local models and thus allow to rate the behavior of the layer architecture with respect to the locality of the model. The Fourier Series decomposes the target function into orthonormal parts similar to the layered approximation structure, but without an explicit information propagation or an enhanced learning architecture. All in all, the chosen approximation structures for comparison share one or another feature of the layer architecture and thus, allow to rate their performance to state of the art approaches with fixed structure.

4.3.1 *Target Function Properties*

Fundamental target function properties like continuity, monotonicity and non-linearity allow to categorize different target functions. They also partly determine how difficult they are to learn for a given online learning system. Target functions which share many properties of the approximation structure used in the learning system are easier to learn because they can be described by fewer parameters and thus require less learning samples for parameter adaptation. If the target function and approximation structure properties are very dissimilar a huge number of parameters many be necessary to realize a reasonable approximation with a learning sample demand increased according to the number of parameters. This section is to identify properties which the input segmentation and the layer architecture share with certain target functions in order to categorize the approach on a qualitative level. Further, this categorization allows to compare the investigated approach to standard approximation structures from related work.

For an easy to grasp start, here a set of test functions is considered where each function focuses on one property or combines several of them. This builds the foundation for a more general perspective of target function complexity in the next subsection. The full list of considered target functions and the properties they relate to is given in Table 1.

The selected target functions span a wide spectrum of properties but do not form a dense population within the frame of scope. The results for learning each target function with different compared approaches are presented as mean values over 1301 trails. These values strongly vary with respect to the considered target function. Therefore, the presented mean loss values in Table 2 are normalized with respect to the best and worst performing approach for each individual target, i.e. the best performing approach is mapped to zero and the worst performing one to one. The original absolute values are given in Table 7 in the appendix.

The common frame over all target functions with respect to learning setting comprises 300 samples which are drawn according to a uniform distribution over the unit interval $[0, 1]$. This amount of samples almost surely allows all approximation structures to converge while balancing the impact of the initial learning phase and the sta-

Table 1: List of simple target functions, their properties and pictographs.

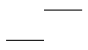
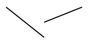
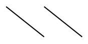
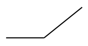
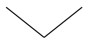
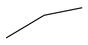
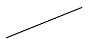
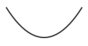
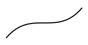
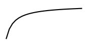
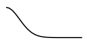
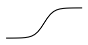
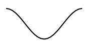
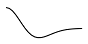
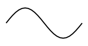
Target function	**Properties**						**Pictograph**
	Discontinuities	Non-differentiability	Saddle points	Monotony changes	Initial slope	Final slope	
Step function	1	1	0	0	0	0	
Linear step	1	1	0	1	-	+	
Parallel lines	1	1	0	1	-	-	
Kink rise	0	1	0	0	0	+	
Absolute value	0	1	0	1	-	+	
Kink saturation	0	1	0	0	+	+	
Linear	0	0	0	0	+	+	
Quadratic	0	0	0	1	-	+	
Cubic	0	0	1	0	+	+	
Hyperbolic	0	0	0	0	+	0	
Gauss	0	0	0	0	0	0	
Sigmoid	0	0	0	0	0	0	
Cosine	0	0	0	1	0	0	
Mexican hat	0	0	0	1	0	0	
Sine	0	0	0	2	+	+	

Table 2: Normalized Mean Cumulative Squared Loss results for different approximation structures and target functions. The normalized performance results for each target function are further aggregated to an average performance of each on-line leaning system as shown in the bottom line of the table.

Target function	GLT linear	2. Order B-Spline	Legendre	Fourier	Aligned	Shifted	Combined
Step function	0.68	0	1	0.91	0.88	0.38	0.3
Linear step	0.19	0.07	0.3	0	0.36	1	0.14
Parallel lines	0.4	0	0.58	0.93	1	0.33	0.11
Kink rise	0.01	0	0.02	1	0.04	0.3	0.02
Absolute value	0	0.05	0.06	0.02	0.07	1	0.08
Kink saturation	0.01	0	0.02	1	0.04	0.09	0.02
Linear	0.01	0	0.02	1	0.04	0.01	0.02
Quadratic	0	0.06	0.05	0.07	0.38	1	0.26
Cubic	0	0.03	0	1	0.32	0.05	0.06
Hyperbolic	0.05	0.11	0	1	0.3	0.08	0.12
Gauss	0.04	0	0.03	1	0.22	0.39	0.07
Sigmoid	0.08	0	0.15	1	0.51	0.28	0.21
Cosine	0.01	0	0.04	0.02	0.04	1	0.06
Mexican hat	0.03	0	0.04	0.67	0.53	1	0.15
Sine	0	0.02	0.03	0.02	1	0.2	0.17
Average	0.1	0.02	0.16	0.64	0.38	0.47	0.12

tionary approximation performance in the cumulative loss. In this basic investigation there are neither noisy labels nor time-variant effects, the focus is purely on target function properties. The formal definitions of the targets are given in Table 9 in the appendix.

The layer depth for all AS-MRA variants is fixed to three. This results in a total number of parameters of six, eight and thirteen for the aligned, shifted and combined AS-MRA, respectively. The number of parameters for the other approximation structures is fixed to seven.

The bottom line in Table 2 also states the average normalized cumulative loss as an overall rank for each approximation structure. In this ranking the second order B-Spline approach performs best followed by the GLT with linear interpolation. The combined AS-MRA achieves third rank but only performs slightly better than the Legendre Polynomials which are ranked fourth. The other variants of the AS-MRA and the Fourier Series perform much worse and form the bottom of the ranking.

The dominance of the local learning approaches in the general ranking is even more remarkable when looking into the details for the Sine and Cosine target functions. The Fourier Series allows for a perfect approximation of these target functions, but is not as well-performing as the local approaches in terms of CL. The only instance the Fourier Series performs best is the non-steady Linear step target function. The comparison of the local approaches to the Legendre Polynomial heads in the same direction, but there is no such clear dominance as the polynomials manage to be on par for the cubic target and are very similar in the linear and quadratic case.

The overall ranking already shows that the combined AS-MRA approach is superior to the variants it is composed of. Looking into the details of the different target functions reveals two principle cases which lead to this effect. In the first case, one of the components of the combined AS-MRA already performs quite well on a particular target function and the other one performs much worse. This is true for the target functions kink rise, absolute value, linear, cubic, hyperbolic and cosine. In these cases the combined AS-MRA performs slightly worse than the best performing single component due to the additional parameter overhead. However, the performance of the combined version is still much better than the one of the worst performing single component.

In the second case, both single components perform rather poor on the considered target function and the combined version outperforms them due to its increased expressiveness compared to the single versions. This is true for all other case, i.e. the ones not listed above. Thus, even in the worst case the combined version performs nearly as good as the performance of one of its components would be and in the best case the combination of the two subparts greatly improves the overall performance.

All in all, the third rank of the combined AS-MRA with respect to the average normalized CL bridges the performance of the local and global approaches. This reflects a fundamental property of the layered approximation structure. The top layers are global ones and are only as useful as their generalization quality. If the approximation capabilities of the top layers do not fit the target function properties, most of learning needs to be handled by the more local bottom layers. In such a case, the generalization of the top layers does not support the prediction performance of the overall system which yields a high CL.

The combined AS-MRA has increased expressiveness in each layer and especially in the top ones which fosters the generalization quality. From a theoretical perspective the increased expressiveness comes at the cost of additional parameters and thus basically hinders gener-

Table 3: Normalized Mean Cumulative Loss results for different approximation structures and target functions with layer depth equal to four.

Target function	GLT linear	2. Order B-Spline	Legendre	Fourier	Aligned	Shifted	Combined
Step function	0.79	0.33	1	0.94	0.66	0.05	0
Linear step	0.69	0.38	1	0.18	0.67	0.64	0
Parallel lines	0.57	0.25	0.72	1	0.51	0.02	0
Kink rise	0.01	0	0.02	1	0.01	0.08	0
Absolute value	0	0.16	0.22	0.07	0.18	1	0.13
Kink saturation	0.01	0	0.02	1	0.01	0.03	0.01
Linear	0.01	0	0.02	1	0.01	0.02	0.01
Quadratic	0	0.23	0.18	0.23	0.37	1	0.25
Cubic	0	0.03	0	1	0.02	0.04	0.01
Hyperbolic	0.05	0.11	0	1	0.07	0.07	0.03
Gauss	0.04	0	0.03	1	0.03	0.05	0.01
Sigmoid	0.08	0	0.15	1	0.27	0.05	0.03
Cosine	0.09	0	0.23	0.11	0.19	1	0.18
Mexican hat	0.05	0	0.07	1	0.17	0.2	0.05
Sine	0	0.14	0.23	0.21	0.81	1	0.37
Average	0.16	0.11	0.26	0.72	0.27	0.35	0.07

alization due to an increased sample demand. In this case, the combination of aligned and shifted subparts frees the combined AS-MRA from being bound to certain target function properties like evenness or oddness which becomes obvious from comparing the performance for Sine and Cosine target function.

The aligned and shifted version are only able to handle one of these target functions where the combined versions successfully approximates both. Hence, the aligned and shifted subparts of the combined AS-MRA behave like the Sine and Cosine subparts of the Fourier Series. Each single part is bound to certain target function properties, but these bounds are released by combining both which yields a more flexible approximation structure. This enhanced flexibility mostly outweighs the drawbacks from the increased number of parameters.

Setting the layer depth to three balances the total number of parameters between the aligned and shifted AS-MRA and the non-layered approximation structures. The dominance of the local approximation structures indicates that the governing factor for a fair comparison between the local approaches and the layered approximation structures is the input resolution rather than the total number of parameters.

Repeating the experiments with a layer depth of four balances the input resolution between the local approximation structures and the layered ones but nearly doubles the number of parameters in each AS-MRA approach. The normalized measures for the experiment with layer depth four are given in Table 3, the corresponding raw values are depict in the appendix in Table 8. Again, the bottom line in Table 3 shows the average normalized CL and this time the combined AS-MRA performs best, followed by the B-Spline and GLT approaches. The other findings about the relationship between the aligned or shifted AS-MRA to the combined version still hold true. In fact, by having a layer depth of four there is no single component which performs better than the combined version. So, the benefits of combining the aligned and shifted AS-MRA increase compared to the approximation using three layers.

This relationship between the layer depth and the approximation power is further investigated within the experiments of the next section. The results from this section show how the combined AS-MRA benefits from the enhanced approximation power compared to its single subparts and how this allows to approximate target functions with different properties more flexibly without suffering too much from the increased number of parameters.

4.3.2 *Target Function Complexity*

The complexity of approximation structures defining a certain class of functions is easy to measure for the on-line learning task considered here as the focus is on approximation structures which are Linear In the Parameters (LIP). Thus, the dimension of the parameter space yields a valid measure of complexity and this also applies to target functions as long as they belong to a certain class of LIP approximation structures.

Although the AS-MRA as an approximation structure is built from LIP approximations in each layer, the overall layer structure behaves slightly different as not all of the parameters in each layer fully contribute to the overall expressiveness of the AS-MRA. E.g. the base node represents the global mean of the target function, but the effect of this parameter to the output is fully suppressed by the differential encoding as long as the first refinement layer is densely populated by learning samples. The characteristic feature of the base node which triggers this effect is its flat output, i.e. all derivatives vanish and the output is a constant function irrespective of the input. Therefore, any subregion in a refinement layer which forms a flat region takes no effect on the overall output as long as there are further refinement layers for this particular region. This aspect of the AS-MRA fosters the layer depth as a more suitable measure for complexity because it allows to reflect the overall complexity without the need to count the actually relevant amount of parameters in a particular approximation.

The performance measures from section 4.1 are not only influenced by the complexity of the target function and approximation structure used for a particular learning scenario as further aspects like data distribution, noise and time variance also greatly impact these measures. Therefore, the experimental setting considered here is reduced to the simplest case for clear and focused results. The type of target function is stationary and restricted to randomly parameterized Legendre Polynomials with normalized coefficient vectors. The accompanying complexity measure is the polynomial degree. The approximation is built by the AS-MRA and only provides the layer depth as a rough complexity measure. In order to keep the learning process as simple as possible, the input space is one dimensional, the instances $\mathbf{x}$ are drawn according to a uniform distribution over $[0, 1]$, i.e. $x \sim \mathcal{U}(0, 1)$, and the labels are generated without noise.

The whole learning process is limited to 1000 samples irrespective of the target and approximation complexity. Hence, for simple targets the impact of the convergence phase onto the CL is dominant while for more complex targets this relationship gets more and more reverted, i.e. for complex targets the CL tends to reflect the initial learning phase which accounts for the variance part of the approximation error. All in all, this allows to compare the different learning settings on a common basis.

All results are compared to the performance of the zero prediction, i.e. an approximation which always outputs zero for every instance. This base line defines a worst case performance and allows to judge whether a learning system is capable of handling a certain target complexity or behaves like a non-learning system. The effects of randomly generated target functions and instances are mitigated using averaged values over 1301 trails for each combination of target complexity and layer depth. Results for the three different AS-MRA variants are shown in figures 23 to 24 in logarithmic scaling with respect to the performance axis as these values span about two orders of magnitude.

The zero prediction marks the worst case performance. In order to have a best case performance as well, the experiments are also conducted using a polynomial approximation structure instead of the AS-MRA. The degree of the polynomial approximation is varied the same way as the target function complexity as shown in Figure 21. The surface plotted in Figure 21 shows all properties of typical online learning behavior. First of all there is a sharp edge along the main diagonal between approximation and target degree which marks the

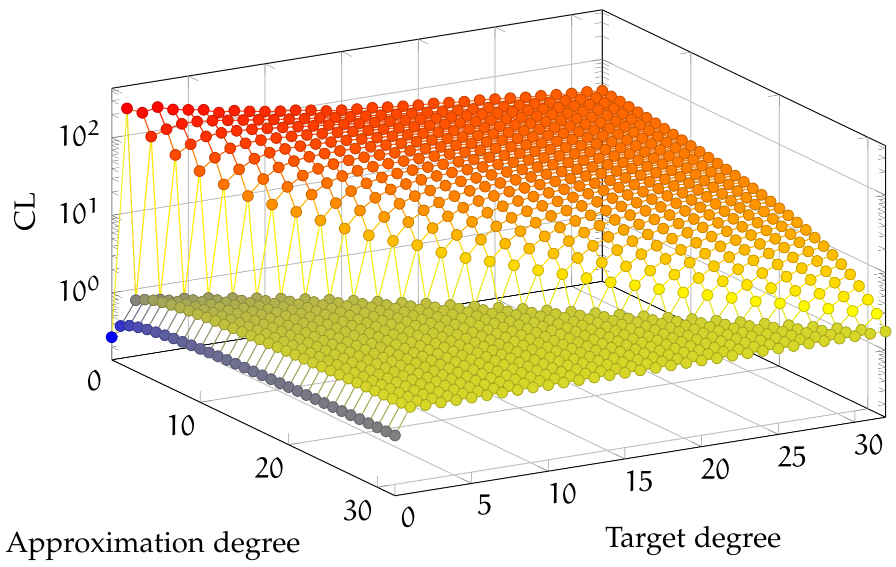


Figure 21: Prediction performance of Legendre-Polynomials as an approximation for random polynomial target functions. The performance is measured by means of the CL and is presented in logarithmic scaling. The degree of the random target polynomials and of the Legendre approximation, respectively, are varied from 0 to 31 yielding the shown mesh with dots colored according to the achieved CL.

best possible performance for each target complexity because there is no more appropriate approximation to a polynomial of a certain degree than using a polynomial of the same degree. This diagonal divides the whole surface into two monotony regions.

In the upper region with high CL values and thus poor performance, the approximation degree is lower than the target degree, i.e. the approximation is not able to fully represent the target function. In this case the worst observed performance is achieved when approximating a quadratic function by a linear one. The performance measure in this region is monotonically decreasing from this point in both directions, i.e. towards approximation degree and target degree. The monotony towards the approximation degree is due to the increasing expressiveness of the learning system and follows the simple rule of more is better as long as the approximation degree is smaller than the target degree.

The monotony towards the target degree is less obvious as it is counterintuitive that a high degree polynomial is easier to learn using a constant or linear function than a low degree polynomial. The reason for this monotony is the general shape of the randomly generated polynomial target functions. These tend to make full use of their expressiveness in terms of monotony changes and thus slightly wave around a global mean. This waving increases as the target degree increases and thus the mean prediction becomes more accurate in the CL measure. So, this monotony rather highlights a property of the randomly generated Legendre Polynomials and the CL measure than an actual learning issue.

The monotony region below the main diagonal is similar to the upper one but has partly reversed monotony. Here the monotony is increasing towards approximation degree and mainly decreasing towards target degree. The best performance is achieved for approximating a constant function by a constant function. The monotony towards the approximation degree is due to overhead parameters which are not necessary for the target but require additional learning data in order to be identified as irrelevant. Overly expressive approximation structures foster overfitting and produce poor generalizations. This effect is especially visible for the zero degree target which also breaks the monotonic decrease towards target degree. From target degree two on the performance is monotonically decreasing to the main diagonal.

The lower performance of the zero degree target is due to the overall learning setting. For approximating a zero degree target only one of the parameters of the approximation structure needs to be nonzero. This constant parameters is dominant in the feature vector as the input space of the experiment is limited to the unit interval. So, the constant parameter always has a weight of one according to the feature vector while all other parameters are assigned a value of at most one. This supports the correct identification of the single relevant parameter as it is highlighted by the combination of input space and feature vector calculation. The importance of all other parameters with respect to the feature vector actually depends on the position of each instant x_t in the input space and thus, they behave similar to each other.

The monotonic decrease of the performance towards target degree is far less obvious than all other monotonies in this surface and thus nearly negligible. It is due to the same reason as the increase along the approximation degree axis. The overhead parameters provide a poor generalization due to the excitation of not needed parameters during learning and additional learning data is required to tame the unnecessary expressiveness.

All findings for the plot about polynomials in Figure 21 represent typical learning behavior for an approximation structure with fixed expressiveness and recursive least squares based learning algorithm. Comparing this behavior to the ones using the AS-MRA variants focuses on the monotonies rather than comparing absolute values as it is hard to directly compare the expressiveness of two fundamentally different approaches. The results for the three AS-MRA with different but fixed layer depth are shown in Figures 22 to 24. In all of these experiments the target degree ranges from zero to 32 but the layer depth only covers one to seven layers.

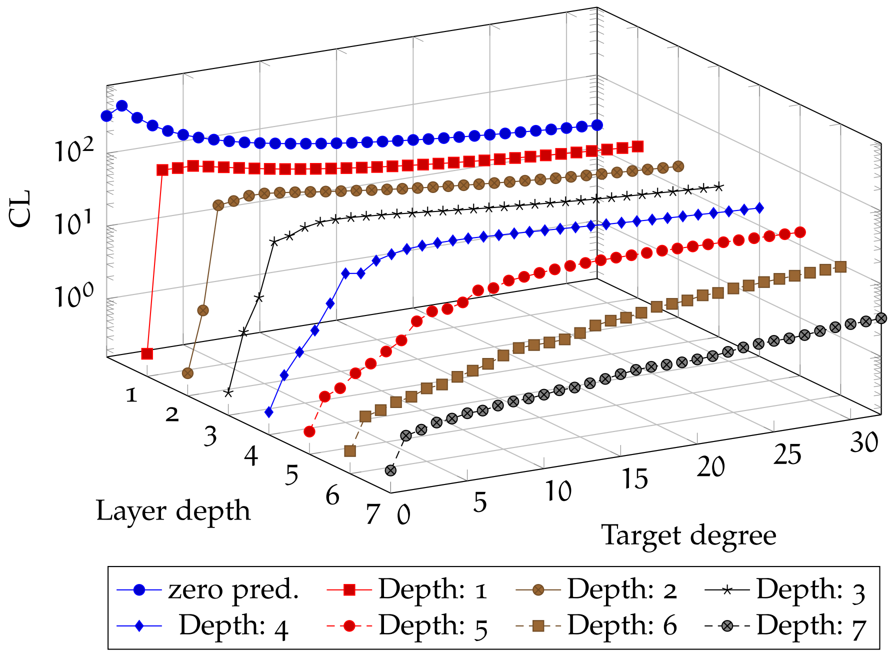


Figure 22: Prediction performance of aligned AS-MRA as an approximation for random polynomial target functions. The performance is measured by means of the CL and is presented in logarithmic scaling. The degree of the random target polynomials is varied from 0 to 31 while the layer depth of the AS-MRA is varied from 1 to 7. This yields the shown set of lines which are plotted in three-dimensional axes similar to the mesh in Figure 21.

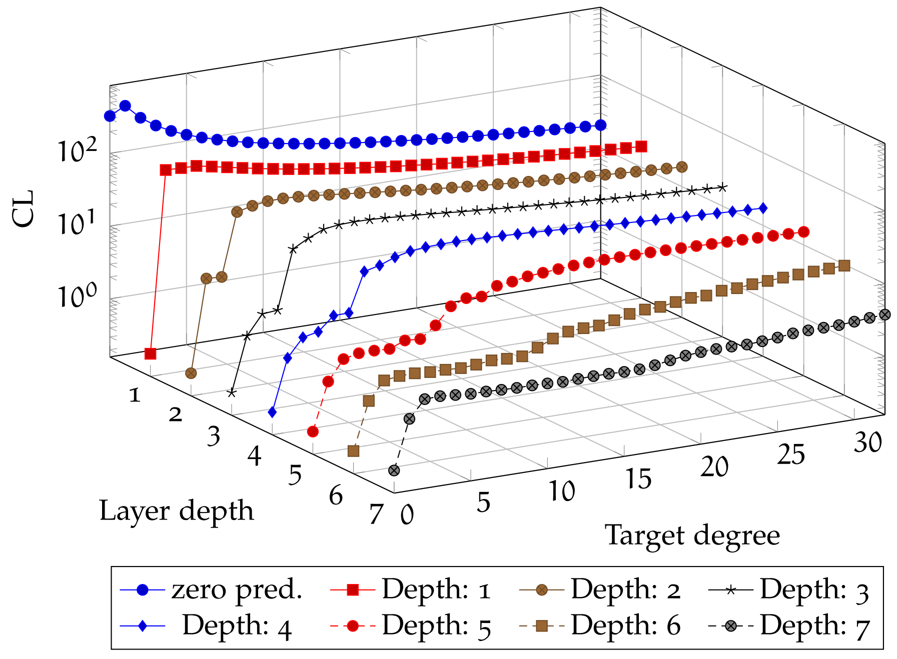


Figure 23: Prediction performance of shifted AS-MRA as an approximation for random polynomial target functions. The performance is measured by means of the CL and is presented in logarithmic scaling. The degree of the random target polynomials is varied from 0 to 31 while the layer depth of the AS-MRA is varied from 1 to 7. This yields the shown set of lines which are plotted in three-dimensional axes similar to the mesh in Figure 21.

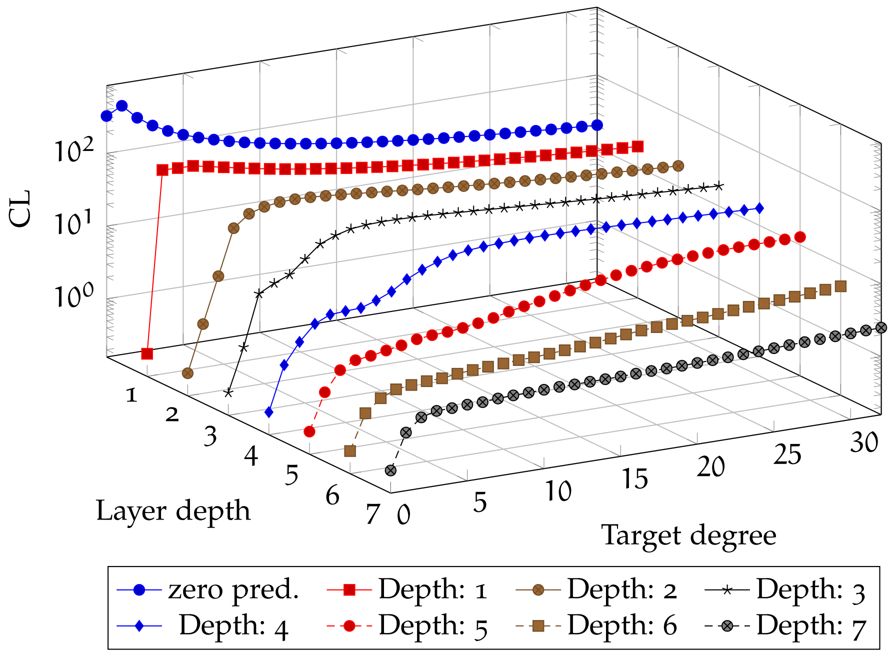


Figure 24: Prediction performance of combined AS-MRA as an approximation for random polynomial target functions. The performance is measured by means of the CL and is presented in logarithmic scaling. The degree of the random target polynomials is varied from 0 to 31 while the layer depth of the AS-MRA is varied from 1 to 7. This yields the shown set of lines which are plotted in three-dimensional axes similar to the mesh in Figure 21.

The most important observation here is about monotony as there is only one monotony in all three plots, which shows an outstanding property of all AS-MRA variants. The performance monotonically increases towards the layer depth, i.e. there is no overfitting or drawback from overhead parameters as for the polynomial approximation case. This is true for all three AS-MRA versions and all target degrees. The reason for this remarkable effect is the learning architecture which always starts with a simple constant approximation and makes use of the refinement layers only when necessary. For the overall expressiveness of the approximation, this acts as a regularization towards zero for all unnecessary layers, thus their parameters are never changed. Of course, this observation is limited to this noise-free setting, but even in a noisy environment the AS-MRA does not suffer from inherent overfitting due to obsolete parameters.

The learning architecture of the layered approximation resembles the group version of LASSO [64] approach with groups that are specific for the approximation structure. This does not represent any kind of prior knowledge about a potential target function, but fosters the principle of minimal curvature during learning. High frequent monotony changes and complex approximation shapes are only possible using the expressiveness of the bottom layers, but these are only used on demand. This also supports the generalization quality of the top layers as they consume most or even all of the learning samples in a rather restricted approximation which fosters plausible extrapolation. The realm of this extrapolation is actually twofold as it covers the input space in the way described in the introduction 1.3. Further, the extrapolation acts as an initialization for all subsequent layers and therefore, also covers the resolution axis.

Although there is no clear monotony towards the target degree, the performance values monotonically converge towards the performance of the zero prediction. Hence, this AS-MRA learning system never performs worse than a trivial non-learning one with limited prior knowledge. This is partly due to the fact that there is no label noise. The sharp edge along the main diagonal for the polynomial approximation is not present for the AS-MRA variants, but at least for a low layer depth, i.e. up to four, there is a certain slope towards the performance of the zero prediction or even a jump. This jump is present to its greatest extend in all approximation variants with one layer and gets eroded to a steady slope with increasing layer depth. This makes it hard to precisely state a maximal target degree for each layer depth of the three AS-MRA variants.

The monotonic convergence of the CL towards the zero prediction along the target degree is partly contradicted in the aligned and shifted AS-MRA variants, e.g. at target degree six and layer depth four in Figure 24 there is a slight decrease of the CL. The same kind of finding applies to Figure23 at target degree five and layer depth four. This

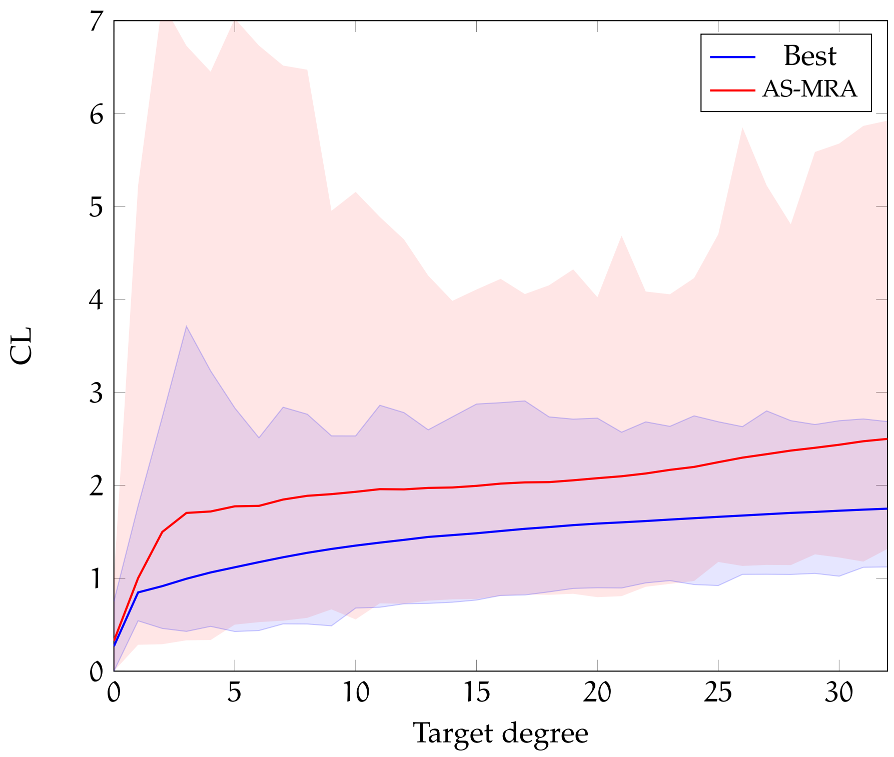


Figure 25: Comparison of the prediction performance of the combined AS-MRA with seven layers and the best approximation results of Legendre Polynomials from Figure 21. The mean CL is presented as a red or blue solid line, respectively. The accordingly colored regions indicate the corresponding minimal and maximal CL over all 1301 trails.

is due to the fitting of approximation and target function properties with respect to evenness and oddness as well as the average distribution of local minima and maxima. These effects do not appear in the combined AS-MRA which again underlines its potential to flexibly adapt to different target function properties.

The results from Figure 21 about polynomials allow for a more challenging comparison between the AS-MRA and a fixed polynomial approximation structure. The values along the main diagonal in Figure 21 mark the best possible prediction performance due to a perfect matching between target function type and approximation structure. In Figure 25 these best approximation results are compared to the CL performance of the combined AS-MRA with layer depth 7. The bold lines in blue and red show the mean CL performance for the best approximation and the AS-MRA variant, respectively. The accordingly colored areas highlight the minimal and maximal CL performance over all experiment repetitions. Thus, they show the steadiness of the learning success as well as best and worst case performance. A low maximal CL value indicates a robust learning and prediction performance. The explanatory power of the minimal CL alone is limited as it is related to single best shots in the conducted repetitions. But comparing all three curves reveals whether the average performance more resembles the worst or best case performance behavior.

The course of the two bold lines representing the mean CL performance is very similar, but there is a near constant gap between them. Only for target degrees of zero and one, the two lines nearly coincide. Thus, the prediction performance of the AS-MRA nearly behaves the same way as the best possible approximation. The relative difference between the measured CL values is huge as the CL of the AS-MRA is roughly twice the best approximation performance, but the absolute difference is small especially compared to the performance of the zero prediction. Moreover, the minimal CL values in this comparison are very similar and nearly coincide in the target degree range from 10 to 24. Great differences occur in the maximal CL values where the best approximation is rather steady and robust compared to the high and strongly varying values for the AS-MRA. However, the shape of the maximal CL does not resemble the mean performance line and therefore is related to single bad shots.

When looking at the development of the prediction performance of the combined AS-MRA with respect to layer depth as shown in Figure 24, there is a monotony in decreasing CL as the layer depth increases, but this decrease is limited and thus shows strong signs of convergence, i.e. additional refinement layers are unlikely to further improve the prediction performance. This is highlighted in Figure 26 which shows a projection of the content of Figure 24 onto CL and target degree. Hence, the performance of all different layer depths is shown and further compared to the best polynomial prediction as well as the zero prediction performance.

In summary, this comparison between combined AS-MRA and the best possible prediction performance due to fitting target and approximation structures shows that the AS-MRA performs remarkably well and even is on par with respect to the best case performance. The robustness of the best polynomial performance is much better than the one of AS-MRA, but even the worst case CL values of the AS-MRA are small compared to the ones of the zero prediction. Thus, the AS-MRA can learn nearly as fast and accurately as a perfectly designed approximation structure. It does not suffer from overfitting issues and its prediction performance in case of underfitting is limited from below by the performance of a non-learning system with rough prior knowledge about the mean of the target function.

4.3.3 *Scalability of Computational Demands*

This section investigates the actual scalability of the computational demands with respect to input dimensionality and layer depth of the used AS-MRA approximation structure. The overall learning scenario

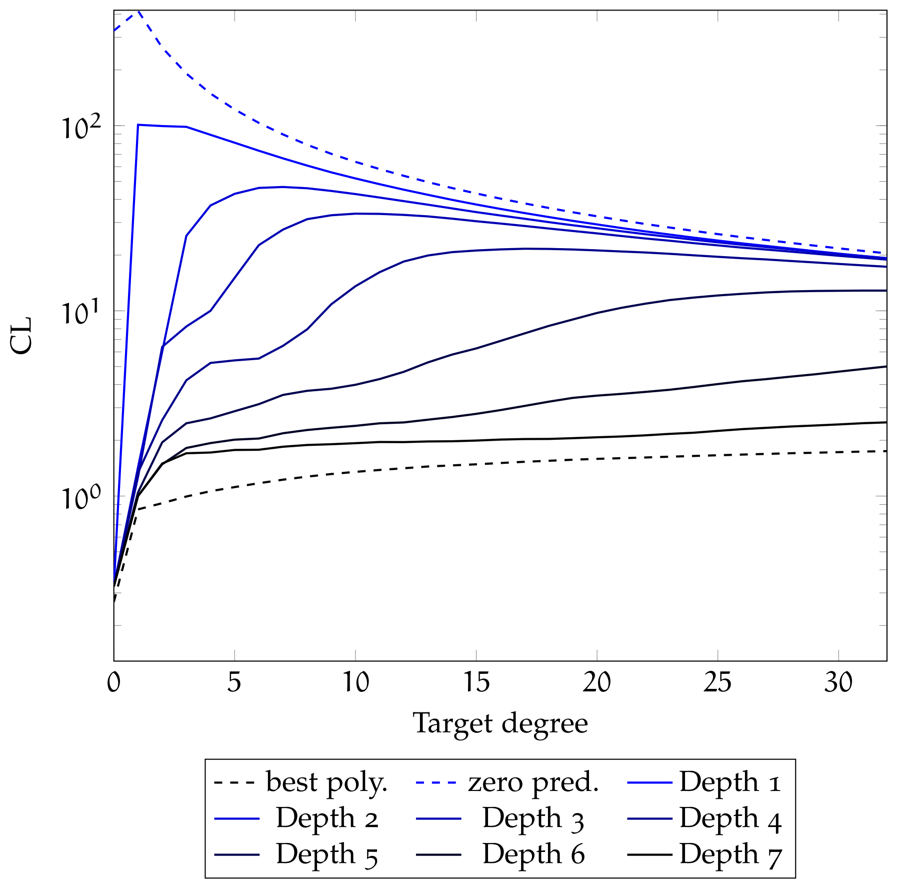


Figure 26: This is a projection of the results in Figure 24. The mean CL for different layer depths is shown as a set of colored solid lines. The color gradient from blue to black follows the layer depth from one to seven. The blue dashed line represents the worst case performance of the zero prediction while the black dashed line marks the best CL achieved by Legendre Polynomials of fitting target degree.

is kept as simple as possible in order to focus on the computational demands and their scalability. The input dimensionality of the learning task is varied using the rational normal curve $\mathrm{RNC}_n(a)$ as formalized in equation (81). The parameter n allows to define the dimension of the instances x_t. As the whole curve originates from only one parameter a, the underlying learning task stems from a one dimensional manifold, but is represented as a non-flat n-dimensional curve to the learning system.

The usage of the rational normal curve avoids beneficial interactions between the geometry of the simplicial approximation structure within each AS-MRA and the data distribution, because there is no linear subspace of lower dimension which can accurately reflect the data. This helps to generate instances x_t which populate inner regions of the involved simplexes and therefore require full memory access for evaluation and learning.

This setting further allows to test the computational demands without stressing the memory consumption. The necessary and available amount of memory depends on the task at hand and the hardware to tackle it. It is within the responsibility of the hardware to provide equally fast access to all resources, while the considered scalability in this experiment depends on the algorithms which operate on that hardware.

$$\mathrm{RNC}_n(a) : \mathbb{R} \rightarrow \mathbb{R}^n \tag{80}$$

$$a \rightarrow \{a, a^2, \ldots, a^n\}, a \in [0,1] \tag{81}$$

$$x_t \sim \mathrm{RNC}_n(\mathcal{U}(0,1)) \tag{82}$$

The plots in Figure 27 show the results on scalability measurements and all confirm the linear scalability of the proposed approach with respect to input dimension and layer depth. Only the scalability of the approximation structures with one layer do not scale towards dimension as the evaluation and learning of a single parameter always takes the same amount of time. The experiments were conducted on a i5-3400 with 16 GB RAM, but the absolute values are not as important as their relation to each other which reveals the linear scaling.

Moreover, the effort for handling the shifted AS-MRA is slightly higher than for the aligned one and the combined AS-MRA is as demanding as the sum of its single parts. So, there is no computational overhead in operating the combined AS-MRA compared to its single parts. The linear scaling regarding the layer depth also meets the expectations. With respect to the prediction performance observed in approximating randomly generated polynomial target functions, this linear scalability makes deep layer architecture of the aligned or shifted AS-MRA appear appealing compared to the doubled effort of the combined AS-MRA. Doubling the layer depth of an aligned or shifted AS-MRA has the potential to outperform a combined AS-MRA, but only if the properties of the target function match those of the chosen approach which requires a certain amount of prior knowledge about the problem at hand. On the other hand, the number of parameters is only doubled when comparing aligned or shifted and combined versions of the AS-MRA of equal depth, but doubling the layer depth causes an exponential growth of the available parameters.

The learning architecture takes care to regularize the parameters as strongly as possible, but this does not guarantee a reasonable generalization in the top layers and may result in an increased learning sample demand for the bottom layers. All in all, when focusing only on computational demands it is possible to exchange a combined AS-MRA approximation to a different version of the AS-MRA with doubled layer depth without changing the evaluation or learning time, but this will also affect the learning behavior in a way that highly depends on overall learning scenario and especially on the target function. Hence, computational demands, expected prediction performance and worst case memory demand can be traded for one and another in designing an AS-MRA learning system, but this trade-off needs to be supported by prior knowledge about the target function.

4.4 LEARNING BEHAVIOR: NOISE REDUCTION VS TIME VARIANCE ADAPTATION

The learning behavior of an incremental learning system mainly refers to its ability to handle noisy samples and a time variant target function. As stated in the introduction, these goals are mutually exclusive and a learning system needs to decide whether to follow a certain sample or to treat it as noise and thus remain stable. The perspective in this experiment directly focuses the stability-plasticity-dilemma and how the proposed VL-RLS algorithm with an adaptive time horizon positions itself in the field of tension between noise reduction and time variance tracking. In order to rate the performance of the proposed algorithms it is compared to standard on-line learning algorithms which mark the extreme cases along this axis. The Passive Aggressive (PA) on-line learning algorithm [36] continuously keeps track of time varying target functions while sacrificing its ability for noise reduction. The pure RLS contrasts this behavior by strictly converging to mean values indicated by the observed samples, thus filtering out the noise but also ignoring time variant effects.

The inherent regularization of the layer architecture also influences the learning behavior but in a rather passive way compared to the actual parameter adjustments taken by the learning algorithm. The general setting for testing the behavior of the three learning algorithms PA, RLS and VL-RLS is to learn the cosine over one period using a combined AS-MRA of depth three. This setting minimizes the influ-

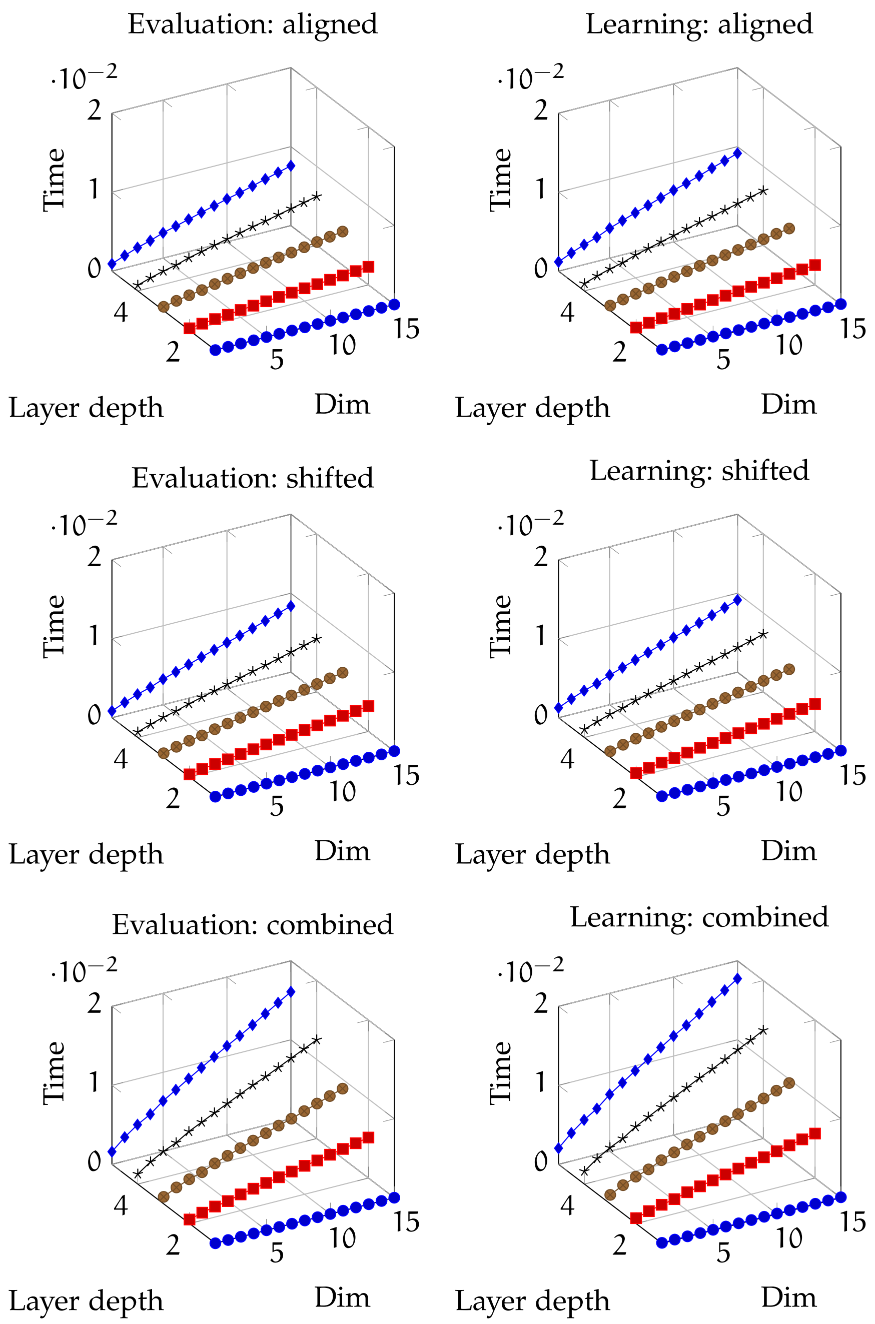


Figure 27: These six plots show the timing measurements for aligned, shifted and combined AS-MRA for evaluation and learning. Each plot shows the dependency of evaluation or learning time, respectively, on AS-MRA layer depth and input dimension. All results cover a span of layer depths ranging from one to five and input dimensions ranging from one to 15.

ence of structural effects like insufficient expressiveness or overfitting due to overly expressive approximation structures. In order to systematically investigate the relationship between noise reduction and time variance tracking, the impact of both components, noise and time variance, is linearly varied from zero to one. This affects the additive noise by adjusting the width of the uniformly distributed mean-free noise. The time variance is realized as a drift in the offset of the target function. The whole experiment spans 1200 samples drawn according to a uniform distribution. The first 400 samples originate from a stationary cosine function without any offset. During the next 400 samples the offset linearly rises to a configurable value and is stationary again for 400 samples. The configurable overall height of the offset allows to gradually steer the impact of the time variance similar to adjusting the noise level. For each such parameterized experiment the root mean values of the CL over 1301 repetitions with different random seeds are gathered in a grid where the noise and time variance levels are varied from zero to one in steps of 0.1. Here the Root Cumulative square Loss (RCL) is considered in order not to distort a potentially linear impact of the noise or time variance level while preserving the error weighting of the CL.

The results of these three experiments for PA, RLS and VL-RLS are visualized in Figure 28 using six different plots. The top row in Figure 28 shows the RCL of the PA and RLS algorithm as reference points for the learning behavior in case of noise and time variance. The RCL of PA linearly increases according to the noise level but is flat along the time variance axis. The RCL for the RLS algorithm also increases linearly according to the noise level but at a low slope which shows the noise reduction capability of RLS. However, the slope of the RCL for RLS also increases along the time variance axis which states the height of the offset. Thus, the noise reduction of RLS manifests in a reduced slope along the noise axis while it suffers from time variance as shown by the non-vanishing slope along the time variance axis. The middle row in Figure 28 shows the virtual best and worst case performance of the reference learning algorithms by taking their minimum and maximum performance measures, respectively. The best case performance shows a vanishing slope along the time variance axis and a reduced slope along the noise axis while the worst case performance combines the drawbacks of PA and RLS. The bottom row in Figure 28 shows the RCL performance of the VL-RLS algorithm and its gain compared to the virtual best performance for the ideal combination of PA and RLS. In general the shape of the RCL for VL-RLS resembles the one of the virtual best performance. It shows a reduced slope along the noise axis compared to PA and is still flat along the time variance axis. But the performance is not as good as the one of the reference algorithms in the extreme cases of dealing either with noise or time variance only. The difference between the RCL of the virtual best per-

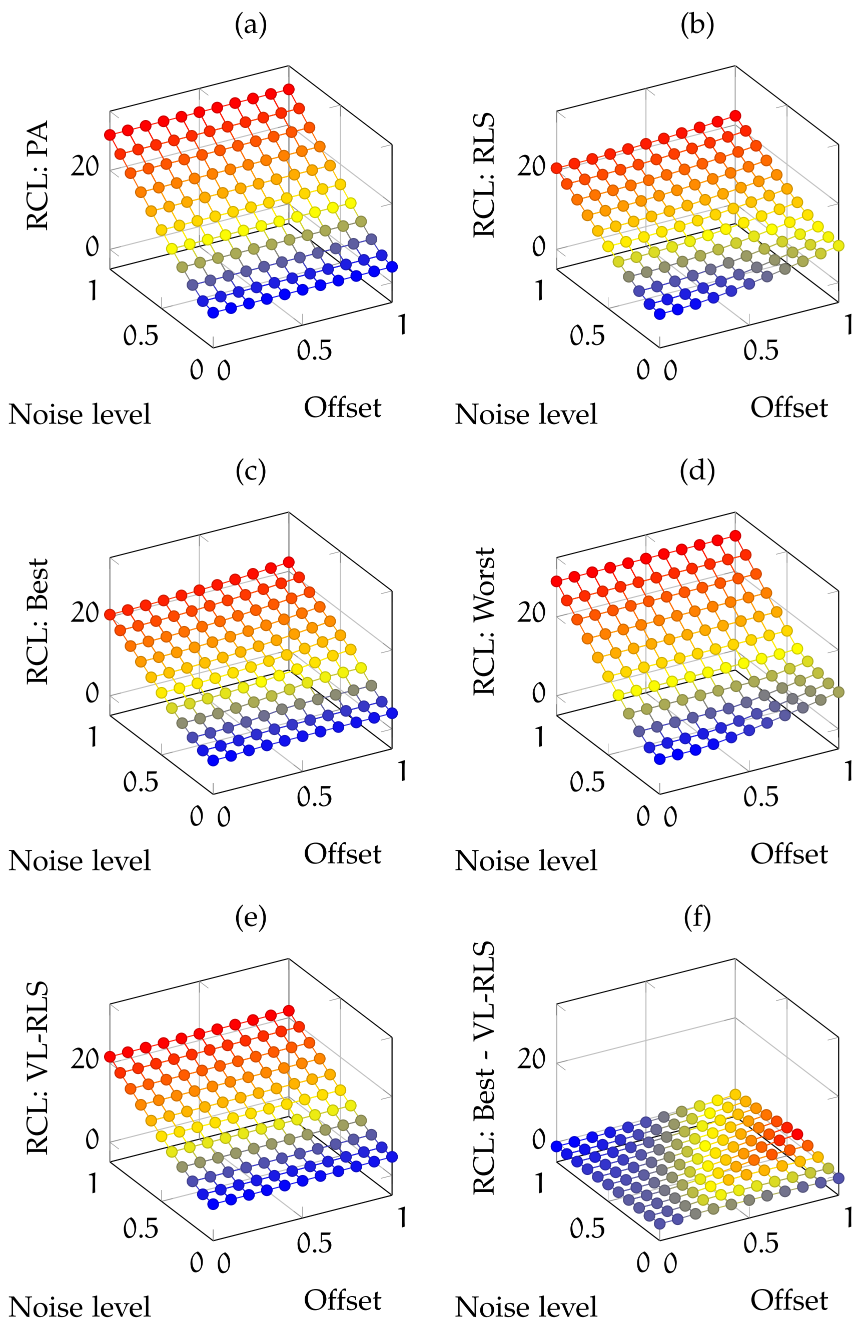


Figure 28: The plots (a), (b) and (e) show the prediction performance of a combined AS-MRA using different learning algorithms, namely PA, RLS and VL-RLS. The performance is reported as RCL, i.e. the square root of the standard CL. Further, the performance is measured for different noise levels and time-variant effects and presented as a mesh plot accordingly. Plots (c) and (d) represent the virtual best and worst performance results when combining the results from (a) and (b). Plot (f) shows the difference between the RCL performance of the virtual best result in (c) and the actual performance of VL-RLS in (e).

formance and the actual VL-RLS on the right side of the bottom row highlights in which cases there is a gain in using VL-RLS. The VL-RLS algorithm is best for handling samples with low or moderate noise which may originate from a target function showing highly time variant behavior.

4.5 GROWING STRATEGY

The layered learning architecture of the AS-MRA approximation structure is inherently self-regularized as shown in the experiments concerning the layer depth and target function complexity in section 4.3.2. Therefore, the growing strategy of the AS-MRA is not essential for learning purposes. It rather supports building a compact approximation with respect to memory demands. The scope of this section is to investigate the impact of the convergence related parameters for absolute and relative tolerance onto the resulting layer depth.

In addition to that, the layered growing strategy is compared to two other incremental learning approaches, namely FLEXible Fuzzy Inference Systems (FLEXFIS) [128] and Fast Incremental Model Trees with Drift Detection (FIMTDD) [93]. FLEXFIS is an evolving fuzzy systems approach and follows a bottom up strategy for building and growing fuzzy rules based on incremental clustering. This makes the FLEXFIS approach similar to instance based learning approaches while the output of each fuzzy rule is learned using RLS and thus allows FLEXFIS to be robust against noise. FLEXFIS is also chosen for comparison here due to its linear scalability with respect to the clustered fuzzy rules, so the overall complexity of FLEXFIS is similar to the one of AS-MRA.

FIMTDD represents a complementary growing approach compared to FLEXFIS as it builds a tree top down starting from a single root node. This strategy fosters generalization rather than memorization compared to the clustering of local information in FLEXFIS. So, FLEXFIS and FIMTDD are chosen the same way as PA and RLS for rating the learning behavior, i.e. in order to cover the principle cases, here for growing strategies. Both approaches FLEXFIS and FIMTDD inherently perform some kind of dimensionality reduction. FLEXFIS due to its local clustering which allows for representing a non-linear sub-manifold in the data distribution and FIMTDD due to its capability of detecting and ignoring irrelevant inputs. As dimensionality reduction is not within the scope of this work, the experimental settings for comparing the different methods will neither include a sub-manifold in the data distribution nor irrelevant inputs.

4.5.1 *Adaptive Layer Depth*

Adapting the layer depth according to user-defined convergence parameters allows the learning system to flexibly adjust the memory demands to the complexity of the target function at hand. The experiment considered here investigates how the layer depth shapes according to tolerance parameters for the relative and absolute approximation quality. The focus is not to give design guidelines how to choose these parameters as this is highly application specific but rather to show how they act in the context of the layered learning architecture.

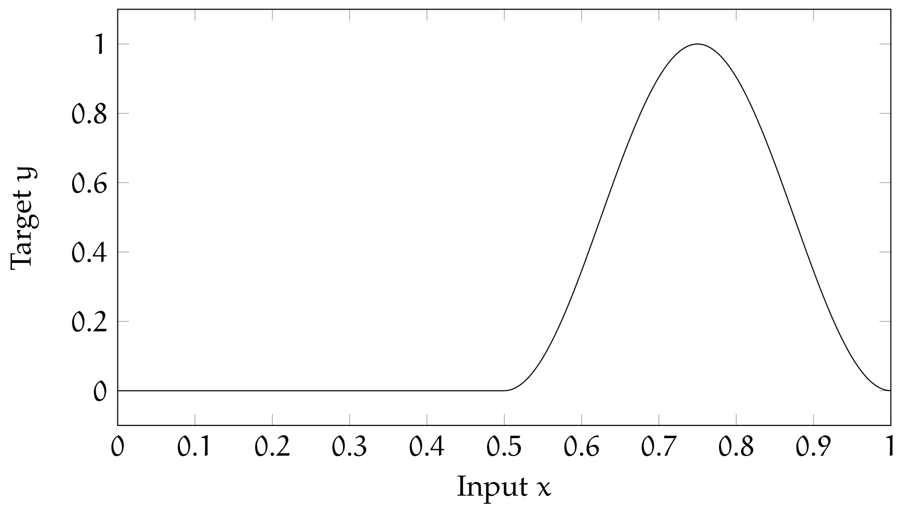
The refinement layers build upon a global approximation and become more local as the layer depth increases. Thus, it is not obvious whether the convergence across the layers is global as indicated by layered decomposition of the target function or whether the local nature of each individual layer fosters a local convergence. In case of a global convergence, the layer depth would be the same over the entire input space. For a local convergence, the layer depth would at least loosely resemble the shape of the target function with respect to local complexity in terms of monotony changes or highly non-linear regions.

The setting to test this property of the AS-MRA uses the combined version of the approach for approximating the target function shown in the top of Figure 29 over the unit interval as input space. The target is a constant zero below 0.5 and performs a half-period of a normalized cosine function above 0.5. This target strictly separates the input space into a flat and a structured region which allows to detect global or local convergence behavior of the AS-MRA for different tolerance parameters at least on a coarse scale. The parameters for relative and absolute tolerance are varied on a logarithmic scale spanning two orders of magnitude. The overall layer depth for each experiment is limited to ten layers and its duration is fixed to 125664 samples in order to yield convergence even for high accuracy demands. At the end of each experiment the resulting layer depth is measured at 51 points distributed uniformly across the input space.

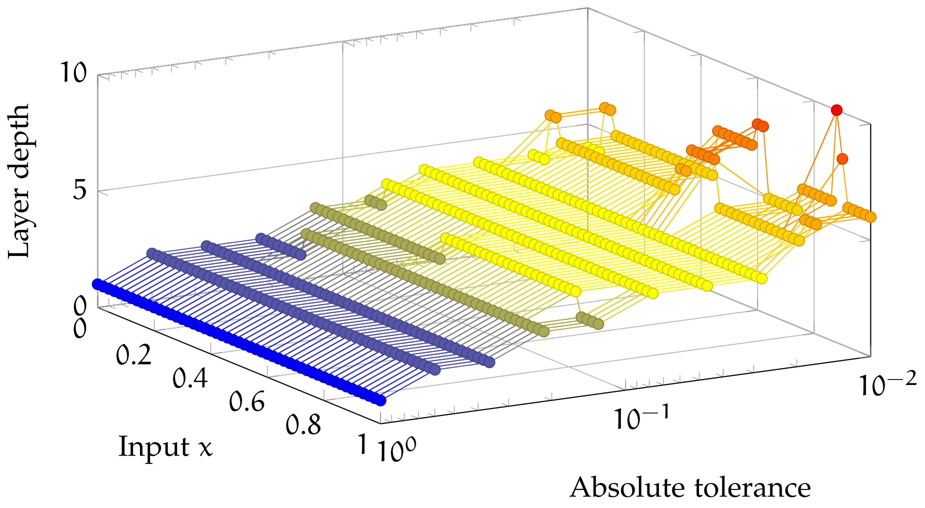
As the experiments for each parameter type and value are repeated 1301 times the mean layer depth of these repetitions is reported as one line for each parameter value. The resulting surfaces for varying absolute and relative tolerance parameters are depict in Figure 29 (b) and (c), respectively. The surfaces for each parameter type plotted there show the layer depth over the whole input space for different parameter values. The variance of these surfaces with respect to layer depth vanishes for the considered repetitions. Hence, the layer depth here appears as an deterministic function of target complexity and required approximation quality. This is in line with standard convergence properties of multi resolution approximation in general, but such results do not apply here in form of a formal proof as the interaction of subsequent layers is inherently time variant.

The effect of the absolute tolerance parameter is easy to grasp as it allows the creation of additional refinement layers as long as the

(a)



(b)



(c)

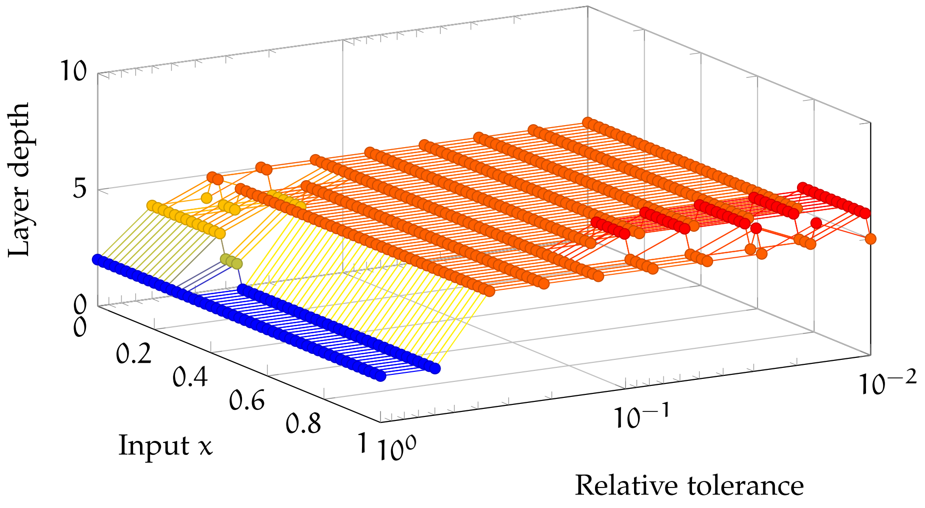


Figure 29: Plot (a) shows the target function used in this experiment. Plots (b) and (c) show the resulting layer depth over the whole input space for different absolute and relative tolerance parameters, respectively. The output range of the target functions allows all tolerance parameters to range from one to 0.01. The parameter axis is in logarithmic scaling as the actual parameter values are logarithmically distributed.

estimated prediction performance exceeds the user-defined tolerance parameter. The corresponding surface in Figure 29 (b) shows a monotonically increasing layer depth with respect to tighter convergence bounds due to smaller tolerance parameters. Considering the coarse discretization of the layer depth to integer values the surface nearly resembles a plane, but as the tolerance parameter axis is scaled logarithmically this implies an exponential dependency between layer depth and tolerance parameters. In fact, a certain layer depth always achieves the same prediction performance on the long run and therefore covers a certain range of tolerance parameters. This is reflected in the surface by regions which are flat along the parameter axis, e.g. a layer depth of four covers tolerance parameters ranging from 0.1 to about 0.04. This also shows that for this particular target function the fourth layer greatly improves the prediction performance.

Moreover, the shape of the layer depth is flat for large absolute tolerance parameters which already detect convergence for a rather rough approximation. This flatness dominates the whole surface except for the smallest tolerance parameters where local peaks in the upper half of the input space and the origin appear. Another exception to the overall flatness are two steps which are present in the fourth and fifth layer depth line. For these parameters the layer depth tends to follow the local complexity of the target function, but the difference is limited to a single layer.

In general, the global convergence of the layered learning architecture dominates the properties of the local approximations in the refinement layers. Only for high accuracy demands indicated by low tolerance parameters, the local nature of the refinement layer becomes relevant for the convergence and thus increases the layer depth locally on demand. The actual layer depth of an AS-MRA in any learning scenario also depends on the instance distribution as a single learning sample cannot increase the layer depth by more than one and a newly created layer is likely to perform well enough to trigger the convergence detection.

The surface for relative tolerance parameters in Figure 29 (c) is fundamentally different from the one for absolute tolerance parameters in Figure 29 (b) as it is mostly flat across the input space and especially for different relative tolerance parameter vales. A relative tolerance parameter of one triggers convergence detection as long as the estimated performance in subsequent layers increases. For a parameter value of 0.1 the relative performance increase needs to be smaller than 10 % for detecting convergence. Thus, the smaller the relative tolerance values the tighter the convergence band which essentially fosters higher layer depth.

As the relative tolerance needs to compare two subsequent layers, the minimal layer depth using this convergence criterion is two and this layer depth is achieved using a tolerance parameter of one. Starting from this baseline the layer depth widely remains on this minimum level for the first non-blocking parameter value of about 0.6 and instantly increases to the dominant layer depth of five for all smaller relative tolerance parameters.

There are two local structures in the otherwise flat step surface. One appears at the lower boundary of the input space between 0.0 and 0.2. It softens the step in the surface from a layer depth of two to five by stretching this transition across a wider range of tolerance parameters up to 0.1, but the overall sharp edge for approaching a layer depth of five is dominant.

The second local structure appears in the input space between 0.7 and 1.0. It shows a layer depth increased to six for tolerance parameters smaller than about 0.1. This local structure is the only sign of local convergence in the layer structure in an otherwise uniform and globally flat layer structure. The reason for this dominant layer depth of five becomes obvious from the wide range of absolute tolerance parameters the layer depth of four spans and the corresponding increase of the prediction performance. This gain is huge especially compared to the improvements introduced by the preceding layers, but the gain of adding a fifth layer is relatively small and hence convergence is detected and no further refinement layers are added.

This is remarkable as it clearly points out the fundamental difference between absolute and relative tolerance parameters with respect to learning behavior and design strategies. Defining an absolute tolerance parameter will always yield a layer structure which meets this requirement as long as there are enough samples to build the refinement layers and the detection mechanism just keeps adding layers as long as the estimated prediction performance appears insufficient. The resulting layer depth is likely to roughly resemble the local complexity of the target function with inherent limitations due to global target decomposition in the learning architecture, the global nature of the top layers and the coarse discretization of the layer depth at all.

The learning behavior induced by using the relative tolerance parameter is different as it is more focused on stopping at performance walls, i.e. if introducing an additional refinement layer increases the prediction performance only slightly, the growing stops and relative convergence is detected. This fosters a more uniform and global convergence as it is guided by the target decomposition and is less focused on local target complexity. Therefore, it yields a layer structure which represents the boundary at which the further performance improvement due to additional layers gets harder. In essence, the absolute tolerance parameter is best suited to force the layer structure to meet precisely defined accuracy requirements while the relative tolerance parameter allows a more agile behavior in cases where such precise requirements and thus prior knowledge is not available.

4.5.2 *Strategy Comparison*

The strategy comparison considers three methods, namely FLEXFIS, FIMTDD and AS-MRA. In this setting, AS-MRA is only used in its combined version in favor of focusing on the growing strategy and the ensemble evaluation, both using a maximal or fixed layer depth of six, respectively. The growing strategy for AS-MRA only follows the overfitting avoidance mechanism and thus there is no user-defined prior knowledge about desired approximation quality in terms of relative or absolute tolerance. The two AS-MRA strategies are further compared to different AS-MRA structures with fixed layer depth without ensemble evaluation.

So, there are four incremental learning systems using different expressiveness adaptation strategies and a set of combined AS-MRA approximations with fixed layer depths. The learning scenario makes use of randomly initialized Legendre Polynomials again, but this time their degree is fixed to four and the dimension of the input space is varied from one up to six. The number of samples is always fixed to 10000 and thus gets exponentially sparse with increasing input dimension. These samples are drawn according to a uniform distribution over the input space $[-1,1]^n$ with n being the parameterized input dimension. The target function $f_n(\mathbf{x})$ for the different input dimensions is the signed geometric mean of randomly initialized one dimensional Legendre Polynomials $\tilde{P}_4(\mathbf{x})$ of degree four, see equation (89). The product inside the geometric mean ensures to yield reasonably complex target functions for **LIP!** (LIP!) approximation structures while the n-th root prevents the product of the Legendre Polynomials to quickly converge towards zero. The sign term cancels the absolute vales which are required in the geometric mean in order to ensure valid root values irrespective of the input dimension and the output of the random Legendre Polynomials.

$$P_0(x) = 1 \tag{83}$$

$$P_1(x) = x \tag{84}$$

$$P_2(x) = \frac{1}{2}(3x^2 - 1) \tag{85}$$

$$P_3(x) = \frac{1}{2}(5x^3 - 3x) \tag{86}$$

$$P_4(x) = \frac{1}{8}(35x^4 - 30x^2 + 3) \tag{87}$$

$$\tilde{P}_4(x) = \sum_{i=0}^{4} a_i P_i(x), \quad a_i \sim \mathcal{U}(-1,1), \text{ s.t. } \|\mathbf{a}\| = 1 \tag{88}$$

$$f_n(\mathbf{x}) = \prod_{i=1}^{n} \operatorname{sign}(\tilde{P}_4(x_i)) \left(\prod_{i=1}^{n} |\tilde{P}_4(x_i)| \right)^{\frac{1}{n}} \tag{89}$$

The samples are labeled according to the parameterized target function $f_n(x)$ and they are further disturbed by a mean-free uniformly distributed additive noise of width 0.1. In this experiment, there is no time variance influence as the scope is to identify how the different growing strategies handle different input dimensions with a limited amount of learning samples. This requires fast adaptation while tackling moderate noise. Introducing time variant effects into this setting would require to cover two convergence phases for each compared method which would increase the sample amount in high dimensions drastically. So, the focus here is only on fast adaptation in different dimensions.

Varying the dimension provides valuable insights about the algorithms as different strategies face different challenges depending on the input dimension. For FIMTDD the one dimensional case is the easiest one because for each split in the tree it only needs to decide where to split and not along which dimension. This becomes harder as the input dimension grows because for each input the question raises of whether this is the best input to split and where to split along this input. Even more severe is the effect of wrong decisions with respect to the chosen input dimension as new samples are required to make a better decision. This is at least partly wasteful with respect to sample efficiency and limits the prediction performance in the beginning of the learning process. The full FIMTDD approach tackles this issue by handling ensembles of trees with on-line bagging, but this ensemble extension to the general tree growing strategy is not considered here as ensemble learning is a meta learning model which applies to all on-line learning system approaches. So, FIMTDD here only represents the growing strategy for a single tree.

FLEXFIS is not as directly affected by a growing input dimension as FIMTDD but faces some fundamental geometry issues because the meaning of concepts like neighborhood and distance change in high dimensions. In the Euclidean metric most points in high dimensions are far away from each other and tend to be located near the boundary of the input space. There are more elaborate approaches in the evolving fuzzy domain which tackle this problem, but they do not scale linearly with respect to the fuzzy rule clustering. This geometry issue also affects the interpolation between the fuzzy rules as the output around the center of a fuzzy rule is dominated by the attached local linear model and the transition between neighboring fuzzy rules tends to become a sigmoid and step shape. All of these aspects render the clustering in higher dimensions more challenging especially compared to the learning of the local linear models attached to each cluster.

The AS-MRA growing strategy is most similar to the tree growing as it starts at coarse global approximation and takes refinement steps by introducing additional layers on demand. The fundamental difference between FIMTDD and AS-MRA with respect to these refinements is the predefined splitting selection as the AS-MRA splits along all input dimensions and at a priori defined positions when adding a refinement layer. This way the AS-MRA is not threatened by wrong splitting decisions but it is bound to a fixed input segmentation which may be suboptimal for a particular target function which requires additional refinement layers to achieve the same prediction performance as the corresponding target specific tree structure.

The AS-MRA also shares some properties of FLEXFIS as the norm-based interpolation scheme for the simplicial input segmentation defines an RBF structure comprising constant models with strictly local support. The simplicial input segmentation is locally biased as it puts special emphasis on the main diagonals of the input space with respect to the hypercubical segmentation. This bias affects the concepts of distance with respect to the interpolation as connectivity between vertexes strongly varies depending on the position in the input space. Vertexes which are far away from each other contribute to the same interpolation as long as they are connected by a main diagonal while other vertexes with the same Euclidean distance never directly cooperate.

The variety of different connectivity patterns for the vertexes increases as the input dimension grows. In essence, the more the AS-MRA layer structure grows, the more local it becomes and therefore potentially begins to resemble the FLEXFIS behavior, but with different geometric issues concerning the interpolation.

So, the focus in this experiment is on how different interpolation and growing strategies perform depending on the dimension of the input space and the AS-MRA growing strategy is also compared to an AS-MRA with fixed layer depth. This allows to roughly rate the AS-MRA growing strategy as a top down or bottom up approach and to determine whether the growing strategy only helps to minimize memory and storage demands or also affects the performance of the learning system.

The results for this experiment report the CL for each considered learning system and input dimension in Table 4. A priori, the development of the CL according to the input dimension follows no obvious monotony because the approximation becomes inherently more complex as the input dimension increases and the overall complexity of the target function increases as well. On the other hand, the increased target complexity in terms of number of parameters also affects the output distribution of the target function as the product fosters concentration towards zero and the signed n-th root pushes the normalized values toward -1 and 1.

The prediction performance of the constant zero prediction is used to rate the overall performance of the compared methods. This reference allows to rate the prediction performance across different in-

Table 4: Mean Cumulative Loss results for different learning systems and input dimensions for learning random polynomial target functions. The prediction performance of the learning systems is further compared to a zero prediction as worst case performance. The best results for each input dimension are marked in bold.

Input Dimension n	FLEXFIS	FIMTDD	Ensemble AS-MRA	Growing AS-MRA	Zero prediction
1	122.1	72.2	**38.3**	402.1	1480.2
2	211.7	578.1	**102.8**	581.3	1031.8
3	357.0	748.5	**213.6**	664.8	865.4
4	535.4	805.6	**476.4**	714.1	785.3
5	659.5	786.6	**650.6**	866.4	706.7
6	682.3	794.4	**674.8**	879.7	684.9

Table 5: Mean Cumulative Loss results for different layer depths of a combined AS-MRA and input dimensions. The best results for each input dimension are marked in bold.

Layer	Input dimension n					
Depth	1	2	3	4	5	6
1	914.1	876.8	819.2	772.1	708.8	**683.2**
2	282.6	481.5	583.8	658.8	**679.5**	698.1
3	55.7	298.1	474.4	619.0	680.6	777.4
4	41.2	144.4	275.9	**477.5**	706.2	822.2
5	**38.3**	104.6	**216.7**	489.9	713.2	828.1
6	38.7	**103.9**	252.7	516.4	714.6	828.1
7	39.8	113.7	276.3	517.2	714.6	828.1
8	41.4	127.8	278.0	517.2	714.6	828.1
9	43.1	137.7	278.0	517.2	714.6	828.1
10	47.0	139.3	278.0	517.2	714.6	828.1

put dimensions and reflects how the shape of the target functions behaves with respect to output distribution, because the more the product inside the geometric mean forces the output of the target to zero, the better the zero prediction becomes. The decreasing trend of the zero prediction CL in Table 4 reflects this dominance of the product term. So, the target function becomes more complex as the input dimension increases, but the more complex structures tend to define smaller details around a generally vanishing output. This makes it even harder for the compared incremental learning systems to outperform the zero prediction CL.

Comparing the results for the learning systems in Table 4 reveals one dominant approach, i.e. the Ensemble AS-MRA. For the one dimensional target function FIMTDD performs second best and for higher input dimensions FLEXFIS is second best. The performance of the Growing AS-MRA is worst for one and two dimensional targets and only slightly outperforms FIMTDD for higher input dimensions. FIMTDD even fails to outperform the CL of the zero prediction at input dimension four, where the performance of FLEXFIS and the Ensemble AS-MRA begin to resemble each other.

Looking at the individual performance development of the different approaches with respect to a growing input dimension reveals some interesting findings. The CL of all considered approaches increases monotonically as the input dimension increases. The performance of FIMTDD strongly drops when increasing the dimension from one to two which renders the growing process for trees inherently more complex. The performance of FLEXFIS decreases more steadily, similar to the one of the Ensemble AS-MRA. The performance of the Growing AS-MRA also decreases slowly, but at a much lower level due to the high CL values which nearly match the zero prediction performance as the input dimension increases.

All in all, no growing strategy is able to outperform the Ensemble AS-MRA in this setting and the behavior of the Growing AS-MRA more resembles the FIMTDD performance than the one of FLEXFIS. This finding also explains the overall poor prediction performance of the Growing AS-MRA as it requires a certain amount of samples to build a layer structure which is capable of representing the target function at least roughly. This initial amount of samples is mostly wasted as its information content is only exploited in a shallow layer structure which ignores target details.

The Ensemble AS-MRA exploits all learning samples the same way right from the start. In summary, the growing strategy of the AS-MRA resembles the growing of trees with similar poor performance, while the Ensemble AS-MRA is superior not only because it does not need to decide about structural components, but also due to its full exploitation of the information content of each learning sample. Its performance becomes similar to the one of FLEXFIS and thus, resembles a bottom up growing strategy without actually altering the layer structure and only relying on the inherent regularization of the learning architecture.

4.6 on-line system identification

The investigations considered so far cover nearly all aspects of the AS-MRA as an incremental on-line learning system. This section considers the performance of the combined AS-MRA for on-line identification of several benchmark dynamic systems from [141] and the design effort for using AS-MRA as a model learning tool. The growing and fixed size AS-MRA variants are compared as they differ with respect to design effort and prediction performance. The modeling aspect in system identification is crucial in many domains and especially in engineering and embedded systems contexts because the resulting models allow filters to distinguish noise and actual system dynamics. Controllers benefit the same way because they can take their acts according to the system dynamics in order to achieve their goals faster and more efficiently. However these additional topics related to on-line system identification are not considered here as the focus is on incremental learning systems.

The learning scenario in system identification is inherently different from all other tasks considered here before because the observed and modeled dynamic system always follows a path through its state space. The learning samples originate from this very same trajectory through the input space and therefore, the samples are not independently identically distributed (i.i.d.). The sample distribution always and inherently impacts the behavior of any incremental learning system, but considering a setting without i.i.d. samples is special because the i.i.d. assumption is key to many convergence proves in order to generalize from one learning step to the whole sequence.

Moreover, [56] coins the term *persistence of excitation* which also relates to the sample distribution for modeling dynamic systems in an on-line learning setting. Basically, the concept requires the sample distribution to sufficiently cover the state space such that the learned model accurately fits the system dynamics and not only memorizes and represents the learning samples. The *persistence of excitation* points to a conflict in applying on-line system identification which is as fundamental as the stability-plasticity-dilemma. This conflict is easily illustrated considering an embedded system which employs an adaptive controller by means of on-line system identification. The controller has the goal to stabilize the system in a well-defined state. In order to get there it requires an accurate or at least plausible system model.

The on-line system identification requires highly representative samples about the system dynamics in order to build a plausible and accurate model. This goal becomes conflicting once the model is accurate enough for the controller to stabilize the system, because a stable system only produces samples around one state and even high amounts of such samples contain no further information for the system identification. In such a case, the model is only accurate in a close vicinity around the stabilized state but not across the whole state or input space.

The conflicting goals of the controller and system identification represent one instance of the exploration-exploitation-dilemma. A standard solution for the system identification perspective is to use test signals which excite most or even all dynamics of the observed system before employing the actual controller. This test signal setting is also used here for identifying the benchmark systems. Like in the previous experiment, there is an inherent reference for rating the prediction performance of the learned model, i.e. the steady prediction which always outputs the last observed state of the system as a prediction for the next state. This steady prediction is very simple as it ignores the dynamics of the system. However, even if this assumption imposes a serious simplification, its prediction performance is usually remarkably accurate and therefore challenging to bet by learning systems on the short run.

System 1: $u_t \sim \mathcal{U}(-1,1)$

$$s_{t+1} = 0.3s_t + 0.6s_{t-1} + 0.6\sin(\pi u_t) + 0.3\sin(3\pi u_t) + 0.1\sin(5\pi u_t) \tag{90}$$

System 2: $u_t \sim \mathcal{U}(-2,2)$

$$s_{t+1} = \frac{s_t s_{t-1}(s_t + 2.5)}{1 + s_t^2 + s_{t-1}^2} + u_t \tag{91}$$

System 3: $u_t \sim \mathcal{U}(-2,2)$

$$s_{t+1} = \frac{s_t}{1 + s_t^2} + u_t^3 \tag{92}$$

System 4: $u_t \sim \mathcal{U}(-1,1)$

$$s_{t+1} = \frac{s_t s_{t-1} s_{t-2} u_{t-1}(s_{t-2} - 1) + u_t}{1 + s_{t-1}^2 + s_{t-2}^2} \tag{93}$$

System 5: $u_{i,t} \sim \mathcal{U}(-1,1), i = 1,2$

$$\mathbf{s}_{t+1} = \begin{pmatrix} \frac{s_{1,t}}{1+s_{2,t}^2} \\ \frac{s_{1,t}s_{1,t}}{1+s_{2,t}^2} \end{pmatrix} + \begin{pmatrix} u_{1,t} \\ u_{2,t} \end{pmatrix} \tag{94}$$

System 6: $u_t \sim \mathcal{U}(-1,1)$

$$s_{t+1} = 0.8s_t + (u_t - 0.8)u_t(u_t + 0.5) \tag{95}$$

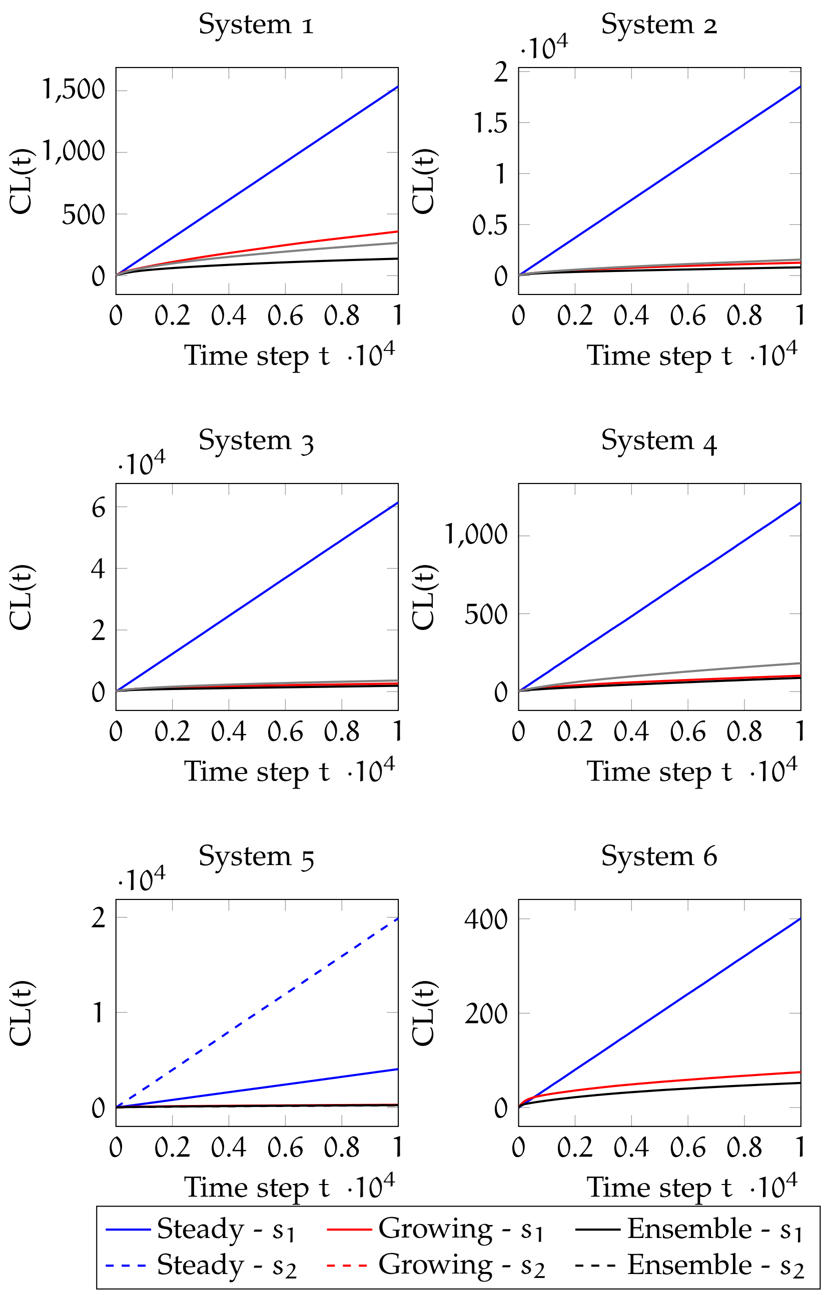


Figure 30: These six plots show the course of the CL for learning the behavior of the corresponding dynamic systems in [141] using two variants of the combined AS-MRA together with the CL performance of the steady prediction as a base line reference. Each plot shows the result of the steady prediction as a blue line and the prediction performance of the growing and fixed AS-MRA as a red and black line, respectively.

The benchmark system definitions in [141] include test signals for each system. The complete definitions are summarized in equations (90) to (95) as an overview. These systems and their corresponding test signals are used here to generate learning tasks comprising 10000 samples. The instances x_t always represent the full state of the system s_t together with the exciting test signal u_t. In addition to that, delayed versions of states s_{t-k} and signals u_{t-k} are included if they are necessary to represent the actual system dynamic according to the definitions in equations (90) to (95).

The corresponding label y_t is the resulting next state s_{t+1} disturbed by a small uniformly distributed noise of width 0.02. The learning samples here are nearly perfect in order to focus on their trajectory nature. As the focus here is on design effort two versions of the combined AS-MRA rival each other and the steady prediction in these experiments. There is an AS-MRA with a fixed layer depth of ten using the ensemble evaluation approach as the first learning system henceforth referred to as Ensemble AS-MRA. The other AS-MRA version follows the growing strategy without further prior knowledge about tolerance parameters or other guiding parameters. This second version is referred to as Growing AS-MRA and this version represents the out-of-the-box prediction of the AS-MRA approach. The results over the whole course of the experiments are displayed in Figure 30 as mean values over 1301 repetitions. In addition to that, table 6 shows the final CL results together with their standard deviation.

The values in each column in table 6 span three orders of magnitude for the different systems. This reflects that each system behaves in its own way with respect to dynamics and learning complexity. In general, the dynamics of the systems are represented by the steady prediction results while the CL values for the learning systems relate to the learning complexity.

Comparing the results for the three approaches shows that the learning systems outperform the steady prediction, but looking at the course of the CL in Figure 30 the necessary amount of training samples varies. Figure 31 highlights the relevant details for each system by focusing on sample demand which is required to outperform the steady prediction. Looking at the performance of the two learning systems the ensemble variant clearly performs better than the growing approach although the difference is relatively small for systems 4 and 5.

Another remarkable detail here is the difference in CL ordering between steady prediction, Growing AS-MRA and Ensemble AS-MRA. The results for the steady prediction correspond to the inner dynamics of each system while values for the learning variants rather reflect the learning complexity. Systems 3,4 and 6 show a consistent ranking across all considered approaches. The two AS-MRA variants only dif-

Table 6

System	Steady Pred.	Growing AS-MRA	Ensemble AS-MRA
1	1536 ± 46.53	358 ± 33.95	138 ± 18.03
2	18546 ± 4135	1264 ± 1246	807 ± 58.66
3	61440 ± 20907	2610 ± 3437	1888 ± 2117
4	1215 ± 44.32	102 ± 14.92	88 ± 12.52
5 (s_1)	4037 ± 1074	290 ± 34.24	250 ± 25.55
5 (s_2)	19887 ± 5396	254 ± 38.33	232 ± 29.54
6	401 ± 25.71	75 ± 14.01	52 ± 10.61

fer in ranking of system 1 while the steady prediction also varies with respect to ranking of system 2.

The different ranking of system 2 does not allow particular insights about the compared approaches but rather points to a general issue in learning dynamic systems. Generating learning samples by observing dynamic systems yields an inherently biased sample distribution due to the system trajectory and attractive steady states which are overrepresented in the samples. Thus, relevant parts of the dynamics are likely to be underrepresented in the samples. This is especially crucial with respect to learning if the behavior of the system near steady states is simple compared to the underrepresented regions of the state space. In such a case the learned model tends to follow the simple steady state dynamics while ignoring the rarely seen complex parts.

The difference between the ranking of system 1 and 5 actually relates to the different approaches represented by Growing AS-MRA and Ensemble AS-MRA. System 1 follows a more complex non-linear behavior with respect to the input u_t compared to system 5 which is linear in u_t. According to the ranking of the CL values, Growing AS-MRA performs better on system 5. This finding is in line with the overall capabilities of AS-MRA structures with limited layer depth as investigated in section 4.3.2. The non-linear dynamics require a certain layer depth to be properly representable. This necessary layer depth is available in Ensemble AS-MRA right from the start while Growing AS-MRA needs to consume samples to build these layers and fine tune their output. Thus, in principle both approaches can represent the non-linear dynamics on the long run, but Ensemble AS-MRA is more sample-efficient and therefore not only outperforms Growing AS-MRA in general, but also handles non-linear system behavior right from the start of learning.

In this on-line system identification scenario the learning sample demand directly maps to learning time and model accuracy development over time. One extreme case in this trade-off is a non-learning expert system entirely build from prior knowledge which does not require any learning, but demands huge design effort. On the other hand, a learning system with literally no prior knowledge like the Growing AS-MRA faces a huge sample demand in order to find an accurate model. The Ensemble AS-MRA introduces only one design parameter, i.e. the layer depth and shows a much higher sample efficiency compared to Growing AS-MRA. Most realistic use cases allow to guide the learning by some form of rough prior knowledge about expected complexity or necessary accuracy, thus the sample demand can be reduced by fixing the layer depth and defining accuracy related convergence conditions.

All in all, both AS-MRA variants require only a very limited amount of samples in order to come up with a model that predicts the system behavior more accurately than the steady prediction. This initial learning phase is detailed in Figure 31 using individual time lines for each benchmark system. Moreover, comparing the learning behavior of Growing AS-MRA and Ensemble AS-MRA allows to learn about the impact of prior knowledge on a general and abstract design level and directly visualizes the tradeoff between prior knowledge and sample demand, i.e. learning time.

For systems 2 and 3 the learning behavior of both AS-MRA is very similar with respect to their prediction performance compared to the steady prediction. In system 2 both variants perform better right from the start and for system 3 both variants require about 20 samples to outperform the steady prediction CL. Slight differences in learning behavior between Growing AS-MRA and Ensemble AS-MRA can be observed for systems 4 and 5 where the Ensemble AS-MRA variant only requires about 30 or 40 samples to outperform the steady prediction, respectively. On the other hand, Growing AS-MRA requires about 45 or 70 samples to do so, respectively. This difference in sample demand to outperform the steady prediction is even more obvious in system 1 where Ensemble AS-MRA only needs about 30 samples while Growing AS-MRA needs more than 100 samples. This absolute difference in sample demand is even higher in system 6 where Ensemble AS-MRA requires around 190 samples while Growing AS-MRA requires 550.

The differences in sample demand illustrate the collaboration across refinement layers in the Ensemble AS-MRA approach and the corresponding sample exploitation. In order to do so, Ensemble AS-MRA requires a fixed layer depth as prior knowledge. Therefore, to some degree this comparison also illustrates the power of prior knowledge to reduce sample demand even if such prior knowledge is stated on an abstract level and does not directly relate to actual system behavior. This kind of prior knowledge rather allows to select a certain approach which allows for higher sample efficiency than to actually reduce sample demand by introducing pre-modeled system behavior. This aspect of sample efficiency is especially important in nonstationary learning tasks as there is an inherent lack of prior knowledge about system behavior.

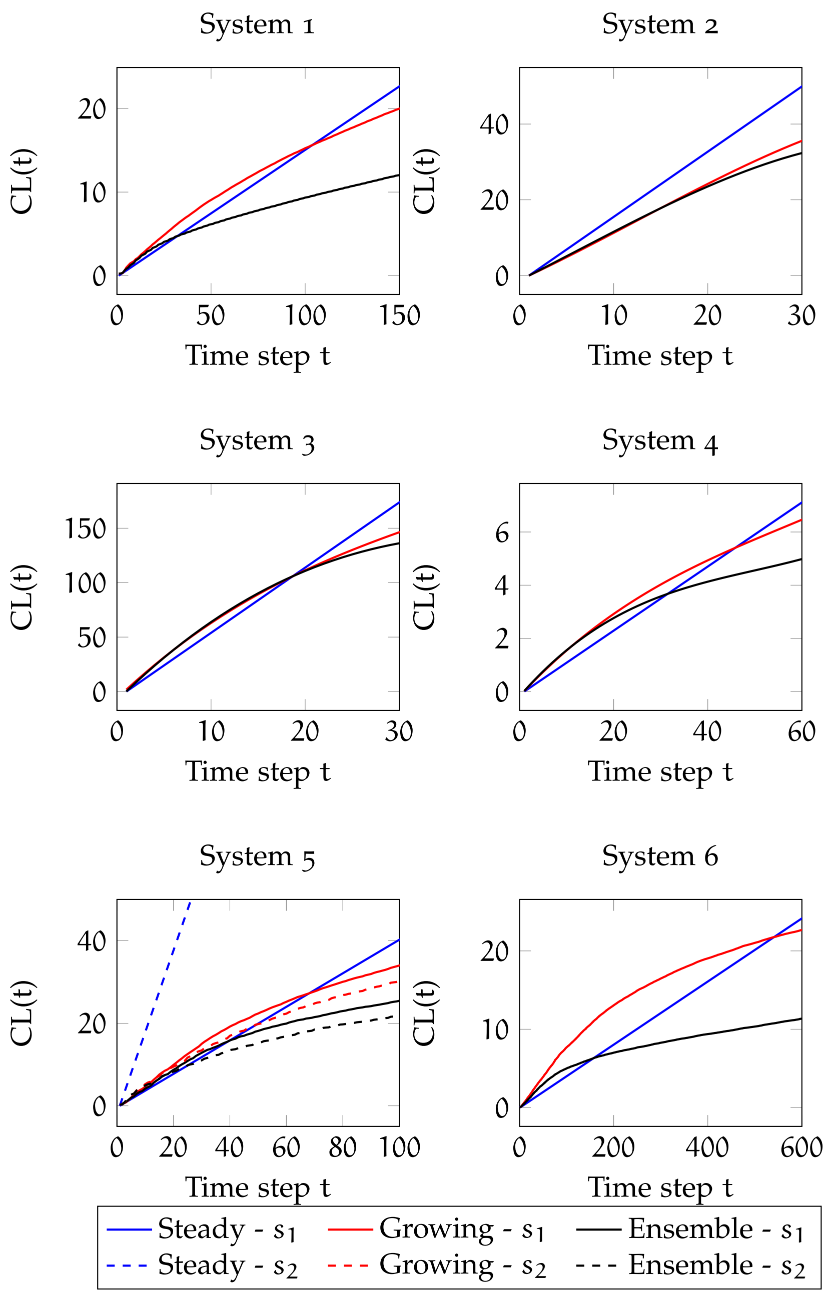


Figure 31: These six plots show individual parts of the development of the CL for learning the behavior of the corresponding dynamic systems in [141] using two variants of the combined AS-MRA together with the CL performance of the steady prediction as a base line reference. Each plot shows the result of the steady prediction as a blue line and the prediction performance of the Growing and Ensemble AS-MRA as a red and black line, respectively.

5 CONCLUSION

5.1 DISCUSSION

A linear discussion of the AS-MRA is inherently difficult as the whole approach consists of three parts which all need to come together, namely the input segmentation and interpolation, the learning architecture and the differential parameter encoding. The used on-line learning algorithm is also important and needs to be able to handle time-variant target functions, but there are different variants for this task and the VL-RLS algorithm presented here is only one of them. These four parts contribute to the AS-MRA and only some of them are applicable in different settings. The most specific part of the AS-MRA is the differential parameter encoding across subsequent layers, but it is essentially bound to the used learning architecture. In principle, the differential encoding is applicable in other contexts as well, but only in combination with the additive layer structure and learning architecture it unfolds its potential to convert the inherently time-variant intermediate target functions which interface subsequent layers into widely stationary targets for each layer. The used learning architecture is rather general as it only states additive refinement layers and defines the residual of each layer as the target for its successor.

As already mentioned, the differential encoding allows to use this learning architecture without dealing with time-variant residual target functions, but this is only possible in an efficient way due to the simplicial input segmentation of each layer. The strictly local influence of each parameter in every layer allows to limit the number of references between the parameters of subsequent layers. The simplicial input segmentation allows referring to at most three parameters from a preceding layer for the combined AS-MRA, for the aligned AS-MRA the limit is two and one for the shifted AS-MRA. This way the differential encoding is fast to evaluate and sticks to the structure of the refinement layers, which provides stationary targets for learning without affecting the scalability of the approximation structure.

The most general part of the AS-MRA is the simplicial input segmentation combined with the smooth interpolation scheme. This approximation structure is linear in the parameters and can be combined with standard on-line learning algorithms to form an on-line learning system. And still, the gain of having a smooth local approximation structure is very little in high resolution grids due to approximation artifacts, e.g. a poorly modeled derivative at each node. The additive layer architecture mitigates these artifacts. So, the differential parameter encoding supports the learning architecture, but is only efficient due to the input segmentation, the input segmentation requires the layer architecture for artifact mitigation and the layer architecture would not be necessary without the input segmentation but would also be infeasible without the differential encoding.

The novel and unique aspect of theAS-MRA is that these three components come together in harmony and allow for additive function decomposition in on-line learning scenarios even when facing time variance. Compared to the close union these parts form, the used on-line learning algorithm and the radial basis function for the simplicial interpolation scheme are only loosely coupled details. However, any on-line learning algorithm used in combination with AS-MRA needs to be able to handle non-stationary targets as the differential encoding is not perfect with respect to the smooth interpolation and the radial basis function governing this interpolation needs to meet certain boundary conditions. The rest of the discussion relates the AS-MRA approach to the design and performance aspects stated in the introduction and also includes the findings of the investigations. This short overview of the core components of the AS-MRA frames further discussion topics. Key features of the AS-MRA relate to individual components, but this does not imply that they are more important for the overall approach than others.

5.1.1 *Design effort*

The overall goal of on-line learning systems is to enhance the design of intelligent systems while reducing the demands for designing and developing them. The fundamental metrics in every design process are effort and outcome. The outcome inherently depends on the prediction performance of the chosen learning approach and thus, its ability to learn an unknown target from sequential learning samples. This is especially true for learning and tracking non-stationary targets, but here the learning can only balance between tracking performance and noise reduction which need to be prioritized by application specific requirements. The design effort depends on different independent categories.

Firstly, there is an effort in specifying the learning scenario for the particular application which comprises the number and kind of inputs and targets for the on-line learning system. The whole field of dimensionality reduction is dedicated to support and mitigate this kind of design effort, but since this topic is not part of this work, the second best option is to have an on-line learning system which scales linear towards input dimensionality with respect to computational demands as this relaxes design decisions which affect the number of inputs. The next aspect of design effort is the configuration and tuning of the chosen on-line learning approach for the selected scenario. This task gets increasingly complex as the involved parameters which are to tune are very specific for the selected approach and only indirectly relate to the underlying problem defined by the learning scenario.

The computational demands of the fully specified and configured approach directly link to the hardware effort for actually employing the designed solution. The hardware effort may be considered low if the approach can reasonably perform on standard hardware components, but there are learning approaches which only operate efficiently, especially with respect to energy consumption, on specially designed hardware which greatly increases the design effort.

Another aspect of design effort is the integration of prior knowledge into the chosen approach. The key questions here are whether the prior knowledge is representable in the approach at all and how difficult it is to map prior knowledge to the concepts and parameters the approach offers. The presented AS-MRA approach adds unique features to some of these design issues and contributes to all of them. The computational demands of the AS-MRA scale linearly regarding input dimension and layer depth, by design and as measured in Figure 27. This feature is unique as the approximation of the AS-MRA is globally smooth and linear in the parameters at the same time.

Hence, the AS-MRA provides a smooth on-line learning system with linear scalability as targeted in the goal definition from the introduction. Moreover, there are no mandatory design parameters which require prior knowledge about the application or certain target function properties. When using the growing strategy for the AS-MRA the only implicit assumption taken by the approach is to detect overfitting as an increasing prediction error in the last three layers. Thus, there is virtually no design effort for tuning the AS-MRA and selecting a learning scenario is comfortable due to the linear scalability.

In fact, the AS-MRA provides a number of optional design parameters which allow to restrict the resource consumption of the AS-MRA with respect to computational and memory demands. But there are also optional parameters which are directly related to the expected prediction accuracy and thus, the performance for the application. These kinds of parameters are not unique as FIMTDD and other tree-based methods provide similar capabilities, but only the AS-MRA realizes these parameters due to a layered learning architecture which allows to learn at different scales simultaneously and therefore enables a fast and instantaneous refinement, if necessary. And this without the need to deduce optimal split boundaries.

Hence, the learning within AS-MRA is performance orientated as in tree-based approaches, but without the structure identification which needs to estimate non-linear parameters of the approximation structure. The only structural non-linear parameter the AS-MRA provides is the layer depth which can be flexibly adjusted while safely detecting overfitting. However, the overfitting detection mechanism also prevents the AS-MRA to escape from underfitting if the structures of the target function skips three orders of magnitude in binary refinement.

The unique feature the AS-MRA provides with respect to design parameters is the relative tolerance for convergence detection. Comparing the prediction performance of approximations with different input resolutions is only possible in a layered learning architecture as even tree-based approximation structures handle their models in leaf nodes only. A similar approach would be to run multiple instances of the same learning system with different input resolutions in an ensemble, but in such a case the fine grained input resolutions would not benefit from the generalization power of the coarser ones.

This directly relates to the bias-variance-dilemma as the prediction performance of the individual ensemble members is much harder to compare due to their different expressiveness. In addition to that, the fine grained resolution members would suffer from approximation artifacts due to local interpolation or an increased scalability when using local linear models. So, the connection and cooperation of the refinement layers in AS-MRA allows for a valid comparison of the prediction performances of all layers. The cooperation of layers in the learning architecture also gives rise to the central properties of the learning behavior of the AS-MRA which are discussed in the next section.

5.1.2 *Learning Properties*

The learning properties of the AS-MRA mainly originate from the layered learning architecture which inherently acts as a regularization for unnecessary expressiveness in the layer structure. This property stems from the propagation of residuals to subsequent refinement layers which naturally vanish as the approximation becomes accurate. But the learning architecture also affects the generalization in terms of extrapolation and interpolation and how a single learning sample is stored and represented in the layer structure.

The generalization in AS-MRA is as complex as the depth of the layer structure and therefore there are two fundamentally different cases to look at, the one with a fixed layer depth and the one using a growing strategy for the layer structure. Although both approaches use the same AS-MRA structure, their behavior is fundamentally different with respect to how they treat and handle learning samples. Using the terms lazy and eager learning here would be misleading, because those refer to well-defined concepts in machine learning which do not directly apply here. However, with respect to the original meaning of lazy and eager learning in machine learning, the AS-MRA is an eager learning approach in all its variants and irrespective of the potential use of growing strategy. The difference in handling a single learning sample in a fixed or growing layer structure boils down to the question of how much information from a sample needs to be stored and how much can be stored.

The fixed layer structure is truly eager even in this regard as it always processes and learns each sample in all layers. This way, each learning sample contributes to the overall output on all input resolutions, i.e. the impact of the sample is considered on a fully global scale as well as in a maximally local vicinity. This fully exploits the information gain in each sample on all user-defined scales.

In contrast to that, the growing strategy is rather conservative as it always asks for the relevance of considering an additional refinement layer while learning form a particular sample. So, the growing strategy stops exploiting the information gain from a sample by the time it is convinced to meet a convergence criterion. This is a reasonable approach on the long run, but it ignores valuable information from samples during the initial learning phase. In this phase the layer structure has not reach an appropriate depth and therefore is not capable of following certain details represented by the samples. This is also crucial facing a shift in the target function which increases the target complexity.

The restrictive information exploitation of growing layer structures in principle helps to become robust against noise, but judging signal from noise is only possible retrospectively and with having a sufficient amount of samples. Further, the learning architecture already fosters noise reduction by means of regularization. On the long run fixed and growing layer structures become the same due to self-regularization, growing and pruning. So, the further discussion concerning learning properties will focus on layer structures of fixed size.

The already mentioned multi resolution storage of learning samples in the layer architecture also affects its generalization properties. The top layers have only limited expressiveness but global influence and therefore are able to provide plausible extrapolation from a limited number of samples. For this very reason information from the very first samples and especially the single first sample is extrapolated across the entire input space. The global extrapolation is risky when dealing with noisy data, but since all samples contribute to the single parameter in the base layer this parameter converges as fast as possible to the global mean. So, the simplicity of the top layers allows them to perform plausible extrapolations while converging rapidly and thus canceling out noise which fortifies their extrapolation and supports subsequent layers.

In addition to that, each layer extrapolates its whole output to its successor as a prior initialization due to the residual propagation in the learning architecture. Thus, the extrapolation from the top layers is horizontally across the input space and vertically across the refinement layers. On the other hand, the usage of the extrapolation of the top layers is avoided whenever possible, because the refinement layers override the output of the preceding layers and the bottom layer rules the output. Hence, the layer structure always makes use of the most local information that is available while assuming it to be most relevant for the query instance x_t.

This procedure relies on the sparse parameter representation in the AS-MRA which only creates a parameter in memory if there is a learning sample contributing to it. The resulting evaluation routine automatically stops if there are no more parameters in the subsequent layer and therefore the output is governed by as much top layer extrapolation as necessary and as much relevant local information from refinement layers as possible. Further, this connects to the bias-variance decomposition as the top layers tend to suffer from bias while the bottom layers suffer from variance. Balancing both for the evaluation aims at minimizing the approximation error and the AS-MRA offers unique capabilities in doing so by combining cooperatively generated approximation on different layers.

The counterpart to extrapolation is interpolation, but for the AS-MRA this aspect is more related to properties of the approximation structure than to learning, because interpolation in local approximation structures with RLS-based learning is very similar for all local approximation structures. The strength of a local approximation structure is to protect locally acquired knowledge from being disturbed from noise or time-variance which applies to distant regions of the input space. The fundamental drawback is the limited extrapolation power.

The learning architecture of the AS-MRA allows to benefit from the strength of a local learning architecture without suffering from its drawbacks. But there is a slight restriction to the protection of local knowledge as the differential encoding is intentionally not perfect. Therefore, changes in higher layers partly affect the output of lower layers. But the changes are minimal and the affected layer can adapt to them as they are trained sequentially, so the subsequent layers can compensate the structurally induced changes according to the current learning sample.

As mentioned above, further details concerning the interpolation are related to approximation structure properties and thus, handled in the following subsection.

5.1.3 *Approximation structure properties*

The layer architecture of the AS-MRA stems from a local approximation structure for each layer and originates from a geometrical motivation to circumvent the scalability issues grid-based local approximation structures usually suffer from. The interpolation in AS-MRA in principle is the same as in other local approximation structures and combines certain aspects of RBF functions and Simplicial B-Splines.

The whole interpolation scheme comprises the simplicial input segmentation and the smooth radial basis functions which need to decrease from one to zero along the unit interval. The exact shape of the interpolation function is basically user-defined, but the vanishing derivatives of this function at the boundary define the overall degree of smoothness the interpolation scheme achieves and therefore at least the first derivatives should vanish. The used polynomial interpolation function $g(x)$ in equation (16) only fulfills these minimal requirement.

This function is used as a radial basis function on a variable norm which is specially tailored for the used symmetric simplicial input segmentation. Without the symmetry of the input segmentation, there would not exists such a norm as it exploits the geometry of the support of the corresponding basis functions in the input segmentation. Thus, the whole AS-MRA approach is inherently geometrical as only the highly symmetric geometry of the input segmentation allows the definition of a globally smooth interpolation on a simplicial grid. And the smoothness of the underlying approximation structure gives rise to the whole AS-MRA including the learning architecture and the differential encoding.

The simplicial geometry of the input segmentation is what brings everything else together and especially ensures the linear scalability of the approach. So, the contribution of this thesis with respect to interpolation is the definition of the norm which allows to use radial basis functions with limited support for a strictly local interpolation and the blending between two such norm-based terms in order to fully respect the geometry of the simplicial interpolation. The resulting smooth basis functions with strictly local support are the foundation for the layer architecture built on top.

The layer structure of the AS-MRA combines aspects of Fourier Series and Wavelets without requiring the basis functions ϕ_i to form an orthonormal basis. The aligned and shifted variants of the AS-MRA resemble the cosine and sine parts of the Fourier Series. They form fully functional approximation structures on their own, but their full potential is only revealed when combining them in one approximation structure. The similarity to the Wavelets is present in the growth of expressiveness in Wavelet and refinement layers, but the admirable mathematical properties of the Wavelets are not present in the AS-MRA. The principle structure of the Wavelets also motivated the layer architecture and the differential encoding is an approach to migrate the layer structure of the wavelets to a local approximation structure equipped with basis functions that form a partition of unit.

So, layer architecture and differential encoding imitate the principle approximation concept of wavelets without meeting their formal properties. The overall approximation structure of an AS-MRA variant is formally a global one as the basis functions in the top layers span the whole input space or at least wide parts of it. So, judging from the categories and metrics defined in chapter 2 the AS-MRA is a global, smooth and linear in the parameters approximation structure. Using the Simplicial tensor product yields no partition of unity, but the variable norm-base aggregation does. Further, its computational demands scale linearly to input dimension and its storage demand scales exponentially towards input dimension.

Being linear in the parameters here does not allow to directly refer to convergence proofs, because the learning architecture and differential encoding need to be considered, too. Nevertheless, the learning in each layer is strictly linear in the parameters and therefore allows to employ standard on-line learning algorithms. I.e. standard convergence results apply to each layer and therefore, the learning architecture at least allows for a convergence of the overall layer structure in a layer-by-layer manner.

With respect to the growing strategy the approximation structure is also related to tree-based approaches and evolving fuzzy systems. When starting from a single base layer the initial situation for the AS-MRA is similar to the one of a tree-based approach, but the AS-MRA has the advantage to virtually split along all input dimensions at once and thus, only needs to consider whether to introduce an additional layer or not. An AS-MRA of fixed size rather resembles the growing procedure in evolving fuzzy systems which start from usually small local clusters and uses the incoming samples to enlarge the clusters in size and number while applying pruning and merging strategies in order to limit the total number of clusters.

In the AS-MRA this perspective maps to first represent the samples in the bottom layer and let the learning architecture erase local structures as they become obsolete due to accurate predictions in the layer above. This self-regularization of the AS-MRA resembles structure adaptation strategies of evolving fuzzy systems starting from small local clusters. But unlike structure adaptation in evolving fuzzy systems which is related to the estimation of non-linear parameters, the self-regularization only affects linear parameters due to residual propagation in the learning architecture.

5.1.4 *Time-variance*

Time-variance is an important and fundamental aspect in on-line learning, but it is essential for AS-MRA and inherently connected to it due to the differential encoding which requires the learning architecture to employ an on-line learning algorithm capable of tracking non-stationary target functions. The reason to this is the need for cooperation across subsequent layers in order to mitigate the approximation artifacts of the finer layers. A perfect differential encoding would fully decouple the individual layers and thus, only the output of the bottom layer would be effective. The resulting overall output would contain many approximation artifacts due to the vanishing slope at the nodes of the local approximation structure. The cooperation is achieved by relating the local parameters of subsequent layers, but this partly ignores the nonlinear interpolation function which shapes the basis functions ϕ_i.

This nonlinearity is not reflected in the differential encoding of the parameters and allows the different layers to cooperate with respect to the slope at the nodes of the finer layers. The corresponding derivative is defined by the combined output of all former layers. Therefore, the overall output can have non-vanishing slopes even at the nodes of the bottom layer. This successfully mitigates approximation artifacts and allows for representing a reasonable derivative of the target function. All of this is related to time-variance, because the resulting combination of differential encoding and learning architecture results in slightly non-stationary intermediate targets due to the cooperation of subsequent layers. This hinders a fast learning to some degree as time-variance inherently increases the necessary amount of learning samples to accurately learn a target function.

Coined as a feature, the AS-MRA is inherently capable of handling non-stationary target functions because it already deals with them as residual targets interfacing subsequent layers in the learning architecture. The VL-RLS algorithm used in AS-MRA is at most only a small contribution to the family of adaptive learning algorithms, because it only applies standard concepts to the vSGD algorithm from [166] and adds a heuristic for learning from scratch. The time horizon estimation from [166] is the core algorithmic component for time-variance handling and it yields an algorithm with acceptable noise reduction and good time-variance tracking as observed in Figure 28 when systematically comparing the impact of noise and time-variance onto the prediction performance. The VL-RLS is neither as fast as PA in terms of time-variance tracking nor is it as robust against noise as the standard RLS is, but it has its merits when dealing with time-variant data which is corrupted by moderate noise.

The time-variance handling of the AS-MRA is limited to each single layer and there is no propagation of estimated time horizon information across the layer structure. Introducing such a layer-wide time horizon estimation and propagation requires careful balancing of preserving valuable local knowledge in lower layers and fast adaptation in higher layers for plausible extrapolation to changing targets. The good time-variance tracking of the VL-RLS and especially its potential to rapidly shrink the estimated time horizon render the whole AS-MRA flexible enough to handle non-stationary target functions even without a coordinated time-horizon estimation. This strategy prefers the preservation of local knowledge because time-variant effects with only local impact always affect to top layers but this should not harm the knowledge in stationary regions of the input space.

5.2 SUMMARY

In this thesis on-line learning for regression on evolving data streams is considered and a new incremental learning system called Adaptive Simplicial Multi Resolution Approximation (AS-MRA) is introduced. The motivation for tackling this problem stems from practical design issues and also includes connections on a theoretical level to other fundamental problems in machine learning like classification and reinforcement learning. The related work covers common approximation structures with a focus on Linear In the Paramaters (**LIP!**) ones as those are most relevant in on-line learning due to the convex optimization problem they allow to face. The considered structures are grouped and ranked according to general properties like forming local or global approximations and the kind and number of nonlinear parameters in each approximation structure, but the focus is on scalability and smoothness, because the overview of all considered approximation structures reveals an uncovered region for smooth approximations whose memory access ratio shrinks exponentially with respect to input dimension. For a particular approximation structure, the Memory Access Ratio (MAR) compares the number of parameters which are necessary for learning and evaluation with the total number of parameters, because each parameter access is directed to memory and therefore potentially slow compared to calculations within a CPU.

The main body of the thesis is dedicated to the development of the envisioned approximation structure and the approach to get there is geometrically motivated from interpolation in simplexes, because the number of vertexes in a simplex grows linearly with respect to its dimension and therefore allows to restrict the number of necessary parameters to look at for evaluation and learning accordingly. The description of the approach starts by introducing a symmetric simplicial input segmentation and defining a smooth interpolation scheme for this segmentation.

The resulting approximation structure meets the formal smoothness and scalability requirements, but suffers from standard approximation artifacts in high resolution grid-based structures. The introduction of a layer structure and a corresponding learning architecture mitigates the approximation artifacts at the cost of non-stationary residual target functions in the learning architecture. The accordingly introduced differential encoding for the parameters of sequential refinement layers intentionally solves the time-variance problems only partly in order to preserve the desired cooperation between the layers.

The resulting smooth layer approximation structure is equipped with a variant of the RLS algorithm capable of handling non-stationary targets and completes the development of the AS-MRA as an on-line learning system. The AS-MRA is further equipped with a growing strategy for adapting the layer structure on demand and with respect to prediction performance related convergence detection. In order to avoid mandatory design parameters in using the growing strategy, there is a fixed build in overfitting detection which stops the growing of new layers if the estimated prediction error is non-decreasing in three subsequent layers.

The investigations on the AS-MRA and its individual parts start by comparing its performance to standard approximation structures of fixed size with respect to various target function properties and systematically consider the relation between target function complexity and prediction performance of AS-MRA variants with fixed layer depth. The experiments on target complexity already point out the inherent self-regularization of the learning architecture and the resulting overfitting suppression. The scalability of the computational demands is theoretically founded and empirically shown and the VL-RLS adaptive learning algorithm shows its merits for handling time-variance in moderately noisy samples.

The growing strategy and especially the prediction performance related tolerance parameter for convergence detection reveal the uniform convergence in the layered learning architecture which only loosely resembles the local structure of the target function complexity. Comparing different growing strategies from related approaches and an AS-MRA with fixed size again points to the potential of the learning architecture to cope with different targets and noise while fully exploiting the information content of each learning sample. This is also confirmed in on-line system identification experiments on benchmark dynamic systems and also underpins the minimal design effort for employing AS-MRA in an application.

In summary, the presented AS-MRA combines low design effort and high learning performance in a smooth and scalable approach. The main contributions of this thesis include the smooth interpolation scheme for symmetric simplicial input segmentations, the learning architecture for local refinement layers and the differential parameter encoding which mitigates time-variance for the residual targets in the learning architecture. The combination of these parts allows to include user-defined performance requirements in the growing strategy of the AS-MRA, while it is best to use a layer structure of fixed depth in order to fully exploit the information content in each learning sample. This allows to focus the design of the AS-MRA as an on-line learning system on available hardware resources rather than expected target function properties.

5.3 OUTLOOK

Although the presented AS-MRA is a fully operational on-line learning system there are some unconnected links to other parts of machine learning and possible extension to further increase the flexibility and performance of the approach. The most important link which is already addressed in the thesis at various points is the issue of dimensionality reduction. This topic is not inherently bound to AS-MRA, but the curse-of-dimensionality with respect to learning sample demands in high dimensions strongly requires to consider dimensionality reduction techniques. Since the resulting lower dimensional projection spaces originating from a dimensionality reduction are aimed to be densely covered by samples, the highly structured and grid-based AS-MRA comes in handy as it inherently performs no kind of dimensionality reduction and therefore is best suited for a modular design which connects reduction and learning methods. So, dimensionality reduction not necessarily needs to become a part of AS-MRA, but it can be beneficial to combine both, especially when adapting the learning architecture to dimensionality reduction algorithms.

A fully uncovered topic is to further exploit the layer structure in order to enhance the growing strategy by means of layer-specific mean parameter values and the parameter distribution in order to rate the magnitude of single parameters and to build further noise and overfitting detection mechanisms. The same is true for the time horizon estimation performed in the VL-RLS algorithm. But all of this needs to be handled carefully in order to avoid fatal forgetting.

The presented structure internally handles prediction error estimations as a sliding mean over the prediction error on incoming learning samples on the estimated time horizon. These parameters can be used to lift the current point prediction to interval outputs which reflect the estimated uncertainty about the prediction with respect to noise. A further integration of Hoeffding bounds would also include a mapping for ignorance based uncertainty, but this requires to adapt the standard counting based Hoeffding bounds to real-valued time horizon estimations. The combination of conflict and ignorance based uncertainty estimations allows to improve the evaluation of the AS-MRA to choose the most reliable output across all layers.

A more ambitious task would be to extend the AS-MRA to enable incremental learning not only for samples but also with respect to input dimensions. The highly structured input segmentation in combination with the linear scalability renders the AS-MRA a viable candidate for such an approach. This would further increase the flexibility with respect to design effort and the cooperation with dimensionality reduction techniques. For the same reason it would be reasonable to extend the AS-MRA to be able to detect irrelevant inputs which supports both dimensionality reduction and input incremental learning issues.

A APPENDIX

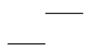
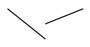
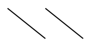
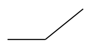
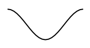
Table 7: Mean Cumulative Loss results for different approximation structures and target functions.

Target function	GLT linear	2. Order B-Spline	Legendre	Fourier	Aligned	Shifted	Combined
Step function	60.221	43.749	67.807	65.706	64.942	52.971	50.924
Linear step	16.030	13.028	19.054	11.060	20.620	37.742	14.882
Parallel lines	57.507	41.755	64.772	78.773	81.624	54.823	46.282
Kink rise	3.423	3.114	3.825	37.794	4.500	13.498	3.906
Absolute value	2.284	4.166	4.980	3.112	5.030	44.089	5.672
Kink saturation	2.395	2.017	2.669	36.478	3.390	4.977	2.641
Linear	2.048	1.654	2.288	36.027	3.026	2.150	2.225
Quadratic	3.011	4.686	4.307	4.714	12.902	28.973	9.642
Cubic	1.416	2.444	1.478	33.451	11.664	2.909	3.335
Hyperbolic	4.085	5.749	2.532	32.668	11.487	4.861	6.219
Gauss	4.385	3.048	4.040	39.726	10.977	17.288	5.445
Sigmoid	7.171	4.342	9.443	38.211	21.698	13.779	11.616
Cosine	3.780	2.814	5.411	4.000	5.564	71.885	6.748
Mexican hat	3.286	2.372	3.580	20.326	16.614	29.288	6.379
Sine	2.954	3.639	4.116	4.000	48.167	11.930	10.550

Table 8: Mean Cumulative Loss results for different approximation structures and target functions with layer depth equal to four.

Target function	GLT linear	2. Order B-Spline	Legendre	Fourier	Aligned	Shifted	Combined
Step function	60.22	43.75	67.81	65.71	55.51	33.37	31.74
Linear step	16.03	13.03	19.05	11.06	15.84	15.53	9.34
Parallel lines	57.51	41.76	64.77	78.77	54.73	30.03	29.28
Kink rise	3.42	3.11	3.83	37.79	3.29	6.01	3.13
Absolute value	2.28	4.17	4.98	3.11	4.41	14.30	3.80
Kink saturation	2.40	2.02	2.67	36.48	2.29	3.21	2.23
Linear	2.05	1.65	2.29	36.03	1.95	2.24	1.93
Quadratic	3.01	4.69	4.31	4.71	5.73	10.35	4.81
Cubic	1.42	2.44	1.48	33.45	1.97	2.75	1.74
Hyperbolic	4.09	5.75	2.53	32.67	4.53	4.78	3.46
Gauss	4.39	3.05	4.04	39.73	4.25	5.06	3.49
Sigmoid	7.17	4.34	9.44	38.21	13.39	6.06	5.42
Cosine	3.78	2.81	5.41	4.00	4.94	14.04	4.84
Mexican hat	3.29	2.37	3.58	20.33	5.38	5.90	3.23
Sine	2.95	3.64	4.12	4.00	7.08	8.06	4.85

Table 9: List of the formal definitions of the simple target functions used in section 4.3.1. All function $f(x)$ map from the unit interval $[0, 1]$ to $[-1, 1]$, i.e. $f : [0, 1] \to [-1, 1]$.

Target function	**Formal definition**	**Pictograph**
Step function	$2 \cdot \mathbb{1}_{x>0.5} - 1$	
Linear step	$(1 - 4x) + \mathbb{1}_{x>0.5} \cdot (6x - 2)$	
Parallel lines	$(1 - 4x) + 2 \cdot \mathbb{1}_{x>0.5}$	
Kink rise	$(4x - 3)\mathbb{1}_{x>0.5} - 1$	
Absolute value	$4\,\lvert x - 0.5\rvert - 1$	
Kink saturation	$3x - 1 - \mathbb{1}_{x>0.5} \cdot (2x - 1)$	
Linear	$2x - 1$	
Quadratic	$8(x - 0.5)^2 - 1$	
Cubic	$8(x - 0.5)^3$	
Hyperbolic	$\frac{2.2x}{x+0.1} + 1$	
Gauss	$2\exp(-16x^2) - 1$	
Sigmoid	$\tanh(10(x - 0.5))$	
Cosine	$\cos(2\pi x)$	
Mexican hat	$1 - 22.08x^2 \exp(-8x^2)$	
Sine	$\sin(2\pi x)$	

[1] C. C. Aggarwal. *Data streams: models and algorithms*, volume 31. Springer Science & Business Media, 2007.

[2] E. Almeida, C. Ferreira, and J. Gama. Adaptive model rules from data streams. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 480–492. Springer, 2013.

[3] P. Angelov. An approach for fuzzy rule-base adaptation using on-line clustering. *International Journal of Approximate Reasoning*, 35(3):275–289, 2004.

[4] P. Angelov and D. Filev. Simpl_ets: A simplified method for learning evolving takagi-sugeno fuzzy models. In *Fuzzy Systems, 2005. FUZZ'05. The 14th IEEE International Conference on*, pages 1068–1073. IEEE, 2005.

[5] P. P. Angelov and D. P. Filev. An approach to online identification of takagi-sugeno fuzzy models. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):484–498, 2004.

[6] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno. Early drift detection method. 2006.

[7] D. Basak, S. Pal, and D. C. Patranabis. Support vector regression. *Neural Information Processing-Letters and Reviews*, 11(10):203–224, 2007.

[8] M. Basseville, I. V. Nikiforov, et al. *Detection of abrupt changes: theory and application*, volume 104. Prentice Hall Englewood Cliffs, 1993.

[9] M. Z. A. Bhotto and A. Antoniou. Robust recursive least-squares adaptive-filtering algorithm for impulsive-noise environments. *IEEE Signal processing letters*, 18(3):185–188, 2011.

[10] G. Biau, L. Devroye, V. Dujmović, and A. Krzyżak. An affine invariant k-nearest neighbor regression estimate. *Journal of Multivariate Analysis*, 112:24–34, 2012.

[11] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 443–448. SIAM, 2007.

[12] A. Bifet, G. Holmes, B. Pfahringer, and E. Frank. Fast perceptron decision tree learning from evolving data streams. *Advances in knowledge discovery and data mining*, pages 299–310, 2010.

[13] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 139–148. ACM, 2009.

[14] P. Binev, A. Cohen, W. Dahmen, R. DeVore, and V. Temlyakov. Universal algorithms for learning theory part i: piecewise constant functions. *Journal of Machine Learning Research*, 6(Sep):1297–1321, 2005.

[15] M. Birattari, G. Bontempi, and H. Bersini. Lazy learning meets the recursive least squares algorithm. In *Advances in neural information processing systems*, pages 375–381, 1999.

[16] P. Bloomfield. *Fourier analysis of time series: an introduction*. John Wiley & Sons, 2004.

[17] M. Blum. Fixed memory least squares filters using recursion methods. *IRE Transactions on Information Theory*, 3(3):178–182, 1957.

[18] Y. Bodyanskiy, N. Lamonova, I. Pliss, and O. Vynokurova. An adaptive learning algorithm for a wavelet neural network. *Expert Systems*, 22(5):235–240, 2005.

[19] G. K. Boray and M. D. Srinath. Conjugate gradient techniques for adaptive filtering. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 39(1):1–10, 1992.

[20] O. Bousquet and L. Bottou. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

[21] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[22] W. L. Briggs and V. E. Henson. *The DFT: an owner's manual for the discrete Fourier transform*. SIAM, 1995.

[23] E. Brodsky and B. S. Darkhovsky. *Nonparametric methods in change point problems*, volume 243. Springer Science & Business Media, 2013.

[24] D. S. Broomhead and D. Lowe. Radial basis functions, multivariable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[25] D. Brzezinski and J. Stefanowski. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):81–94, 2014.

[26] A. Buschermöhle. Reliable on-line machine learning for regression tasks in presence of uncertainties. 2014.

[27] A. Buschermöhle and W. Brockmann. On-line learning with minimized change of the global mapping. *Evolving Systems*, 6(2):131–151, 2015.

[28] E. Cambria, G.-B. Huang, L. L. C. Kasun, H. Zhou, C. M. Vong, J. Lin, J. Yin, Z. Cai, Q. Liu, K. Li, et al. Extreme learning machines [trends & controversies]. *IEEE Intelligent Systems*, 28(6):30–59, 2013.

[29] R. M. Canetti and M. D. España. Convergence analysis of the least-squares identification algorithm with a variable forgetting factor for time-varying linear systems. *Automatica*, 25(4):609–612, 1989.

[30] C. Cantelmo and L. Piroddi. Adaptive model selection for polynomial narx models. *IET control theory & applications*, 4(12):2693–2706, 2010.

[31] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in neural information processing systems*, pages 409–415, 2001.

[32] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[33] J. Chen and R. J. Patton. *Robust model-based fault diagnosis for dynamic systems*, volume 3. Springer Science & Business Media, 2012.

[34] C. K. Chui. *An introduction to wavelets*. Elsevier, 2016.

[35] A. Cotter, M. Gupta, and J. Pfeifer. A light touch for heavily constrained sgd. In *Conference on Learning Theory*, pages 729–771, 2016.

[36] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006.

[37] K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weight vectors. In *Advances in neural information processing systems*, pages 414–422, 2009.

[38] K. Crammer and D. D. Lee. Learning via gaussian herding. In *Advances in neural information processing systems*, pages 451–459, 2010.

[39] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 41(7):909–996, 1988.

[40] C. De Boor, C. De Boor, E.-U. Mathématicien, C. De Boor, and C. De Boor. *A practical guide to splines*, volume 27. Springer-Verlag New York, 1978.

[41] C. de Boor and R. DeVore. Approximation by smooth multivariate splines. *Transactions of the American Mathematical Society*, 276(2):775–788, 1983.

[42] T. G. Dietterich. Machine learning for sequential data: A review. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 15–30. Springer, 2002.

[43] T. G. Dietterich et al. Ensemble methods in machine learning. *Multiple classifier systems*, 1857:1–15, 2000.

[44] F. Ding and T. Chen. Performance bounds of forgetting factor least-squares algorithms for time-varying systems with finite measurement data. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 52(3):555–566, 2005.

[45] S. Ding, X. Xu, and R. Nie. Extreme learning machine and its applications. *Neural Computing and Applications*, 25(3-4):549–556, 2014.

[46] S. Ding, H. Zhao, Y. Zhang, X. Xu, and R. Nie. Extreme learning machine: algorithm, theory and applications. *Artificial Intelligence Review*, 44(1):103–115, 2015.

[47] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM, 2000.

[48] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

[49] K.-L. Du and M. Swamy. Radial basis function networks. In *Neural Networks and Statistical Learning*, pages 299–335. Springer, 2014.

[50] J. Duarte and J. Gama. Ensembles of adaptive model rules from high-speed data streams. In *Proceedings of the 3rd International Conference on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications-Volume 36*, pages 198–213. JMLR. org, 2014.

[51] J. Duarte, J. Gama, and A. Bifet. Adaptive model rules from high-speed data streams. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(3):30, 2016.

[52] E. Eleftheriou and D. Falconer. Tracking properties and steady-state performance of rls adaptive filter algorithms. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(5):1097–1110, 1986.

[53] R. Elwell and R. Polikar. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, 22(10):1517–1531, 2011.

[54] K. Eriksson, D. Estep, and C. Johnson. Piecewise linear approximation. In *Applied Mathematics: Body and Soul*, pages 741–753. Springer, 2004.

[55] E. Eweda and O. Macchi. Convergence of the rls and lms adaptive filters. *IEEE Transactions on Circuits and Systems*, 34(7):799–803, 1987.

[56] J. A. Farrell and M. M. Polycarpou. *Adaptive approximation based control: unifying neural, fuzzy and traditional adaptive approximation approaches*, volume 48. John Wiley & Sons, 2006.

[57] G. Feng, G.-B. Huang, Q. Lin, and R. Gay. Error minimized extreme learning machine with growth of hidden nodes and incremental learning. *IEEE Transactions on Neural Networks*, 20(8):1352–1357, 2009.

[58] P. M. Ferreira and A. E. Ruano. Online sliding-window methods for process model adaptation. *IEEE transactions on instrumentation and Measurement*, 58(9):3012–3020, 2009.

[59] M. Forray. Approximation theory and methods, 1981.

[60] T. Fortescue, L. S. Kershenbaum, and B. E. Ydstie. Implementation of self-tuning regulators with variable forgetting factors. *Automatica*, 17(6):831–835, 1981.

[61] G. Freud. *Orthogonal polynomials*. Elsevier, 2014.

[62] H. Freudenthal. Simplizialzerlegungen von beschrankter flachheit. *Annals of Mathematics*, pages 580–582, 1942.

[63] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[64] J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.

[65] J. Gama and M. M. Gaber. *Learning from data streams: processing techniques in sensor networks*. Springer, 2007.

[66] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Brazilian Symposium on Artificial Intelligence*, pages 286–295. Springer, 2004.

[67] J. Gama, P. Medas, and R. Rocha. Forest trees for on-line data. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 632–636. ACM, 2004.

[68] W. Gao, R. Jin, S. Zhu, and Z.-H. Zhou. One-pass auc optimization. In *International Conference on Machine Learning*, pages 906–914, 2013.

[69] J. Gertler. *Fault Detection and Diagnosis*. Springer, 2015.

[70] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[71] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. One-pass wavelet decompositions of data streams. *IEEE Transactions on knowledge and data engineering*, 15(3):541–554, 2003.

[72] G.-O. Glentis, K. Berberidis, and S. Theodoridis. Efficient least squares adaptive algorithms for fir transversal filtering. *IEEE signal processing magazine*, 16(4):13–41, 1999.

[73] G. Goodwin, E. Teoh, and H. Elliott. Deterministic convergence of a self-tuning regulator with covariance resetting. In *IEE Proceedings D-Control Theory and Applications*, volume 130, pages 6–8. IET, 1983.

[74] G. C. Goodwin and K. S. Sin. *Adaptive filtering prediction and control*. Courier Corporation, 2014.

[75] L. Guo, J.-H. Hao, and M. Liu. An incremental extreme learning machine for online sequential learning problems. *Neurocomputing*, 128:50–58, 2014.

[76] L. Guo, L. Ljung, and P. Priouret. Performance analysis of the forgetting factor rls algorithm. *International journal of adaptive control and signal processing*, 7(6):525–537, 1993.

[77] M. Gupta, A. Cotter, J. Pfeifer, K. Voevodski, K. Canini, A. Mangylov, W. Moczydlowski, and A. Van Esbroeck. Monotonic calibrated interpolated look-up tables. *The Journal of Machine Learning Research*, 17(1):3790–3836, 2016.

[78] F. Gustafsson and F. Gustafsson. *Adaptive filtering and change detection*, volume 1. Wiley New York, 2000.

[79] T. Hastie and R. Tibshirani. *Generalized additive models*. Wiley Online Library, 1990.

[80] S. S. Haykin, S. S. Haykin, S. S. Haykin, and S. S. Haykin. *Neural networks and learning machines*, volume 3. Pearson Upper Saddle River, NJ, USA:, 2009.

[81] H. Hellendoorn and D. Driankov. *Fuzzy model identification: selected approaches*. Springer Science & Business Media, 2012.

[82] J. L. Hennessy and D. A. Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011.

[83] T. M. Heskes and B. Kappen. On-line learning processes in artificial neural networks. *North-Holland Mathematical Library*, 51:199–233, 1993.

[84] R. J. Howlett and L. C. Jain. *Radial basis function networks 1: recent developments in theory and applications*, volume 66. Springer Science & Business Media, 2001.

[85] R. J. Howlett and L. C. Jain. *Radial basis function networks 2: new advances in design*, volume 67. Physica, 2013.

[86] G. Huang, G.-B. Huang, S. Song, and K. You. Trends in extreme learning machines: A review. *Neural Networks*, 61:32–48, 2015.

[87] G.-B. Huang and L. Chen. Convex incremental extreme learning machine. *Neurocomputing*, 70(16):3056–3062, 2007.

[88] G.-B. Huang, D. H. Wang, and Y. Lan. Extreme learning machines: a survey. *International journal of machine learning and cybernetics*, 2(2):107–122, 2011.

[89] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 985–990. IEEE, 2004.

[90] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.

[91] K. J. Hunt. A survey of recursive identification algorithms. *Transactions of the Institute of Measurement and Control*, 8(5):273–278, 1986.

[92] E. Ikonomovska, J. Gama, and S. Džeroski. Learning model trees from evolving data streams. *Data mining and knowledge discovery*, 23(1):128–168, 2011.

[93] E. Ikonomovska, J. Gama, and S. Džeroski. Online tree-based ensembles and option trees for regression on evolving data streams. *Neurocomputing*, 150:458–470, 2015.

[94] J.-S. Jang. Anfis: adaptive-network-based fuzzy inference system. *IEEE transactions on systems, man, and cybernetics*, 23(3):665–685, 1993.

[95] H. Jin, X. Chen, J. Yang, and L. Wu. Adaptive soft sensor modeling framework based on just-in-time learning and kernel partial least squares regression for nonlinear multiphase batch processes. *Computers & Chemical Engineering*, 71:77–93, 2014.

[96] V. John. Adaptive polynomial filters. *Circuits and Systems Tutorials*, page 59, 1996.

[97] A. Justel, D. Peña, and R. Zamar. A multivariate kolmogorov-smirnov test of goodness of fit. *Statistics & Probability Letters*, 35(3):251–259, 1997.

[98] N. Kasabov. Evolving fuzzy neural networks-algorithms, applications and biological motivation. *Methodologies for the conception, design and application of soft computing, World Scientific*, 1:271–274, 1998.

[99] N. K. Kasabov and Q. Song. Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE transactions on Fuzzy Systems*, 10(2):144–154, 2002.

[100] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 180–191. VLDB Endowment, 2004.

[101] O. Kisi. Wavelet regression model as an alternative to neural networks for river stage forecasting. *Water resources management*, 25(2):579–600, 2011.

[102] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281–300, 2004.

[103] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *ICML*, pages 487–494, 2000.

[104] R. Klinkenberg and I. Renz. Adaptive information filtering: Learning drifting concepts. In *Proc. of AAAI-98/ICML-98 workshop Learning for Text Categorization*, pages 33–40. Citeseer, 1998.

[105] J. Kohlmorgen and S. Lemm. An on-line method for segmentation and identification of non-stationary time series. In *Neural*

Networks for Signal Processing XI, 2001. Proceedings of the 2001 IEEE Signal Processing Society Workshop, pages 113–122. IEEE, 2001.

[106] G. Konidaris, S. Osentoski, and P. S. Thomas. Value function approximation in reinforcement learning using the fourier basis. In *AAAI*, volume 6, page 7, 2011.

[107] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak. Ensemble learning for data stream analysis: a survey. *Information Fusion*, 37:132–156, 2017.

[108] H. W. Kuhn. Some combinatorial lemmas in topology. *IBM Journal of research and development*, 4(5):518–524, 1960.

[109] L. I. Kuncheva. Change detection in streaming multivariate data using likelihood detectors. *IEEE Transactions on Knowledge and Data Engineering*, 25(5):1175–1180, 2013.

[110] Y. Lan, Y. C. Soh, and G.-B. Huang. Ensemble of online sequential extreme learning machine. *Neurocomputing*, 72(13):3391–3395, 2009.

[111] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[112] S.-H. Leung and C. So. Gradient-based variable forgetting factor rls algorithm in time-varying environments. *IEEE Transactions on Signal Processing*, 53(8):3141–3150, 2005.

[113] L. Li, M. L. Littman, and T. J. Walsh. Knows what it knows: a framework for self-aware learning. In *Proceedings of the 25th international conference on Machine learning*, pages 568–575. ACM, 2008.

[114] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on neural networks*, 17(6):1411–1423, 2006.

[115] A. Liaw, M. Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[116] J.-s. Lim, S. Lee, and H.-S. Pang. Low complexity adaptive forgetting factor for online sequential extreme learning machine (os-elm) for application to nonstationary system estimations. *Neural Computing and Applications*, 22(3-4):569–576, 2013.

[117] T. Lim and M. Macleod. On-line interpolation using spline functions. *IEEE Signal Processing Letters*, 3(5):144–146, 1996.

[118] J. Liu and E. Zio. An adaptive online learning approach for support vector regression: Online-svr-fid. *Mechanical Systems and Signal Processing*, 76:796–809, 2016.

[119] J. Liu and E. Zio. A svr-based ensemble approach for drifting data streams with recurring patterns. *Applied Soft Computing*, 47:553–564, 2016.

[120] L. Ljung and S. Gunnarsson. Adaptation and tracking in system identification—a survey. *Automatica*, 26(1):7–21, 1990.

[121] W.-Y. Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.

[122] E. Lughofer. *Evolving fuzzy systems-methodologies, advanced concepts and applications*, volume 53. Springer, 2011.

[123] E. Lughofer and P. Angelov. Handling drifts and shifts in on-line data streams with evolving fuzzy systems. *Applied Soft Computing*, 11(2):2057–2068, 2011.

[124] E. Lughofer, J.-L. Bouchot, and A. Shaker. On-line elimination of local redundancies in evolving fuzzy systems. *Evolving Systems*, 2(3):165–187, 2011.

[125] E. Lughofer, C. Cernuda, S. Kindermann, and M. Pratama. Generalized smart evolving fuzzy systems. *Evolving Systems*, 6(4):269–292, 2015.

[126] E. Lughofer, E. Hüllermeier, and E.-P. Klement. Improving the interpretability of data-driven evolving fuzzy systems. In *EUSFLAT Conf.*, pages 28–33, 2005.

[127] E. Lughofer and E.-P. Klement. Flexfis: A variant for incremental learning of takagi-sugeno fuzzy systems. In *Fuzzy Systems, 2005. FUZZ'05. The 14th IEEE International Conference on*, pages 915–920. IEEE, 2005.

[128] E. D. Lughofer. Flexfis: A robust incremental learning approach for evolving takagi–sugeno fuzzy models. *IEEE Transactions on fuzzy systems*, 16(6):1393–1410, 2008.

[129] J. Ma, J. Theiler, and S. Perkins. Accurate on-line support vector regression. *Neural computation*, 15(11):2683–2703, 2003.

[130] M. A. Maloof and R. S. Michalski. Learning evolving concepts using partial-memory approach. In *Notes of the 1995 AAAI Fall Symposium on Active Learning*, 1995.

[131] M. Markou and S. Singh. Novelty detection: a review—part 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.

[132] M. Markou and S. Singh. Novelty detection: a review—part 2:: neural network based approaches. *Signal processing*, 83(12):2499–2521, 2003.

[133] F. J. Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

[134] Y. Meyer. Orthonormal wavelets. In *Wavelets*, pages 21–37. Springer, 1990.

[135] T. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997.

[136] B. Mizrach. Multivariate nearest-neighbour forecasts of ems exchange rates. *Journal of Applied Econometrics*, 7(S1), 1992.

[137] D. W. Moore. Simplical mesh generation with applications. Technical report, Cornell University, 1992.

[138] E. Moroshko, N. Vaits, and K. Crammer. Second-order non-stationary online learning for regression. *Journal of Machine Learning Research*, 16:1481–1517, 2015.

[139] H. Mouss, D. Mouss, N. Mouss, and L. Sefouhi. Test of page-hinckley, an approach for fault detection in an agro-alimentary production system. In *Control Conference, 2004. 5th Asian*, volume 2, pages 815–818. IEEE, 2004.

[140] N. Müller and R. Isermann. On-line adaptation of grid-based look-up tables using a fast linear regression technique. 2004.

[141] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on neural networks*, 1(1):4–27, 1990.

[142] O. Nelles. Orthonormal basis functions for nonlinear system identification with local linear model trees (lolimot). *IFAC Proceedings Volumes*, 30(11):639–644, 1997.

[143] N. C. Oza. Online bagging and boosting. In *Systems, man and cybernetics, 2005 IEEE international conference on*, volume 3, pages 2340–2345. IEEE, 2005.

[144] N. C. Oza and S. Russell. Experimental comparisons of online and batch versions of bagging and boosting. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 359–364. ACM, 2001.

[145] N. C. Oza and S. Russell. *Online ensemble learning*. University of California, Berkeley, 2001.

[146] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[147] C. Paleologu, J. Benesty, and S. Ciochina. A robust variable forgetting factor recursive least-squares algorithm for system identification. *IEEE Signal Processing Letters*, 15:597–600, 2008.

[148] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.

[149] J. Park and I. W. Sandberg. Approximation and radial-basis-function networks. *Neural computation*, 5(2):305–316, 1993.

[150] J. Parkum, N. K. Poulsen, and J. Holst. Selective forgetting in adaptive procedures. *IFAC Proceedings Volumes*, 23(8):137–142, 1990.

[151] J. Parkum, N. K. Poulsen, and J. Holst. Recursive forgetting algorithms. *International Journal of Control*, 55(1):109–128, 1992.

[152] J. E. Parkum. Recursive identification of time-varying systems, 1992. Supervised by Niels K. Poulsen, and Henrik Spliid, DTU Informatics, http://www.imm.dtu.dk.

[153] L. Pietruczuk, L. Rutkowski, M. Jaworski, and P. Duda. A method for automatic adjustment of ensemble size in stream data mining. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 9–15. IEEE, 2016.

[154] M. J. D. Powell. *Approximation theory and methods*. Cambridge university press, 1981.

[155] M. Pratama, S. G. Anavatti, P. P. Angelov, and E. Lughofer. Panfis: A novel incremental learning machine. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):55–68, 2014.

[156] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[157] H.-J. Rong, G.-B. Huang, N. Sundararajan, and P. Saratchandran. Online sequential fuzzy extreme learning machine for function approximation and classification problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(4):1067–1072, 2009.

[158] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[159] D. Saad. *On-line learning in neural networks*, volume 17. Cambridge University Press, 2009.

[160] D. Saad and S. A. Solla. Exact solution for on-line learning in multilayer neural networks. *Physical Review Letters*, 74(21):4337, 1995.

[161] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1393–1400. IEEE, 2009.

[162] M. E. Salgado, G. C. Goodwin, and R. H. Middleton. Modified least squares algorithm incorporating exponential resetting and forgetting. *International Journal of Control*, 47(2):477–491, 1988.

[163] T. Sato and K. Takei. Real-time robust identification algorithm for structural systems with time-varying dynamic characteristics. In *Smart Structures and Materials' 97*, pages 393–404. International Society for Optics and Photonics, 1997.

[164] M. Scarpiniti, D. Comminiello, R. Parisi, and A. Uncini. Nonlinear spline adaptive filtering. *Signal Processing*, 93(4):772–783, 2013.

[165] S. Schaal and C. G. Atkeson. Constructive incremental learning from only local information. *Neural computation*, 10(8):2047–2084, 1998.

[166] T. Schaul, S. Zhang, and Y. LeCun. No more pesky learning rates. *ICML (3)*, 28:343–351, 2013.

[167] J. C. Schlimmer and R. H. Granger Jr. Beyond incremental processing: tracking concept drift. In *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*, pages 502–507. AAAI Press, 1986.

[168] L. Schumaker. *Spline functions: basic theory*. Cambridge University Press, 2007.

[169] M. R. Segal. Machine learning benchmarks and random forest regression. *Center for Bioinformatics & Molecular Biostatistics*, 2004.

[170] A. Sen and M. S. Srivastava. On tests for detecting change in mean. *The Annals of statistics*, pages 98–108, 1975.

[171] W. A. Sethares. The least mean square family. *Adaptive system identification and signal processing algorithms*, pages 84–122, 1993.

[172] A. Shaker and E. Hüllermeier. Iblstreams: a system for instance-based classification and regression on data streams. *Evolving Systems*, 3(4):235–249, 2012.

[173] O. Sigaud, C. Salaün, and V. Padois. On-line regression algorithms for learning mechanical models of robots: a survey. *Robotics and Autonomous Systems*, 59(12):1115–1129, 2011.

[174] V. Šmídl and F. Gustafsson. Bayesian estimation of forgetting factor in adaptive filtering and change detection. In *Statistical Signal Processing Workshop (SSP), 2012 IEEE*, pages 197–200. IEEE, 2012.

[175] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[176] S. G. Soares and R. Araújo. A dynamic and on-line ensemble regression for changing environments. *Expert Systems with Applications*, 42(6):2935–2948, 2015.

[177] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.

[178] G. Szeg. *Orthogonal polynomials*, volume 23. American Mathematical Soc., 1939.

[179] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE transactions on systems, man, and cybernetics*, (1):116–132, 1985.

[180] P. T. P. Tang. Table-lookup algorithms for elementary functions and their error analysis. In *Computer Arithmetic, 1991. Proceedings., 10th IEEE Symposium on*, pages 232–236. IEEE, 1991.

[181] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[182] A. N. Tikhonov. On the stability of the functional optimization problem. *USSR Computational Mathematics and Mathematical Physics*, 6(4):28–33, 1966.

[183] B. Toplis and S. Pasupathy. Tracking improvements in fast rls algorithms using a variable forgetting factor. *IEEE Transactions on acoustics, speech, and signal processing*, 36(2):206–227, 1988.

[184] A. Tsymbal. The problem of concept drift: definitions and related work. 2004.

[185] K. Uosaki and M. Yotsuya. Adaptive identification with multiple forgetting factors. *Transactions of the Society of Instrument and Control Engineers*, 28(9):1046–1051, 1992.

[186] N. Vaits and K. Crammer. Re-adapting the regularization of weights for non-stationary regression. In *International Conference on Algorithmic Learning Theory*, pages 114–128. Springer, 2011.

[187] S. Van Vaerenbergh, J. Via, and I. Santamaría. A sliding-window kernel rls algorithm and its application to nonlinear channel identification. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 5, pages V–V. IEEE, 2006.

[188] V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[189] S. Vijayakumar, A. D'Souza, and S. Schaal. Lwpr: A scalable method for incremental online learning in high dimensions. 2005.

[190] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[191] M. VOGT, N. MÜLLER, and R. ISERMANN. On-line adaptation of grid-based look-up tables using a fast linear regression technique. *Journal of dynamic systems, measurement, and control*, 126(4):732–739, 2004.

[192] L. Wang and R. Langari. A variable forgetting factor rls algorithm with application to fuzzy time-varying systems identification. *International journal of systems science*, 27(2):205–214, 1996.

[193] L.-X. Wang and J. M. Mendel. Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. *IEEE transactions on Neural Networks*, 3(5):807–814, 1992.

[194] X. Wang and V. Makis. Autoregressive model-based gear shaft fault diagnosis using the kolmogorov–smirnov test. *Journal of Sound and Vibration*, 327(3):413–423, 2009.

[195] M. Welling. Support vector regression. *Department of Computer Science, University of Toronto, Toronto (Kanada)*, 2004.

[196] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.

[197] B. Widrow, M. E. Hoff, et al. Adaptive switching circuits. In *IRE WESCON convention record*, volume 4, pages 96–104. New York, 1960.

[198] N. Wiener. *Extrapolation, interpolation, and smoothing of stationary time series*, volume 7. MIT press Cambridge, MA, 1949.

[199] F. Xhafa, L. Barolli, A. Barolli, and P. Papajorgji. Modeling and processing for next-generation big-data technologies. *Cham: Springer International Publishing*, 2015.

[200] S. Yakowitz. Nearest-neighbour methods for time series analysis. *Journal of time series analysis*, 8(2):235–247, 1987.

[201] L. Yingwei, N. Sundararajan, and P. Saratchandran. A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural computation*, 9(2):461–478, 1997.

[202] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[203] S. Yun and K.-C. Toh. A coordinate gradient descent method for l 1-regularized convex minimization. *Computational Optimization and Applications*, 48(2):273–307, 2011.

[204] W. Yung and K. Man. Optimal selected forgetting factor for rls estimation. *IFAC Proceedings Volumes*, 26(2):331–334, 1993.

[205] Y. Zhou, S. Chan, and K. Ho. A new variable forgetting factor qr-based recursive least m-estimate algorithm for robust adaptive filtering in impulsive noise environment. In *Signal Processing Conference, 2006 14th European*, pages 1–5. IEEE, 2006.