RELIABLE ON-LINE MACHINE LEARNING FOR REGRESSION TASKS IN PRESENCE OF UNCERTAINTIES

ANDREAS BUSCHERMÖHLE

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
June 26, 2014

SUPERVISOR:
Prof. Dr.-Ing. Werner Brockmann

DISPUTATION:
September 29, 2014

LOCATION:
Osnabrück

ABSTRACT

Machine learning plays an increasingly important role in modern systems. The ability to learn from data enhances or enables many applications. Recently, quick in-stream processing of possibly a huge or even infinite amount of data gains more attention. This thesis deals with such on-line learning systems for regression that learn with every example incrementally and are reliable even in presence of uncertainties.

A new learning approach, called IRMA, is introduced which directly incorporates knowledge about the model structure into its parameter update. This way it is aggressive to incorporate a new example locally as much as possible and at the same time passive in the sense that the global output is changed as little as possible. It can be applied to any model structure that is linear in its parameters and is proven to minimize the worst case prediction error in each step. Hence, IRMA is reliable in every situation and the investigations show that in every case a bad performance is prevented by inherently averting overfitting even for complex model structures and in high dimensions.

An extension of such on-line learning systems monitors the learning process, regarding conflict and ignorance, and estimates the trustworthiness of the learned hypothesis by the means of trust management. This provides insight into the learning system at every step and the designer can adjust its setup if necessary. Additionally, the trust estimation allows to assign a trustworthiness to each individual prediction the learning system makes. This way the overall system can react to uncertain predictions at a higher level and increase its safety, e. g. by reverting to a fallback.

Furthermore, the uncertainties are explicitly incorporated into the learning process. The uncertainty of the hypothesis is reflected by allowing less change for more certain regions of the learned system. This way, good learned knowledge is protected and a higher robustness to disturbances is achieved. The uncertainty of each example used for learning is reflected by adapting less to uncertain examples. Thereby, the learning system gets more robust to training examples that are known to be uncertain.

All approaches are formally analyzed and their characteristic properties are demonstrated in empirical investigations. In addition, a real world application to forecasting electricity loads shows the benefits of the approaches.

ZUSAMMENFASSUNG

Maschinelles Lernen wird in modernen Systemen immer wichtiger. Die Möglichkeit, aus Beispieldaten zu lernen, verbessert viele Anwendungen oder ermöglicht diese überhaupt erst. Aktuell gewinnen Systeme an Bedeutung, die mit einem Strom (unendlich) vieler Daten umgehen können. Diese Arbeit befasst sich mit solchen online lernfähigen Systemen zur Regression, die mit jedem einzelnen Beispiel inkrementell lernen und dabei mit Ungewissheiten umgehen.

Ein neues Lernverfahren namens IRMA wird vorgestellt, welches bei der Aktualisierung der Parameter Wissen über die Modellstruktur nutzt. IRMA ist einerseits aggressiv, um das vorgegebene Beispiel möglichst gut zu berücksichtigen, und andererseits passiv, um die globale Veränderung der Ausgabe möglichst klein zu halten. Der Ansatz kann auf jegliche Modellstruktur angewandt werden, die linear in den Parametern ist. Es wird nachgewiesen, dass durch IRMA der schlimmstmögliche Vorhersagefehler in jedem Schritt minimiert wird. Damit ist IRMA in jeder Situation verlässlich und die Untersuchungen zeigen, dass schlechte Ergebnisse stets verhindert werden, da auch für komplexe Modellstrukturen und in hohen Dimensionen Überanpassung inhärent unterbunden wird.

Des Weiteren wird das online lernfähige System erweitert, um den Lernprozess in Bezug auf Konflikt und Unwissen zu überwachen. Dazu wird die Vertrauenswürdigkeit der erlernten Hypothese im Sinne des Trust Management eingeschätzt. Auf diese Weise ist ein tieferes Verständnis des lernfähigen Systems möglich und der Entwickler kann, wenn nötig, Einstellungen daran anpassen. Die Einschätzung der Vertrauenswürdigkeit ermöglicht es, außerdem auch jeder einzelnen Vorhersage des lernfähigen Systems eine Vertrauenswürdigkeit beizumessen. Damit kann dann das Gesamtsystem auf ungewisse Vorhersagen auf höherer Entscheidungsebene reagieren und somit die Sicherheit erhöhen.

Darüber hinaus werden die Ungewissheiten explizit beim Lernen berücksichtigt. Die Ungewissheit der Hypothese beeinflusst das Lernen so, dass sicheres Wissen dabei weniger angepasst wird. Dadurch wird gut erlerntes Wissen geschützt und das Lernen ist robuster gegen Störungen. Die Ungewissheit jedes einzelnen Beispiels wird dadurch berücksichtigt, dass weniger zuverlässige Beispiele einen geringeren Einfluss haben. Somit wird das lernfähige System robuster gegen Beispiele, von denen bekannt ist, dass sie störungsbehaftet sind.

Alle Ansätze werden formal analysiert und ihre charakteristischen Eigenschaften werden in empirischen Untersuchungen demonstriert. Zudem zeigt eine reale Anwendung zur Vorhersage von Stromverbräuchen die Vorteile der Ansätze.

PUBLICATIONS

Some ideas and figures of this work have been published previously in the following publications:

[1] W. Brockmann, A. Buschermoehle, and J. Huelsmann. A generic concept to increase the robustness of embedded systems by trust management. In *Proc. Int. Conf on Systems Man and Cybernetics*, pages 2037–2044. IEEE Press, 2010.

[2] W. Brockmann, A. Buschermoehle, J. Huelsmann, and N. Rosemann. *Trust Management – Handling Uncertainties in Embedded Systems*, pages 589–591. Autonomic Systems. Springer, 2011.

[3] W. Brockmann, A. Buschermoehle, and J. Schoenke. COBRA - a generic architecture for robust treatment of uncertain information. In *INFORMATIK 2013: Informatik angepasst an Mensch, Organisation und Umwelt*, pages 2727–2741. GI, Bonner Köllen Verlag, 2013.

[4] A. Buschermoehle and W. Brockmann. Stable on-line learning with optimized local learning, but minimal change of the global output. In *Proc. Int. Conf. on Machine Learning and Applications*, pages 21–27. IEEE Press, 2013.

[5] A. Buschermoehle and W. Brockmann. On-line learning with minimized change of the global mapping – optimized local learning by incremental risk minimization. *Evolving Systems*, 2014. (in press).

[6] A. Buschermoehle and W. Brockmann. Reliable localized on-line learning in non-stationary environments. In *Proc. Int. Conf. on Evolving and Adaptive Intelligent Systems*, pages 1–7. IEEE Press, 2014.

[7] A. Buschermoehle, J. Huelsmann, and W. Brockmann. A structured view on sources of uncertainty in supervised learning. In *Proc. Int. Conf. on Scalable Uncertainty Management*, volume 7520, pages 566–573. Springer, 2012.

[8] A. Buschermoehle, J. Huelsmann, and W. Brockmann. UOSLib – a library for analysis of online-learning algorithms. In *Proc. Workshop on Computational Intelligence*, pages 355–369. KIT Scientific Publishing, 2013.

[9] A. Buschermoehle, N. Rosemann, and W. Brockmann. Stable classification in enviroments with varying degrees of uncertainty.

In *Proc. Int. Conf on Computational Intelligence for Modelling, Control and Automation*, pages 447–452. IEEE press, 2008.

[10] A. Buschermoehle, J. Schoenke, and W. Brockmann. Trusted learner: An improved algorithm for trusted incremental function approximation. In *Proc. Int. Symp. on Computational Intelligence in Dynamic and Uncertain Environments*, pages 16–24. IEEE Press, 2011.

[11] A. Buschermoehle, J. Schoenke, and W. Brockmann. Uncertainty and trust estimation in incrementally learning function approximation. In *Proc. Int. Conf on Information Processing and Management of Uncertainty*, pages 32–41. Springer, 2012.

[12] A. Buschermoehle, J. Schoenke, N. Rosemann, and W. Brockmann. The incremental risk functional: Basics of a novel incremental learning approach. In *Proc. Int. Conf. on Systems Man and Cybernetics*, pages 1500–1505. IEEE Press, 2013.

[13] J. Huelsmann, A. Buschermoehle, and W. Brockmann. Incorporating dynamic uncertainties into a fuzzy classifier. In *Proc. Int. Conf. of the European Society for Fuzzy Logic and Technology*, pages 388–395. EUSFLAT, Atlantis Press, 2011.

[14] N. Rosemann, A. Buschermoehle, and W. Brockmann. Beschleunigung der Selbstoptimierung durch Selbstsimulation. In *Proc. Workshop on Computational Intelligence*, pages 114–128. KIT Scientific Publishing, 2009.

In times of profound change, the learners inherit the earth,
while the learned find themselves beautifully equipped
to deal with a world that no longer exists.

— Eric Hoffer

When you know a thing, to hold that you know it;
and when you do not know a thing, to allow that you do not know it
- this is knowledge.

— Confucius

ACKNOWLEDGMENTS

It would not have been possible to write this doctoral thesis without the help and support of the kind people around me. I would like to thank all those people who made this thesis possible and an indispensable experience for me.

This thesis would not have been possible without the help, support and patience of my supervisor Prof. Dr.-Ing. Werner Brockmann. I thank him for the guidance and his practical view on the requirements of machine learning systems. Likewise, many fruitful discussions within the *Smart Embedded Systems Group* helped in gaining new ideas or deeper insight into problems and approaches. Especially, I want to thank my colleagues Jens Hülsmann and Nils Rosemann who have been with me a long time on this road and with whom I have worked on investigations and written papers. Special thanks are directed to Jan Schoenke who worked with me on many aspects of my thesis as a student assistant. Just as well I want to thank Christian Lintze and Jonas Schneider for the many discussions.

Last, but by no means least, I thank my parents Agnes and Franz Buschermöhle for their early on support in my scientific curiosity and my education, as well as my brothers Michael and Frank. I would particularly like to thank my girlfriend Alexandra Perk for her continued support, motivation, and sympathy during my research for this thesis.

Osnabrück, June 26, 2014 Andreas Buschermöhle

CONTENTS

LIST OF TABLES

LISTINGS

ACRONYMS

AROW	adaptive regularization of weights
COBRA	confidence optimization-braced real-time architecture
CW	confidence weighted learning
GH	Gaussian herding
GLT	grid-based lookup table
IRMA	incremental risk minimization algorithm
KWIK	knows what it knows
LIP	linear in parameters
ORCA	organic robotic control architecture
PA	passive-aggressive algorithm

RLS	recursive least squares
ROMMA	relaxed online maximum margin algorithm
SIRMA	second order incremental risk minimization algorithm
SOP	second order perceptron
UOSLib	unified online-learning systems library

LIST OF SYMBOLS

Scalars are denoted with lower case letters, e. g. x, and vectors with bold face letters, e. g. $\mathbf{x}$. The transpose of a vector $\mathbf{x}$ is denoted by $\mathbf{x}^T$. The ith element of a vector $\mathbf{x}$ is denoted by x_i. Since on-line learning deals with a sequence of steps, $\mathbf{x}_t$ is the tth vector in a sequence of vectors $\mathbf{x}_0, \mathbf{x}_1, \ldots$. The ith element of $\mathbf{x}_t$ is denoted by $x_{t,i}$. The most important and globally used symbols are listed in the following:

$\tilde{f}$	Model
$\mathbf{x}$	Instance (input vector)
X	Input space
d	Input dimensionality, i. e. number of independent variables
y	Label (output)
Y	Output space
F	Feature space
n	Feature dimensionality, i. e. number of features and parameters
ω	Parameter vector
Ω	Parameter space
ϕ	Non-linear transformation / model structure
$\hat{\phi}$	Activity vector
n_d	Number of training examples
n_g	Number of ground truth examples
L	Loss function
D	Distance function
C	Complexity function
L_c	Cumulative loss
L_d	Data loss
L_g	Ground truth loss
C_a	Aggressiveness parameter of PA
R_{inc}	Incremental risk functional

σ	Stiffness
τ	Growth of SIRMA stiffness
$\hat{\sigma}$	Maximal stiffness of sigmoidal SIRMA
λ	Forgetting factor
Σ	Covariance matrix of second order updates
ϑ	Trust level
Φ	Ignorance measure of the parameter vector
$\bar{\Delta}$	Conflict measure of the parameter vector
δ_t	Tolerance of linear trust mapping
δ_s	Sensitivity of linear trust mapping
ϑ^{I_l}	Linear ignorance trust
ϑ^{I_h}	Hyperbolic ignorance trust
η_I	Hyper-parameter of hyperbolic ignorance trust
ϑ^{C_l}	Linear conflict trust
ϑ^{C_h}	Hyperbolic conflict trust
η_C	Hyper-parameter of hyperbolic conflict trust
ϑ^{D_l}	Linear direct incremental trust
$\hat{\Delta}$	Normalized parameter adjustment
ϑ^{D_h}	Hyperbolic direct incremental trust
η_A	Hyper-parameter of hyperbolic direct incremental trust mapping
$\vartheta^{\star}$	Combined parameter trust
$\vartheta_{\widehat{y}_t}$	Prediction trust
e_w	Trust weighted mean squared error
$\overline{\vartheta}$	Mean trust
ζ	Example density of a parameter
ψ	Ignorance to stiffness mapping
ϑ_{x_t,y_t}	Fused trust of an example
ρ	True example density distribution

1 INTRODUCTION

1.1 MOTIVATION

The ever increasing complexity of systems results in an engineering bottleneck for the developer. Thus the need for systems that can *learn* to improve their functional behavior arises. Consider, for example, the influence of weather conditions on the consumption of electricity in the power grid. Acquiring data of this influence is relatively easy, but its analysis by experts to build a system model is highly tedious. Yet, an expert is able to derive a generalized concept from the acquired data and use this concept as a model in new situations. Therefore, an automation of this process of building a model from data could reduce the manual effort severely. This way, one single learning program could learn to handle a variety of tasks.

Machine learning deals with such systems that can learn from *training examples* based on *induction* and then be used to *predict* the outcome on evaluation data for new unlabeled instances (see Fig. 1). Its two main principles are representation and generalization. On the one hand, the training examples need to be represented properly by some kind of model structure, i. e. in general one hypothesis chosen from a *set of hypotheses*. On the other hand, the model needs to generalize well to unseen inputs. The idea dates back to the first learning system, called the *Perceptron*, introduced by Rosenblatt in 1958 [95] followed by the first work on *computational learning theory* by Novikoff in 1962 [90]. Since then, many different branches of machine learning have developed.

One branch, called *regression*, deals with learning a continuous relationship between a dependent variable and one or more independent variables from training examples. Particularly this means, the output value, i. e. the dependent variable, has an ordinal relation, e. g. 1 Wh

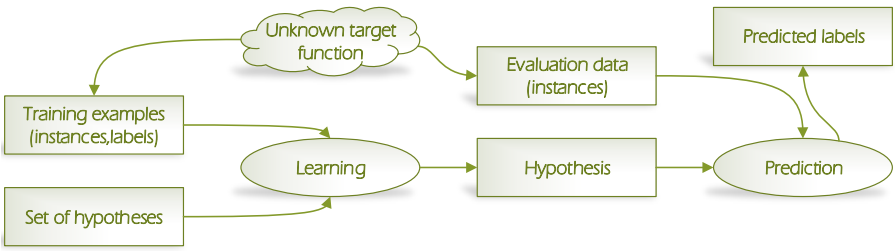


Figure 1: Flowchart of a general machine learning setting.

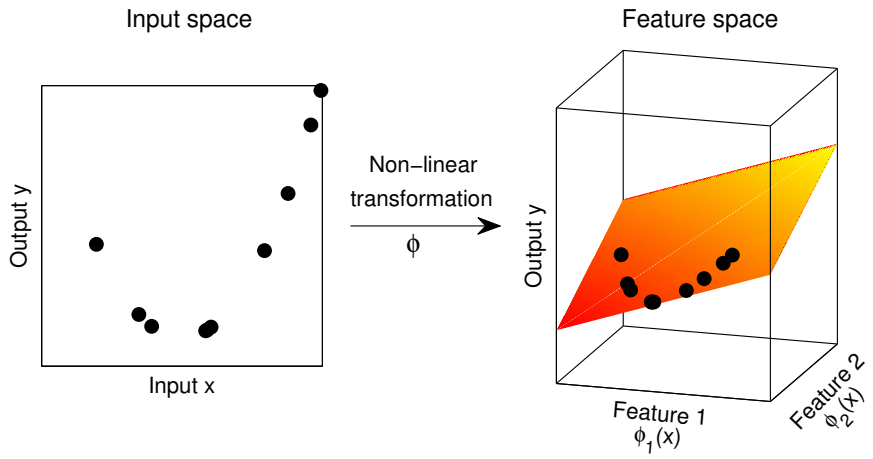


Figure 2: For LIP models, the model structure transforms data from input space to feature space where a hyperplane is used to approximate the training examples.

of consumed electricity is more than 0.8 Wh of consumed electricity. Generally, a model $\tilde{f}$ can be described by

$$y = \tilde{f}(x, \omega_{(l)}, \omega_{(n)}) = {\omega_{(l)}}^{T} \cdot \psi(x, \omega_{(n)}) \tag{1}$$

with the dependent variable $y \in Y \subseteq \mathbb{R}$, the vector of independent variables $x \in X \subseteq \mathbb{R}^d$ and two parameter vectors $\omega_{(l)}$ and $\omega_{(n)}$. The latter has a non-linear influence on the relation $X \to Y$, as it parameterizes a non-linear mapping ψ, whereas the first has a linear influence on the relation $X \to Y$ (see Section 1.4.2 of [48] for more details). As shown by [17], a model that is non-linear in its parameters can achieve squared errors of order $\mathcal{O}(\frac{1}{d})$, while a model that is linear in parameters (LIP) is bounded by $\mathcal{O}(\frac{1}{n^{2/d}})$ for n parameters and d independent variables. So in principle, a lower error is possible for $d > 2$ using non-linear parameters. But, learning these parameters $\omega_{(n)}$ is a much more complex task due to local minima of the error functional to be optimized than learning the linear parameters $\omega_{(l)}$. Additionally, the influence of non-linear parameters is hard to understand for an expert using such a learning system.

Thus, this work focuses on the class of LIP models, also known as *generalized linear models* [22, 48, 88], henceforth written as

$$y = \omega^{T} \cdot \phi(x) \tag{2}$$

with the parameter vector ω representing the hypothesis about the correct model and a non-linear transformation of the input vector x through a set of basis functions $\{\phi_i(x)\}_{i=1}^{n}$.

As shown in Fig. 2 a non-linear relationship between the dependent variable (output) and the independent variables (inputs) is transformed in such a way that it can be approximated linearly in feature space. The choice of the basis functions determines the model

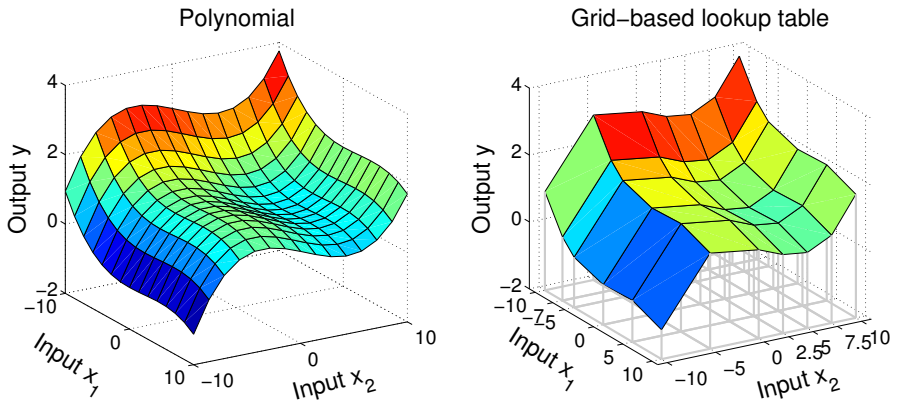


Figure 3: Two examples for LIP model structures, i. e. a fifth order polynomial (left) and a GLT with 6×7 nodes (right), approximating the same input-output relation.

structure used for approximation and consequently its expressiveness. Two LIP model structures, i. e. a polynomial and a grid-based lookup table (GLT) with linear interpolation, are shown as examples in Fig. 3. It is important to note that the resulting continuous relationship between the dependent variable and the independent variables may just as well be non-linear, depending on the model structure.

Within regression, an important machine learning approach is *online learning* [32]. While classical *batch learning* is based on training a model on some batch of data, and evaluating it on other data, on-line learning continuously learns on a stream of data, one example at a time, and has to make predictions continuously as well.

This approach has several benefits. The training examples are not collected but only incorporated each on its own. This way, the memory demand as well as the computational effort are not only low, but also fixed at design time. Thus, an on-line learning algorithm is suited for *real-time* applications and *embedded systems* with low computational power. Because of this simplicity, on-line learning algorithms are typically much more easy to understand and an expert can set up the algorithm efficiently and plan ahead its behavior beforehand. Its computational simplicity further allows to apply on-line learning algorithms in the realm of *big data* where a batch approach is not feasible even with modern high performance machines. Besides these advantages in complexity, the incorporation of new examples as they arrive makes an adaptation to changing conditions, i. e. in *non-stationary* environments, possible. Consequently, it enables adaptiveness to time variant system properties and life long learning.

This in-stream learning is natural on certain tasks like the power grid example. New measurements of the electricity consumption are available all the time and a model can be updated, to adapt to varying weather conditions, new consumers, new generators, or otherwise altered generation and consumption profiles. Similarly in time series

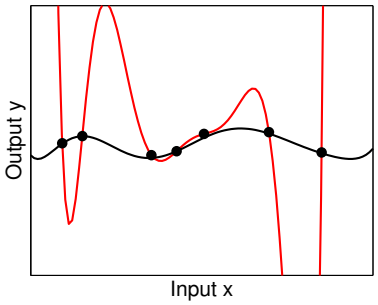


Figure 4: Overfitting on sparse data (black dots) may lead to random effects on generalization (red line) whereas a smooth approximation is less sensitive (black line).

prediction or adaptive control, new examples are always available as a stream of data.

But for successful on-line learning several challenges have to be met as well. As only a single example is used to revise a hypothesis, it gives only local information in two ways. On the one hand, the example is local in the input space, i. e. it defines one point of the input-output relation to be learned. On the other hand, it is local in time, i. e. it defines the input-output relation at one point in the data stream. Most on-line learning approaches directly regard locality in time, as the example is incorporated into the model at the time of occurrence and will be forgotten if a new example arrives at the same or a similar point in input space. Yet, with this example the global relation of the model is adapted and locality in input space hence needs to be assured by the learning algorithm.

Due to the local information of a single example, minimizing the model error on this example alone is an ill-posed problem. So further constraints need to be set to define how to incorporate a new example. Typically these constraints somehow require the current hypothesis about the model to be changed only as little as possible. This way, the additional constraints determine the on-line learning algorithm. Different approaches will be discussed in Chapter 2.1 in more detail.

At the beginning of learning, the examples are sparse and the hypothesis is not yet converged suitably. Still the learning algorithm has to adapt quickly to the sparse data without overfitting [46] and perform well right from the start. Especially in high dimensional input spaces, this initialization with sparse data may occur at any time during learning as shifts or drifts in the underlying process occur. Figure 4 illustrates how overfitting can lead to random predictions due to bad generalization capabilities.

Furthermore, the examples within the stream often strongly depend on each other. Data of a dynamic system or from a time series typically comes along a continuous trajectory. Therefore, it cannot be expected to get examples randomly and equally distributed across the input space.

Despite all these challenges, the on-line learning system has to provide a reliable prediction *at any time for any input*. The prediction usually influences a system directly or indirectly. For example a time series prediction is used to make decisions [14, 34], a learned model of a system might be used for model predictive control [30], or the learning system itself is an adaptive controller [48, 101]. So the prediction is critical for the system's safety, and erroneous predictions have to be prevented. Consequently, any uncertainties influencing the learning system are dangerous and a well determined reliable prediction as well as robust learning must be assured. Reliability of a learning system, defined as *"the ability to perform certain tasks conforming to required quality standards"* [40], in this context comprises the following qualities:

- RECALL If an example $(\mathbf{x}_t, y_t)$ was presented for learning, a label similar to y_t is expected close to $\mathbf{x}_t$ for prediction later on. Even if other examples are presented this should hold until a new example is presented close to $\mathbf{x}_t$. So there should be no fatal forgetting.
- COMPLIANT GENERALIZATION If in some region no examples were presented, the prediction should lie in between neighboring examples. So overfitting causing false generalization must be prevented.

At the same time, robustness of a learning system to malicious influences, defined as *"that the distribution of the estimator changes only slightly if the distribution of the observations is slightly altered"* [56], in this context comprises:

- If an information (e. g. training example or parameter vector) is fluctuating or uncertain, the influence on the result should be low, i. e. a low variance at the outcome.
- If the presented examples are noisy, an averaging output should be given.
- If an uncertain example is presented, it should have low influence on the adaptation of the model.

1.2 ON-LINE LEARNING SETTING

The on-line learning setting is characterized by learning on a sequence of data which can be described in steps (see Fig. 5). In each step t the learning algorithm is presented an *instance* $\mathbf{x}_t \in X \subseteq \mathbb{R}^d$ which is transferred from input space X to feature space F by the model structure through a vector of *basis functions* $\boldsymbol{\phi}(\mathbf{x}_t) : X \to F \subseteq \mathbb{R}^n$. This input is then used to predict its *label* $\hat{y}_t \in Y \subseteq \mathbb{R}$ through a LIP

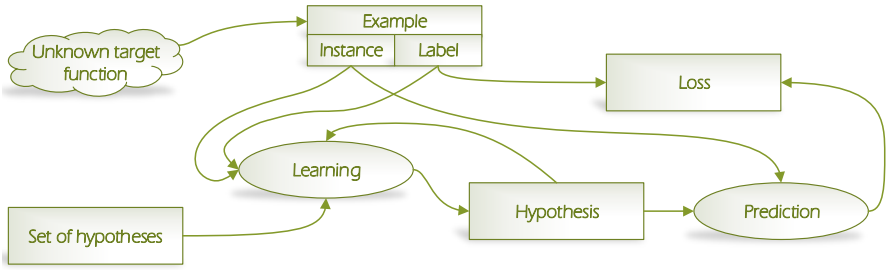


Figure 5: Flowchart of the on-line machine learning setting.

model $\hat{y}_t = \boldsymbol{\omega}_t^T \boldsymbol{\phi}(\mathbf{x}_t)$ with the *parameter vector* $\boldsymbol{\omega}_t \in \Omega \subseteq \mathbb{R}^n$, which is the hypothesis maintained by the learning algorithm. Afterwards, the correct label y_t is given and the learning algorithm suffers an *instantaneous loss* $L(y_t, \hat{y}_t) \geqslant 0$ reflecting how wrong the prediction was. With the new pair of an instance and its corresponding label, henceforth called an *example* $(\mathbf{x}_t, y_t)$, the learning algorithm updates its hypothesis to $\boldsymbol{\omega}_{t+1}$ with the aim to minimize the *cumulative loss*

$$L_c = \sum_{t=0}^{n_d-1} L(y_t, \hat{y}_t) \tag{3}$$

where n_d is the length of the data sequence provided so far.

In a static case, there is one target vector $\boldsymbol{\omega}^* \in \Omega$ that minimizes the cumulative loss. Whereas in presence of time variance, the target vector is subject to shifts or drifts and thus a time dependent sequence of target vectors $\boldsymbol{\omega}_t^* \in \Omega,\ t = 0, \ldots, n_d - 1$ minimizes the cumulative loss.

Variations of this basic on-line learning setting include a time delay between the presentation of an instance for prediction and the presentation of the true label, i. e. the prediction for instance $\mathbf{x}_t$ is done with a parameter vector $\boldsymbol{\omega}_j$ with $j < t$. Or they require more than one prediction between the presentation of two examples, i. e. not every instance is presented with a label. For example in the power grid scenario, a prediction of the power consumption for every quarter of the next 24 hours would require a total of $96 = 24 \cdot 4$ predictions every 15 minutes. But only one new example would be presented in that time. Hence, not only the cumulative loss of (3) is important to be small but the global model error to the unknown true target function should also be small to ensure reliable predictions for multiple random instances to be labeled. Especially this global error is severely affected by overfitting.

Depending on the on-line learning task, the loss function and prediction model differ. For the task of regression, a common choice for the loss function is the squared error

$$L_r(y_t, \hat{y}_t) = (y_t - \hat{y}_t)^2. \tag{4}$$

For a LIP model, the squared error loss of one example thus defines

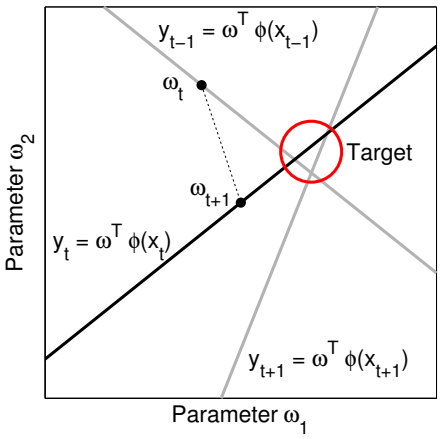


Figure 6: On-line learning depicted in a two dimensional parameter space. Three subsequent training examples each define a hyperplane of possible solutions (lines) and the parameter vector is adapted accordingly. The intersection area of the examples contains the target parameter vector.

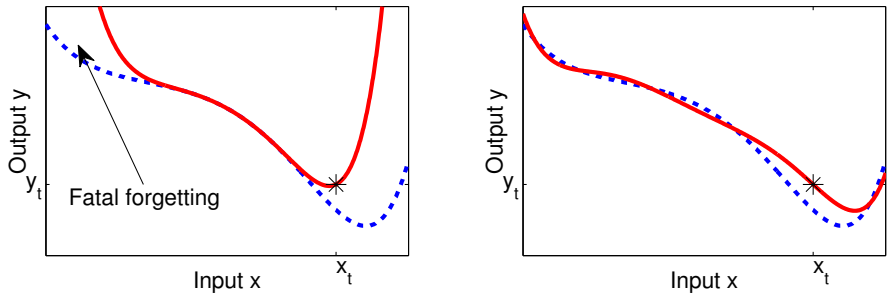


Figure 7: On-line learning of a hypothesis (blue dashed) depicted in input space. The new hypothesis (red solid) is chosen to meet the example (asterisk) with fatal forgetting (left) in contrast to securing the global knowledge (right).

a hyperplane

$$y_t = \boldsymbol{\omega}^T \cdot \boldsymbol{\phi}(\mathbf{x}_t) \; . \tag{5}$$

of possible solutions $\boldsymbol{\omega}$ in parameter space Ω which all satisfy a zero loss on the example. Accordingly, a typical scenario within a sequence of examples is depicted in Fig. 6 for a two-parameter case within the parameter space. After the current hypothesis $\boldsymbol{\omega}_t$ was selected based on the last example at $t-1$, a new example is presented (black line) and a new hypothesis $\boldsymbol{\omega}_{t+1}$ chosen without knowledge about any other examples (gray lines). Together all examples specify some target region due to noise or a restricted set of hypotheses. In the ideal case of no noise and an exactly representable target function, all examples would intersect at one point in parameter space, i. e. the optimal hypothesis.

In input space the scenario looks quite different. The example contains only local information about the target input-output relation.

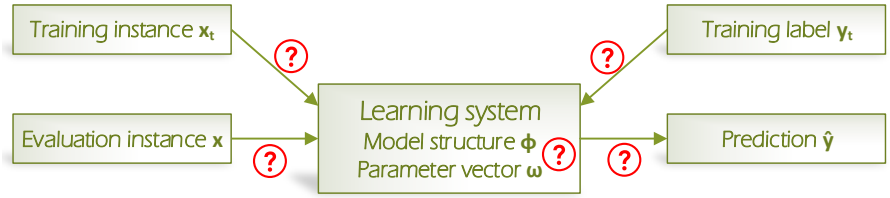


Figure 8: Overview of a learning system and its influences which introduce uncertainties.

Hence, when adapting the global model based on this local example, fatal forgetting can occur, but should be prevented. Two different updates of the parameter vector are shown as an example in Fig. 7. In both cases the example is learned perfectly, but the influence on the global model differs significantly. While the left plot shows a high global change of the model especially far away from the local example, the global model is kept as far as possible in the right plot and fatal forgetting is prevented when updating the parameter vector.

1.3 UNCERTAINTIES IN LEARNING SYSTEMS

The hypothesis of a learning system about the correct model is always afflicted with uncertainty. Indeed, it is typical that more than one hypothesis may explain the presented examples, which means that the validity of a chosen hypothesis and its predictions can usually not be sure. Apart from this, a learning system is subject to additional sources of uncertainty as well, both at training and prediction, which altogether may result in a possibly wrong prediction and thus increase the risk of an unsafe system behavior.

There are two inherently different kinds of uncertainty, called *aleatoric* and *epistemic* [58, 99]. Aleatoric uncertainty refers to variability, e. g. from noise or unobserved influences. In contrast, epistemic uncertainty refers to missing knowledge, e. g. due to faults or a lack of training examples. So epistemic uncertainties can be reduced through learning by acquiring more examples, whereas aleatoric uncertainties are irreducible.

An abstract view regarding uncertainties in clustering and classification is presented in [57]. In this thesis, the sources of uncertainty are categorized according to [7] regarding their possible consequences for on-line learning (see Fig. 8). On the one hand the training examples and the instances for prediction are subject to uncertainties. And on the other hand the learning algorithm and the learned model introduce uncertainties as well. All these sources of uncertainty accumulate at a given prediction. From this perspective, the sources of uncertainty can be structured into the following five categories.

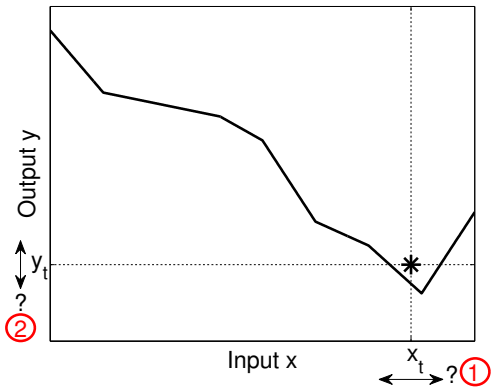


Figure 9: Uncertainty of the training example, e. g. due to noise or failures. The model is shown as a black curve and the uncertain example as an asterisk. The numbering refers to the numbered uncertainty categories in the text.

1. UNCERTAINTY OF TRAINING INSTANCE The training instance x_t defines where in input space the model is trained to give a certain label y_t. As these data are often given by sensor readings or as output signals of prior processing modules, the training instances might be uncertain. For sensors it is possible that e. g. sensor-noise, drift or a fault lead to a gradually or totally wrong instance x_t. Or, prior processing modules, e. g. other learning systems in a cascaded structure, result in an uncertain output generating the next uncertain input. Thus it gets ambiguous, *where* the model should be trained within the input space using the given example (see Fig. 9).

2. UNCERTAINTY OF TRAINING LABEL The training label y_t defines the desired output value, the model should give at a certain point x_t in the input space. In regression tasks, these training labels y_t may be subject to similar uncertainties as the training instances x_t. Again, this results in an uncertainty, but about *what* the model should be trained to using the example (see Fig. 9).

3. UNCERTAINTY OF MODEL STRUCTURE The approximation is done by some model structure restricted by the basis functions ϕ and its input vector x. Hence it is only capable of representing a limited set of functions, e. g. polynomials of a certain order or piecewise linear functions with fixed number and position of reference points (see Fig. 10). Either this model structure is not expressive enough causing conflicting information about the correct choise of a parameter vector ω. Or the structure is too expressive causing either the well known overfitting problem or some parts of the parameter vector ω not to be set up correctly. Just as well, the input vector may be either chosen too small, i. e. there are unobserved influences, or too big, i. e. there

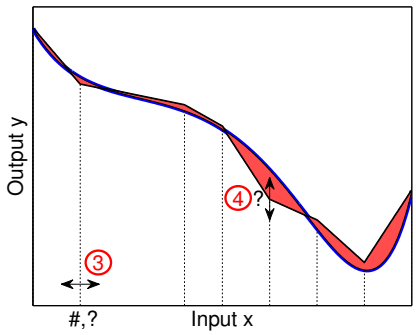


Figure 10: Uncertainty of the model. The model is shown as a black curve approximating the blue curve with some error (red). The numbering refers to the numbered uncertainty categories in the text.

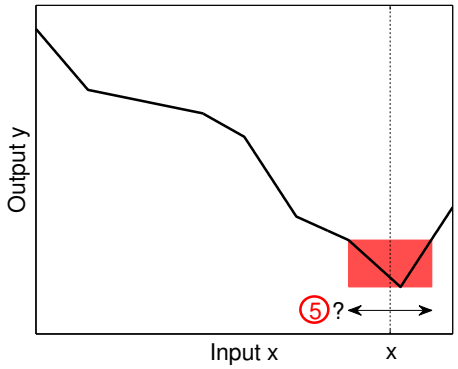


Figure 11: Uncertainty of the evaluation instance. The model is shown as a black curve and the uncertain range of the instance for evaluation in red. The numbering refers to the numbered uncertainty categories in the text.

are irrelevant features. Consequently, it is uncertain, *how well the approximation* of the target function can be achieved with the chosen model structure ϕ.

4. UNCERTAINTY OF PARAMETER VECTOR Corresponding to the model structure, the parameter vector ω determined by the learning algorithm is influenced in many ways. The learning algorithm, which chooses $\omega \in \Omega$, and its optimization criterion might by inappropriate. The settings of this learning algorithm, e. g. the learning rate or a regularization term, can be chosen suboptimally and thus lead to systematic remaining errors. Additionally, the training instances x_t need not cover the complete input space, so that some parameters might not be set up properly as not enough information is present. Similarly, the training data may contain conflicts, i. e. different labels y_t at close instances x_t. Hence, it is uncertain, *how well a parameter vector* ω fits to the optimal target vector (see Fig. 10).

5. UNCERTAINTY OF PREDICTION INSTANCE In contrast to training, at prediction, the input $\mathbf{x}$ defines for which instance a label should be predicted. These instances again may have the same sources of uncertainty as the training instance $\mathbf{x}_t$. But now for a fixed parameter vector $\boldsymbol{\omega}$, there is uncertainty, *where* the model should be evaluated (see Fig. 11). The uncertainty of the instance does not necessarily define a closed interval as shown in Fig. 11. Other uncertainty representations are possible as discussed in Section 1.5.

All sources of uncertainty mentioned above influence the model evaluation and hence the risk of a wrong prediction $\hat{\mathbf{y}}$. This risk can be divided into two categories. Either the uncertainties result in (some degree of) *ignorance* so that it is not known which prediction $\hat{\mathbf{y}}$ should be given as not enough examples supplied information for the respective point $\mathbf{x}$ in input space. This means there is epistemic uncertainty about the prediction. Or the uncertainties result in (some degree of) *conflict* so that multiple labels $\hat{\mathbf{y}}$ are possible as the examples or the model evaluation are contradictory. Hence, there is aleatoric uncertainty about the prediction. Both categories are gradual in nature as ignorance increases with the distance of the evaluation point $\mathbf{x}$ to training instances $\mathbf{x}_t$ in input space and conflict increases with the amount of plausible labels in output space. This notion of conflict and ignorance was introduced in [61] for classification problems, but can also be easily applied for regression tasks as presented in [11].

If the predicted label $\hat{\mathbf{y}}$ is uncertain and its uncertainty cannot be compensated within the learning system, an additional information about its degree of uncertainty is valuable to increase safety by successive processing modules. This way it is possible to explicitly react to uncertainties of a prediction, irrespective of their source, and maintain a safe operation of the whole system. Hence it is of interest not only to minimize the uncertainty of the learning system, but also to express remaining degrees of uncertainty of the predicted label $\hat{\mathbf{y}}$.

1.4 REQUIREMENTS

From the different aspects discussed in this chapter, the following list of requirements for a reliable on-line learning system can thus be derived.

a) FAST ADAPTATION The learning system has to perform well right from the start. So a fast adaptation to the examples is necessary which is reflected in the cumulative loss to be minimized.

b) CONTINUOUS ADAPTATION In non-stationary environments the input-output relation changes through time by shift and/or drift. Hence, the learning system has to adapt to new data and forget

older data when necessary to account for those changes. This is reflected as well in a low cumulative loss.

c) PROTECTION OF GATHERED KNOWLEDGE With the incorporation of new examples into the learning system, typically the global model is affected and it is important to make only local changes and to protect the global model as one example only gives local information. This property is reflected in the reproduction of the training examples.

d) LOW LIMITED COMPUTATIONAL DEMAND The computational demand with respect to memory and time complexity should be limited and low to allow learning on embedded systems, high speed data streams, and big data.

e) ROBUSTNESS AGAINST UNCERTAIN EXAMPLES Outliers and noise of the presented examples should affect the learning system as little as possible. Examples that are uncertain should be faded out (gradually) from learning accordingly.

f) ROBUSTNESS AGAINST WRONG PARAMETERS A wrong parameter vector should not affect the quality of learning, i. e. the learning system should not try to keep the knowledge of wrong parameters. Additionally it should still yield a reliable prediction in presence of wrong parameters.

g) ROBUSTNESS TO AN INAPPROPRIATE MODEL STRUCTURE A model structure that is too expressive or not expressive enough for the true input-output relation should not affect the predictive reliability but only its quality. So overfitting should be prevented and stability in case of underfitting achieved.

h) ROBUSTNESS TO UNCERTAIN INSTANCES If an uncertain instance is presented to predict its label, a reliable prediction should be given. This means the prediction must be insensitive to variations of the values for uncertain input dimensions.

i) REFLECT UNCERTAINTY OF MODEL The learned model, i. e. the parameter vector, is subject to various uncertainties during learning. These uncertainties must be estimated on the one hand to allow supervision by an expert and on the other hand to react to these uncertainties at learning and prediction.

j) REFLECT UNCERTAINTY OF PREDICTION If the above robustness to several uncertainties cannot sufficiently increase the reliability of a prediction, its remaining uncertainty must be reflected to allow other processing modules or a supervisor to utilize this additional information. So if the uncertainty cannot be dealt with at this low level, a higher level should be able to deal with it. Consequently, a reliability measure for each individual prediction is necessary.

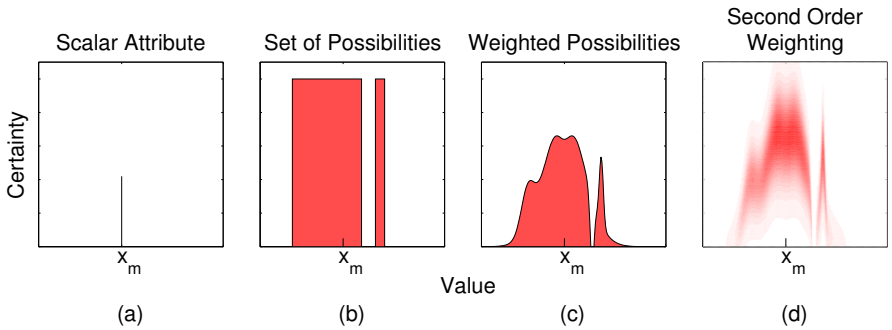


Figure 12: Four abstractions of uncertainty measures can be distinguished, assigning a value x_m some certainty.

1.5 REPRESENTING UNCERTAINTY

It was argued in the previous sections that uncertainties have a severe influence on learning systems that cannot be neglected. Thus, the learning system has to deal with its uncertainties internally and provide additional information about its uncertainties to its user. While the notion of uncertainty is immediately associated with probability, various other uncertainty measures have been developed. An abstract view on these uncertainty measures is presented here in the following. A general survey trying to summarize all representations within the common framework of *generalized information theory* is presented in [68].

1.5.1 *State of the Art Methods*

The simplest way of representing the uncertainty of a piece of information is to attribute the information with a scalar (Fig. 12 (a)). The scalar expresses the certainty of an information. This method is rarely used, but some research has been done starting with *certainty factors* [100] modeling the belief in a hypothesis in the range of $[-1, 1]$. Another scalar approach is introduced with *pain levels* by [50] to gradually model sensor failures at runtime. The fault status of each sensor in a system is estimated gradually based on the history of its measurements resulting in an additional signal attribute given to an injury agent declaring that the sensor is either working or broken. In this approach, each sensor with a pain level above some defined threshold is discarded from further processing and it is possible for a sensor to recover and being reintegrated into processing. More recently, *health signals* [9, 29, 67] are used to attribute the informational and operational health status of system modules in robotics. They represent the degree of all kinds of uncertainties in a unified way normalized to the range of $[0, 1]$. This approach is generalized to the *trust management* approach in [1, 2, 3, 13] attributing potentially

any information within a system architecture with a *trust signal*. With this simple attribution the engineering and processing complexity increases only minimally.

The next more complex way of representing the uncertainty of an information is to form a set of all possible values (Fig. 12 (b)) or, as a special case, to require the value to lie within an interval through interval analysis [84]. The use of this technique has been formalized in *info gap decision theory* [19]. It allows to consider all possible values of an information and their respective results, but it has several drawbacks. Depending on the restrictions on the used sets, the complexity of calculations increases and additional information is necessary to estimate the set of possible values, e. g. from a single measurement. Furthermore, in the end of processing the set of possible values has to be projected in a meaningful way to a single scalar value as the output.

Further representations elaborate on the set of possible values by giving each possible value a weight (Fig. 12 (c)). The most well-known approach is *probability theory*, where each value of the set is given a probability similar to the frequency of occurrence [69]. This approach gives a well-defined theory of calculation, but again generally the calculations get much more complex and additional knowledge is necessary to map a measurement to a certain probability distribution. Often a simplifying assumption is used, namely that of a normally distributed random variable. In this case the additional assumption only needs to give an estimate of the mean value and its standard deviation, hence the calculations remain in their original complexity. Yet, with a normal distribution only bounded random variables like noisy measurements can be represented properly, but not all sources of uncertainties can be considered adequately this way [1].

Similar to probability theory, *fuzzy set theory* gives weights to the elements of a set but with a different semantical meaning [66, 122], which can be viewed as a possibilistic approach, representing the vagueness of an information. The fuzzy membership function is hence not restricted to a total of one over all elements of the set. Again the characteristic function, which represents the degree of membership across the set has to be derived from additional information, e. g. by the designer, and the complexity of calculations increases.

In recent years, another layer of uncertainty is added for increased expressiveness and the weights of the set's elements are viewed as a weighted set as well, i. e. a second order of weighting (Fig. 12 (d)). In the case of probability theory this results in *imprecise probabilities*, e. g. *Dempster-Shafer theory* [43, 111, 113] where each probability gets an upper bound called plausibility and a lower bound called belief. Similarly fuzzy sets are extended to *type-2 fuzzy sets* [31]. But obviously, the extensions increase the complexity even more, both for calculation and for additional assumptions to derive and to engineer the

distributions. And the designer is hampered in controllability and interpretability of the uncertainty measure.

In a nutshell, the four abstractions of uncertainty measures have different amounts of expressiveness, but with increased expressiveness the computational complexity as well as engineering complexity, i. e. necessary amount of data or a priori knowledge, increases, too. Whereas the scalar attribute still uses the original value, the other abstractions open up a set of values which typically has to be merged finally to a single value at some point of processing, e. g. if a controller accesses an actuator or a certain amount of energy has to be bought for the power grid. Furthermore, it complicates the understandability of the outcome for an expert. That is why many applications of these uncertainty measures revert to simplifications, e. g. closed intervals, normal distributions, trapezoidal sets, interval valued probability, or interval type-2 fuzzy sets, instead of using the full expressiveness of the respective method.

Consequently, in this work the simplest uncertainty measure, i. e. a scalar attribute, is used to represent uncertainties in learning systems by the means of trust management. The relevant details of this approach will be introduced in the next section. And finally, it will be shown if and how far this simple kind of uncertainty handling is suited to deal with all five categories of uncertainty listed in Section 1.3.

1.5.2 *Trust Management Approach*

The aim of trust management [1, 2] is to incorporate the knowledge about uncertainties in a way that allows a system to flexibly adapt to dynamic uncertainties, but which is easily calculated and intuitively engineered. The intention is to find a generic and general mechanism for considering the uncertainties of information explicitly throughout the system but with low overhead. With this concept different signal and uncertainty qualities are combined in a way such that the effect of uncertainty handling is uniform and comprehensible. This way, an expert can easily incorporate his knowledge about uncertainties into the system and understands what the system does in uncertain situations.

Trust management was originally developed for embedded systems but can be applied to any other system architecture the same way. The basic concept is to model the uncertainties explicitly with a unified semantics. This is done by an additional attribute, which is associated to the respective signal carrying the uncertain information. This attribute is called *trust signal*. A trust signal is a meta-information to a normal signal, e. g. an input or output of a learning system, which depends on uncertain information and is processed along the signal flow. It has a scalar value from the interval $[0, 1]$, called the *trust level*,

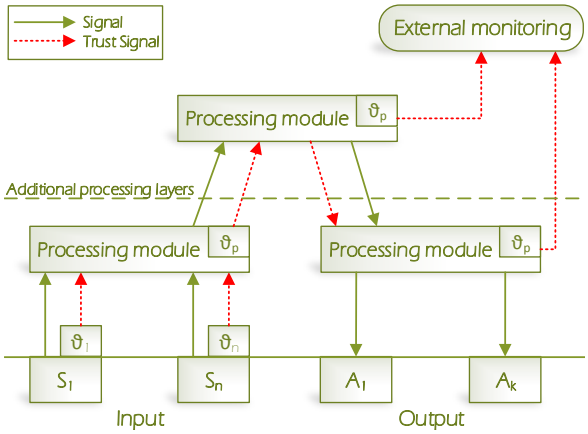


Figure 13: Schematic Trust Management Architecture.

and indicates the trustworthiness of the signal it is associated with (compare Fig. 12 (a)). The trust level of a signal is used explicitly in its further processing. If the trust level of a signal is zero, the signal must have no influence on the further calculations. Because it corresponds to no trustworthiness, a safe and robust fallback strategy should be used then. In contrast to that, the effect of a signal should be normal if its trust level is equal to one, because it relates to absolute trustworthiness, i. e. there is no doubt about the correctness of the respective signal value. The influence should be gradual in between.

This way, a unified description of different kinds of uncertainty, i. e. aleatoric as well as epistemic, with one representation is possible. For example a single sensor measurement can be influenced by different factors. The sensor's basic noise or additional noise dependent on the temperature cause aleatoric uncertainty. Whereas a measurement at the limit of its range or a sensor failure cause epistemic uncertainty. All of them influence the certainty of the resulting measurement and hence the information about the true value and are combined in one estimate of its trustworthiness.

Because the trustworthiness of input signals in general influences also the trustworthiness of a module's output, the relevant trust signals are fused to an outgoing trust signal that accompanies the processing result. Additionally, such trust signals can be used by higher decision layers to change the system behavior or for external monitoring like in the ORCA- [28, 29, 85] and COBRA-architecture [3]. This is illustrated by the schematic trust management architecture in Fig. 13. At the bottom is the system's input-output interface, e. g. the hardware layer of an embedded system, i. e. its sensors and actuators. For every sensor signal S_i, a trust signal ϑ_i is generated. In the simplest case, it reflects for instance the noise amplitude. The vagueness or trustworthiness, respectively, of a particular signal is then expressed through a trust level according to the signal to noise ratio. It is im-

portant to note that no modification of the particular signal value is performed in order to maintain transparency and generality.

All in all by accompanying potentially every signal in the signal flow with a trust signal, the influence of untrusted sources should be faded out in order to increase the general robustness. Whereas there is no loss of precision under fully trustworthy conditions. The trust signal flow thus extends over the whole system's architecture until the outputs, e. g. actuators, have to be addressed with concrete values without a trust level.

1.5.3 *Using Trust Signals*

One use case of a trust signal is to provide it for external monitoring, e. g. by a supervisory system or system operator. But to get a benefit from the modeling of uncertainties within the system itself, the trust signals need to influence the internal processing. This means they change the local and global behavior of the system by utilizing the knowledge about the uncertainties or trustworthiness, respectively. To enable this, either redundant information, to replace the uncertain information, or redundant actions, to choose a more robust action, need to be available in a system. This can be done either by switching between alternative strategies, or by adapting a strategy gradually. Switching the whole strategy can pose a more powerful tool but it may lead to complex dynamic interaction patterns between trust signals and the underlying dynamic system.

The gradual adaption to trust signals is based on the two above mentioned axiomatic cornerstones. If the trust level of a signal is zero, its value must not influence the calculation. And if the trust level is one, the value should have its normal influence on the result. Between these two cases, a processing algorithm should yield a continuous, monotone gradual blending. So the less trustworthy a value is the less it should impact the processing results. If none of the relevant signals is trustworthy at all, the algorithm should provide a worst case result, e. g. a default value which does not change the system state or brings the system into a safe state. Handling this worst case fallback on an architectural level is discussed in [3].

1.5.4 *Propagation and Fusion of Trust Signals*

Along the signal flow, the output of a module should be accompanied by a trust signal as well. So the trust signals of the used information have to be fused for the trust signal of an output with respect to the present calculation of the algorithm. There are three different categories of trust signal processing. If non-redundant signals are used and every information is crucial to the result, the trust at the output must not excel the lowest trust level of any used information, but

might even be lower. In contrast, if a module fuses redundant information to get a more reliable result, the trust level can increase within the module depending on the degree of conflict or conformance between the redundant sources. So these categories directly map to the trustworthiness of the outcome:

a) non-redundant relevant information $\rightarrow$ decreasing trust,
b) redundant conflicting information $\rightarrow$ averaging trust,
c) redundant conforming information $\rightarrow$ increasing trust.

There exist only two restrictions on such trust signal fusion. The fusion must be monotone and continuous. This means that the fused trust level for a set of used trust levels $\{\vartheta_i\}$ is equal to or lower than the trust level for another set $\{\tilde{\vartheta}_i\}$ with at least one higher trust level and that it makes no abrupt changes. The second restriction is that a module cannot produce trust out of nothing. So, if all trust levels of relevant information are zero the outgoing trust level must be zero, too.

Thus the fusion of n_I trust signals can be done by a continuous function $F : [0,1]^{n_I} \rightarrow [0,1]$ which satisfies the following properties for two sets of trust levels $\{\vartheta_i\}_{n_I}$ and $\{\tilde{\vartheta}_i\}_{n_I}$:

$$\vartheta_i \leqslant \tilde{\vartheta}_i \; \forall \; i \in \{1,\ldots,n_I\} \quad \rightarrow \quad F(\{\vartheta_i\}_{n_I}) \leqslant F(\{\tilde{\vartheta}_i\}_{n_I}) \tag{6}$$

$$\vartheta_i = 0 \; \forall \; i \in \{1,\ldots,n_I\} \quad \rightarrow \quad F(\{\vartheta_i\}_{n_I}) = 0 \tag{7}$$

The monotony property directly leads to the conclusion that a maximal trust in the result is achieved if all input values are fully trusted. This maximal trust might not be equal to one because of internal uncertainties of the processing module, e. g. a learning module might not be fully trusted due to a restricted expressiveness.

A special function class that meets properties a) and c) are t-norms and s-norms [82]. A t-norm is a suitable fusion method for processing of signals without redundancy (a) and s-norms for redundant conforming information (c). Complementary to those norms, compensatory operators lie in between and produce an averaged trust level [83], which relates to redundant conflicting information (c). Hence these functions constitute suitable fusion methods for the above mentioned categories, to determine the trust level of an output signal.

1.5.5 *Properties of Trust Management*

Representing the trustworthiness or uncertainty, respectively, by a single scalar attribute is a way to incorporate uncertainties which still only needs to process the sole value of the information. The trust level attribute can then be used to rate and control the trustworthiness of the system's behavior. Thus the engineering and processing

complexity increases only minimally. The influence of a trust signal is easy to understand and interpret and leads to a comprehensible behavior of the uncertainty treatment. With its unified semantics the trust management approach is applicable throughout a system and constitutes a good trade-off between computational effort and engineerability. In case of redundancy of the in- or output an increased quality can be expected. Otherwise it is reflected that the output again cannot be trusted if a reduction of the uncertainty is not possible. Any processing module can be extended to incorporate trust signals either by fusing the trust levels aside the entire processing or by integrating them into it. And any extended module can then be integrated uniformly in a system wide trust management architecture.

1.6 GOAL AND OUTLINE OF THE THESIS

The goal of this work is to develop an on-line learning system meeting all the requirements of Section 1.4 that can be embedded in the framework of trust management. This comprises a reliable and robust on-line learning algorithm, a supervision of the model's trustworthiness and an assessment of the trustworthiness of each individual prediction.

In the next chapter on-line learning is discussed for the certain case, i. e. for all data and the parameter vector being certain. First an overview of state of the art approaches to on-line learning is given in Section 2.1 before the incremental risk minimization algorithm (IRMA) is introduced in Section 2.2. IRMA is formally analyzed in Section 2.3 and empirically investigated on synthetic data in Section 2.4 as well as on a real world application in Section 2.5. The chapter concludes with discussing the properties of the approach in Section 2.6.

In Chapter 3 the on-line learning setting is extended to explicitly represent uncertainties of the learning system. First the state of the art regarding uncertainty estimation in learning systems is reviewed in Section 3.1. Then the trusted parameters approach is introduced in Section 3.2 and formally analyzed in Section 3.3. Again empirical investigations and an application to the same real world example as before are presented in Section 3.4 before concluding with a discussion of the properties in Section 3.5.

In Chapter 4 the discussion of on-line learning is extended to the uncertain case with respect to the uncertainty categories presented in Section 1.3. Section 4.1 extends the IRMA approach to incorporate knowledge about the uncertainty of the parameter vector and the training example into the second order incremental risk minimization algorithm (SIRMA). This extension is as well formally analyzed in Section 4.2 and empirically investigated and applied to the real world example in Section 4.3. Finally, the consequences for the uncertain case are discussed in Section 4.4.

The last chapter provides an overall conclusion of this thesis with a discussion of the proposed approaches in relation to the state of the art in Section 5.1 and summarizes the main results in Section 5.2. Afterwards Section 5.3 gives an outlook on future directions, as this field of research still poses a plethora of relevant open questions.

2 ON-LINE LEARNING – THE CERTAIN CASE

In this chapter, the classical on-line learning case without any explicit treatment of uncertainties is examined. An overview of state of the art learning algorithms reveals an essential deficit which is covered by the presented approach. It accounts for the locality of an example and secures the globally gathered knowledge inherently. This way, reliable on-line learning is possible with any LIP model structure.

2.1 STATE OF THE ART ON-LINE LEARNING ALGORITHMS

2.1.1 *General Framework*

Most on-line learning algorithms have been developed for binary classification, i. e. the distinction of two non-ordinal labels. But they are often applicable (with slight variations) for regression, i. e. an ordinal real valued label, as well. Thus, an overview of on-line learning in general is given here to provide the key concepts.

The basic idea of each approach is based on a general functional

$$F(\boldsymbol{\omega}) = w_L \cdot L(y_t, \boldsymbol{\omega}^T \boldsymbol{\phi}(\mathbf{x}_t)) + w_D \cdot D(\boldsymbol{\omega}_t, \boldsymbol{\omega}) + w_C \cdot C(\boldsymbol{\omega}) \quad (8)$$

consisting of two to three weighted parts and choosing the new parameter vector such as to minimize this functional, i. e.

$$\boldsymbol{\omega}_{t+1} = \arg\min_{\boldsymbol{\omega}} F(\boldsymbol{\omega}). \quad (9)$$

The primary goal of the minimization is to be *corrective*, i. e. to minimize the instantaneous loss L on the example $(\mathbf{x}_t, y_t)$ presented in a step. The secondary goal is to be *conservative*, i. e. to minimize the distance D between the previous hypothesis and the new one. Obviously, these two goals are contradictory if the current hypothesis does not already fulfill the presented example. Optionally, in addition the complexity C of the chosen hypothesis is minimized. The importance of these goals is weighted by w_L, w_D, and w_C, respectively. Depending on the choice of the measures L, D, and C, different learning algorithms can be derived. These on-line learning algorithms can be distinguished by their update type regarding *multiplicative* or *additive* updates of the parameter vector.

2.1.2 *Multiplicative Updates*

The algorithms using a multiplicative update of the parameter vector assume the predicted label to be a weighted sum of "experts" which

each give their own prediction about the label. So the parameter vector represents the weights and consequently it is restricted to a convex combination of the "experts", i. e. $\omega_i \geqslant 0 \; \forall \; i = 1, \ldots, n, \; \sum_{i=1}^{n} \omega_i = 1$. For each example the error of each "expert" is observed. The more wrong an "expert" is, the more its weight is reduced in a multiplicative way. This idea was first introduced as the *winnow* algorithm [75] and the *weighted majority* algorithm [76]. But the restriction on the weight vector to be convex clearly also restricts the predictive ability of the algorithm. Hence, the *exponentiated gradient* [64] introduces an algorithm which allows positive and negative weights. For that purpose, two weight vectors are maintained simultaneously, one with positive, and one with negative influence, which are both restricted to a convex combination each. The multiplicative update rule can be derived with the distance D being the relative entropy, also known as Kullback-Leibler divergence [70]

$$D_{KL}(\boldsymbol{\omega}_t, \boldsymbol{\omega}) = \sum_{i=1}^{n} \omega_i \ln \frac{\omega_i}{\omega_{t,i}}. \tag{10}$$

As stated in [64, 75], due to upper bounds on the loss the main advantage of multiplicative algorithms comes into play if the parameter space is high dimensional and only a sparse parameter vector, depending on a small number of features, is necessary for the approximation. Consequently, a multiplicative update is not suited for learning a low to medium dimensional function with a fixed model structure but rather in high dimensional cases where more features are available than necessary and no model structure is used. A typical application domain is text analysis through a bag of words where the instances are vectors of several thousand inputs and only some are relevant for the text analysis task at hand [72]. Additionally, the restrictions of a convex linear approximation and that zero parameters never get back any influence are very strong. Thus, additive updates are of greater importance for regression with LIP model structures.

2.1.3 *First Order Additive Updates*

In contrast to the multiplicative updates, additive updates of the parameter vector add (or subtract) each presented instance to the parameter vector. They date back to the *perceptron* algorithm [20, 90, 95] and the *Widrow-Hoff* algorithm, also known as *least mean squares* [115].

These algorithms can be divided into *first order* and *second order* learning algorithms [18]. First order algorithms directly update the parameter vector with first order information, i. e. no derivative, using the example in the same way all the time. This way, the algorithm has no memory except the parameter vector $\boldsymbol{\omega}$. So the same example and the same parameter vector at different time steps t in the data sequence will result in the same update to the new parameter

Table 1: Variants of the passive-aggressive algorithm.

Variant	β
PA	$\frac{1}{\|\boldsymbol{\phi}(\mathbf{x}_t)\|^2}$
PA-I	$\min\left(C_a, \frac{1}{\|\boldsymbol{\phi}(\mathbf{x}_t)\|^2}\right)$
PA-II	$\frac{1}{\|\boldsymbol{\phi}(\mathbf{x}_t)\|^2 + \frac{1}{2C_a}}$

vector $\boldsymbol{\omega}_{t+1}$. In contrast, second order algorithms also adapt their adaptation dynamically and data dependent. This means, the effect of incorporating an example into the parameter vector is changed as well during learning.

First order algorithms are based on a gradient descent on the loss function [121]. This results in a general parameter update of the form

$$\boldsymbol{\omega}_{t+1} = \alpha \cdot \boldsymbol{\omega}_t + \beta \cdot (y_t - \hat{y}_t) \cdot \boldsymbol{\phi}(\mathbf{x}_t) \tag{11}$$

with the two coefficients α for weighting the old parameter vector $\boldsymbol{\omega}_t$ and β for the gradient step size.

The *perceptron* algorithm [20, 90, 95] adapts its parameter vector according to (11) with $\alpha = 1$ and an adjustable step size β as a hyper-parameter. This method is extended in the work on the passive-aggressive algorithm (PA) [36] for classification but it can be readily applied to regression. The idea is to minimize the change in the parameter vector (passiveness) while getting a zero loss for the current example (aggressiveness), resulting in a projection of the current parameter vector onto the hyperplane of all possible solutions (compare Fig. 6). The notion of passiveness corresponds to minimizing the distance D while aggressiveness corresponds to minimizing the loss L in (8).

For PA the weight of the old parameter vector is again $\alpha = 1$ and three different variants of the step size β were proposed (see Table 1). The first (PA) normalizes the step size so that a zero loss for the example is achieved through a full projection. The other two variants (PA-I and PA-II) allow to decrease the influence of an example through a hyper-parameter $C_a > 0$. It therefore reflects the aggressiveness of the algorithm.

In all cases the distance D between the new and old parameter vector is measured with respect to the squared Euclidean norm according to

$$D_E(\boldsymbol{\omega}_t, \boldsymbol{\omega}) = \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|^2 = \sum_{i=1}^{n} (\omega_{t,i} - \omega_i)^2. \tag{12}$$

The same learning algorithm was also derived in the framework of *follow the proximally regularized leader* [80, 81].

Another distinct first order approach is the relaxed online maximum margin algorithm (ROMMA) [74] which was developed only for exact binary classification. It is based on the idea of choosing the new parameter vector $\boldsymbol{\omega}_{t+1}$ such that all previous examples are still correctly classified resulting in constraints that define a convex polyhedron [22, 107]. This convex polyhedron is approximated by the last parameter vector. In this case, a special feature of the binary classification task is used, i. e. that one example is correctly classified by a complete half space of the parameter space, so no transfer to regression is possible.

The family of *p-norm* algorithms [52, 65] generalizes additive and multiplicative updates in one framework, using the generic Bregman divergence [26] as a distance measure. The Kullback-Leibler divergence (10) as well as the squared Euclidean distance (12) are special cases of the Bregman divergence and are thus subsumed in this framework.

2.1.4 *Second Order Additive Updates*

As first order methods always adapt uniformly to new examples, they can quickly adapt to changing conditions but are thus prone to noise and outliers. To cope with these, second order algorithms extend the approach of first order algorithms by introducing second order information through an additional matrix Σ_t into the update by

$$\boldsymbol{\omega}_{t+1} = \alpha_1 \cdot \boldsymbol{\omega}_t + \beta_1 \cdot (y_t - \hat{y}_t) \cdot \Sigma_t \boldsymbol{\phi}(\mathbf{x}_t) \tag{13}$$

$$\Sigma_{t+1} = \alpha_2 \Sigma_t + \beta_2 \Sigma_t \boldsymbol{\phi}(\mathbf{x}_t)^T \boldsymbol{\phi}(\mathbf{x}_t) \Sigma_t \tag{14}$$

thus adapting the influence of an example on the different parameters dynamically and individually. The matrix Σ_t can be interpreted as parameter-wise adaptive learning rates which decrease over time, i. e. the amount of examples presented.

Following up on the idea of ROMMA, the *ellipsoid method* [119] maintains an ellipsoid approximation of the polyhedron of parameter vectors consistent with all previous examples. Likewise, it is not directly transferable to regression. The ellipsoid is represented by the matrix Σ_t. A similar idea views the parameter vector not as a single vector but as a random variable drawn according to a normal distribution $\mathcal{N}(\boldsymbol{\omega}_t, \Sigma_t)$. This approach is known as confidence weighted learning (CW) [47] as it keeps track of the confidence of each parameter vector being the target vector, represented by its probability. It uses the Kullback-Leibler divergence (10) of the new parameter distribution to the old one as a measure of distance D between the new hypothesis and the previous one. In contrast to the multiplicative update where the parameter vector itself is the distribution whose divergence is minimized, CW uses the normal distribution of the parameter vector $D_{KL}(\mathcal{N}(\boldsymbol{\omega}_t, \Sigma_t), \mathcal{N}(\boldsymbol{\omega}, \Sigma))$. As in its classical form it is

only applicable for binary classification, further development of this method led to *Exact Convex Confidence Weighted Learning* [37] and *Exact Soft Confidence Weighting* [112] making the method applicable for regression as well.

This approach is extended by the adaptive regularization of weights (AROW) [38, 105] which additionally minimizes the amount of uncertainty about the parameter vector in each learning step by a complexity measure $C = \phi(x_t)^T \Sigma \phi(x_t)$, i.e it forces the normal distribution to sharpen. In the case of regression, AROW results in similar update equations as the widespread recursive least squares (RLS) algorithm [21, 48, 62] where Σ_t is interpreted as the covariance matrix of the parameter estimation error. RLS thus minimizes (by definition) the squared error to all presented examples, regardless of their age. To reduce the influence of older examples and allow for a continuous adaptation, a forgetting factor $0 < \lambda \leqslant 1$ is introduced in RLS, known as recursive least squares with exponential forgetting [77]. RLS with forgetting results in the update with

$$\alpha_1 = 1 \quad , \quad \beta_1 = (\lambda + \phi(x_t)^T \Sigma_t \phi(x_t))^{-1} \tag{15}$$

$$\alpha_2 = \lambda^{-1} \quad , \quad \beta_2 = (\lambda^2 + \lambda \phi(x_t)^T \Sigma_t \phi(x_t))^{-1}. \tag{16}$$

With a forgetting factor close to one, the noise stability is high, but it is badly suited for continuous adaptation. Lowering the forgetting factor in principle allows for better continuous adaptation, but decreases the stability at the same time [88].

Alternatively, to allow RLS to adapt to non-stationary situations, the covariance matrix is reset to the identity after a given number of examples was presented, thus increasing the adaptiveness again [35, 53, 96]. Similarly the AROW approach is extended with a covariance reset [106], but here resetting of the second order information is done, based on spectral properties of the covariance matrix, i. e. if the lowest eigenvalue drops below a lower bound.

A different second order approach is the second order perceptron (SOP) [33]. It performs a whitening transform on the input data to reduce the correlation matrix of the transformed data to the identity. Thus, every example can be used more effectively to update the parameter vector. The resulting update equation again is similar to that of RLS and AROW.

Another second order approach, called Gaussian herding (GH) [39], replaces the Kullback-Leibler divergence in such a way that the movement of the parameter vector distribution is led by a velocity field, thus reducing quick changes in the distribution and becoming more robust to high noise levels in the data. Here, often not the complete matrix Σ_t is stored but only a diagonal projection (see e. g. [37, 39]) in order to reduce the computational complexity.

2.1.5 *Consequences*

The above dicussion of the state of the art leads to the conclusion that of all additive algorithms, the gradient based PA and its variants are most suitable, if the data are not subject to strong noise and if a continuous adaptation to changing conditions is needed in a regression task. If the noise level on the data is high, a second order approach is more suitable. Then AROW, SOP, and RLS are possible choices which are very similar. RLS is the most widespread method as it is optimal regarding the mean squared error. Furthermore, GH is especially suitable for high noise levels [39]. With the forgetting factor of RLS, a trade-off between robustness to noise and adaptability to changes is possible. Yet, practically a good setup of the forgetting factor is hard to find as it is very sensitive [88]. Typically the forgetting factor is chosen from $\lambda \in [0.9, 1]$ and slight changes on the third decimal point affect the stability severely.

But all approaches have a common drawback. The parameter adaptation is chosen in the realm of parameter space. As described in Section 1.1, usually a model structure is used to transform from input space to parameter space to get a more expressive linear model. This results in a non-linear influence of the input vector on the output value. Yet, a common basic characteristic is that all state of the art methods treat every parameter in the same way regardless of its influence on the output. But if the mapping from input to parameter space consists of non-local basis functions, e. g. polynomials, or basis functions with different amounts of influence on the output, e. g. a GLT with not equally spaced grid positions, treating the parameters equally is not likely to result in the desired behavior and will not satisfy all of the requirements of Section 1.4. That is because the passive part of all methods always tries to change the parameter vector as little as necessary. Consequently, the parameters with the highest impact on the output are changed the most and the global input-output relation is likely to change much more than necessary, especially also apart from the validity of the example. Therefore, fatal forgetting as well as overfitting is very likely to occur which is dangerous to the performance and robustness of the learning system. Especially first order algorithms like PA show a poor performance when learning a polynomial model structure, i. e. with globally effective parameters. The second order improvement of RLS helps to deal with these model structures on the long run, but neither is it reliable at every step in the learning process nor is it stable with forgetting enabled. So a reliable continuous adaptation is not possible with these methods.

One countermeasure against overfitting is *regularization*. The third term of (8), i. e. the complexity measure C, accounts for this. A widely

applied technique is *Tikhonov regularization* [42, 86, 104] which incorporates the complexity penalty

$$C(\boldsymbol{\omega}) = \|\Gamma\boldsymbol{\omega}\|^2 \tag{17}$$

with a suitably chosen Tikhonov matrix Γ which is often set to the identity matrix $\Gamma = \mathbb{1}$. This way the resulting functional behavior can be shown to be as flat as possible, thus preventing overfitting [93]. But as this regularization does not include any knowledge about prior learning examples, it is usually only suited for learning with a set of training data and not for on-line learning and hinders the predictive quality.

2.2 INCREMENTAL RISK MINIMIZATION APPROACH

2.2.1 *General Approach*

The basics of the incremental risk minimization algorithm (IRMA) were introduced in [5, 12]. It is inspired by the *risk functional* for batch learning [107]

$$R(h) = \int_X L(y, h(\mathbf{x}))\, dF(\mathbf{x}, y) \tag{18}$$

that describes the risk of loss for a chosen hypothesis $h \in \mathcal{H}$ given the data distribution $F(\mathbf{x}, y)$. For a set of examples $(\mathbf{x}_i, y_i), i = 1, \ldots, n_d$ drawn independently form the distribution $F(\mathbf{x}, y)$, the integral can be approximated by the *empirical risk* [107]

$$R_e(h) = \frac{1}{n_d} \sum_{i=1}^{n_d} L(y_i, h(\mathbf{x}_i)) \tag{19}$$

$$= \frac{1}{n_d} \sum_{i=1}^{n_d-1} L(y_i, h(\mathbf{x}_i)) + \frac{1}{n_d} L(y_{n_d}, h(\mathbf{x}_{n_d})). \tag{20}$$

In the on-line case older examples, i. e. $i = 1, \ldots, n_d - 1$ are not available. But they are embedded in the hypothesis h_t, especially its encoded input-output relation, as they were learned before. So the output resulting from the current hypothesis h_t can be used to describe the risk of a loss regarding those examples for a newly chosen hypothesis h. Hence, IRMA uses the change of the model output as a distance

$$D(h) = \frac{\sigma_t}{2} \cdot \int_X L(h_t(\mathbf{x}), h(\mathbf{x}))\, d\mathbf{x} \tag{21}$$

on a bounded input space X as the distance measure for the newly chosen hypothesis. In contrast to the state of the art, the change of the hypothesis is thus not measured by the change of the parameter vector itself, but its influence on the change of the global input-output

relation. This way, fatal forgetting is prevented effectively. Together with the example loss the incremental approximation of the risk functional is given by

$$R_{inc}(h) = \frac{\sigma_t}{2} \cdot \int_X L(h_t(x), h(x))dx + \frac{1}{2}L(y_t, h(x)) \tag{22}$$

with a weighing factor $\sigma_t > 0$ to choose the new hypothesis according to

$$h_{t+1} = \arg\min_h R_{inc}(h) \,. \tag{23}$$

The factor σ_t can be seen as to steer the stiffness of the model. The bigger its value is, the more a change of the model output is punished. If a lot of previous examples are subsumed by the parameter vector and hence the contribution of the current example is comparably small, a big value of σ_t accounts for this evidence. In contrast, in the initial learning phase with no or only low data background, this weighing factor should be low, thus putting more weight on the present example. Consequently it should be chosen as a monotonically increasing value ($\sigma_t \geqslant \sigma_{t-1}$) as the learning process progresses.

2.2.2 *Application to LIP Regression*

So far the definition of IRMA in (22) is generally applicable for any learning system maintaining a hypothesis h. Using the squared loss of (4) for regression and the general LIP model given by (2) as the hypothesis h, the risk functional (22) takes the form

$$R_{inc}(\omega) = \frac{\sigma_t}{2} \cdot \int_X ((\omega_t - \omega)^T \phi(x))^2 dx + \frac{1}{2}(y_t - \omega^T \phi(x_t))^2 \,. \tag{24}$$

Minimizing this incremental risk functional yields the update of the parameter vector in each step. Solving the equation for a zero partial derivative

$$\frac{\partial R_{inc}}{\partial \omega_i} = 0 \quad \forall\, i \in [1; n] \tag{25}$$

to minimize (24) results in

$$\begin{aligned} &\frac{\partial}{\partial \omega_i}\frac{\sigma_t}{2} \cdot \int_X ((\omega_t - \omega)^T \phi(x))^2 dx + \frac{1}{2}(y_t - \omega^T \phi(x_t))^2 \\ &= \sigma_t \cdot \int_X (\omega_t - \omega)^T \phi(x)\phi_i(x)dx + (y_t - \omega^T \phi(x_t))\phi_i(x_t) = 0 \end{aligned} \tag{26}$$

which can be rewritten in vector form:

$$\sigma_t \cdot \int_X (\omega_t - \omega)^T \phi(x)\phi(x)dx + (y_t - \omega^T \phi(x_t))\phi(x_t) = 0 \tag{27}$$

Using the definition of matrices A

$$(A)_{i,j} = \int_X \phi_i(\mathbf{x})\phi_j(\mathbf{x})d\mathbf{x} \tag{28}$$

and $B(\mathbf{x}_t)$

$$(B(\mathbf{x}_t))_{i,j} = \phi_i(\mathbf{x}_t)\phi_j(\mathbf{x}_t), \tag{29}$$

this gets

$$\sigma_t A(\boldsymbol{\omega}_t - \boldsymbol{\omega}) + \boldsymbol{\phi}(\mathbf{x}_t)y_t - B(\mathbf{x}_t)\boldsymbol{\omega} = 0 \tag{30}$$

$$\sigma_t A\boldsymbol{\omega}_t - \sigma_t A\boldsymbol{\omega} + \boldsymbol{\phi}(\mathbf{x}_t)y_t - B(\mathbf{x}_t)\boldsymbol{\omega} = 0 \tag{31}$$

$$\sigma_t A\boldsymbol{\omega}_t + \boldsymbol{\phi}(\mathbf{x}_t)y_t = \sigma_t A\boldsymbol{\omega} + B(\mathbf{x}_t)\boldsymbol{\omega} \tag{32}$$

$$A\boldsymbol{\omega}_t + \frac{1}{\sigma_t}\boldsymbol{\phi}(\mathbf{x}_t)y_t = (A + \frac{1}{\sigma_t}B(\mathbf{x}_t))\boldsymbol{\omega} \tag{33}$$

as a condition for the critical point. Furthermore, as

$$\begin{aligned}\frac{\partial^2}{\partial\boldsymbol{\omega}^2}\frac{\sigma_t}{2}\cdot\int_X((\boldsymbol{\omega}_t - \boldsymbol{\omega})^T\boldsymbol{\phi}(\mathbf{x}))^2 d\mathbf{x} + \frac{1}{2}(y_t - \boldsymbol{\omega}^T\boldsymbol{\phi}(\mathbf{x}_t))^2 \\ = \sigma_t\cdot\int_X \boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{x})d\mathbf{x} + \boldsymbol{\phi}(\mathbf{x}_t)^T\boldsymbol{\phi}(\mathbf{x}_t) > 0\end{aligned} \tag{34}$$

the critical point minimizes the incremental risk functional.

Hence, the update of IRMA is given by

$$\boldsymbol{\omega}_{t+1} = (A + \frac{1}{\sigma_t}B(\mathbf{x}_t))^{-1}\left[A\boldsymbol{\omega}_t + \frac{1}{\sigma_t}\boldsymbol{\phi}(\mathbf{x}_t)y_t\right]. \tag{35}$$

With the basis $\{\phi_i(\mathbf{x})\}_{i=1}^n$ in the function space $L^2(X)$, A is the *Gramian matrix* given by the standard inner product on functions. Hence the matrix A is positive definite for linearly independent $\phi_i(\mathbf{x})$ and has an inverse A^{-1}. Choosing the substitution $\mathbf{u} = \boldsymbol{\nu} = \frac{1}{\sqrt{\sigma_t}}\boldsymbol{\phi}(\mathbf{x}_t)$, the second part of (35) can be expressed as $\frac{1}{\sigma_t}B(\mathbf{x}_t) = \mathbf{u}\boldsymbol{\nu}^T$. Thus the *Sherman–Morrison* formula yields the entire inverse[1]

$$(A + \mathbf{u}\boldsymbol{\nu}^T)^{-1} = A^{-1} - \frac{A^{-1}B(\mathbf{x}_t)A^{-1}}{\sigma_t + \boldsymbol{\phi}(\mathbf{x}_t)^T A^{-1}\boldsymbol{\phi}(\mathbf{x}_t)}. \tag{36}$$

So the minimization has a unique solution and as the matrix A is constant for any given model structure and can be inverted off-line, (36) poses a numerically cheap way to compute the inverse for each learning step [93].

1 This numerically cheap parameter update is based upon a contribution of Nils Rosemann from a collaboration in [12].

This results in the following algorithm for on-line learning by IRMA:

Listing 1: Incremental Risk Minimization Algorithm

```
Parameter: stiffness σ_t, initial parameter vector ω_0
for t = 0 to n_d − 1 do{ //i.e. for each learning step
  receive instance x_t
  predict label ŷ_t = ω_t^T φ(x_t)
  receive true label y_t
  suffer loss L(y_t, ŷ_t)
  use Sherman-Morrison to calculate (A + uv^T)^{-1}
  update parameter vector ω_{t+1} = (A + uv^T)^{-1}(Aω_t + (1/σ_t) φ(x_t) y_t)
}end
```

2.3 FORMAL ANALYSIS OF IRMA

2.3.1 *Local Convergence and Stiffness*

For a formal analysis of the approach, first the influence of IRMA on the local error is presented to show the local contraction of each learning step. If the parameter vector ω is not changed, the incremental risk functional has the value

$$R_{inc}(\omega = \omega_t) = (y_t - \omega_t^T \phi(x_t))^2 \tag{37}$$

which is equivalent to the local error of the approximation. In this case, the partial derivative

$$\frac{\partial R_{inc}(\omega = \omega_t)}{\partial \omega_i} = 2(y_t - \omega_t^T \phi(x_t))\phi_i(x_t) \tag{38}$$

shows that the gradient is zero, only if the target value y_t is already met by the approximation, as ϕ is a basis and thus at least one $\phi_i(x_t) \neq 0$. In this case, no adaptation of the parameter vector is needed to incorporate the new example (x_t, y_t). Otherwise, a change of parameters $\Delta\omega_t = \omega_{t+1} - \omega_t \neq 0$ minimizes the risk functional leading to

$$\begin{aligned} 0 &< \sigma_t \cdot \int_X ((\Delta\omega_t)^T \phi(x))^2 dx \\ &\Leftrightarrow \\ ((\omega_t + \Delta\omega_t)^T \phi(x_t) - y_t)^2 &< \sigma_t \cdot \int_X ((\Delta\omega_t)^T \phi(x))^2 dx \\ &\quad + ((\omega_t + \Delta\omega_t)^T \phi(x_t) - y_t)^2 \\ &< (\omega_t^T \phi(x_t) - y_t)^2 \,. \end{aligned} \tag{39}$$

Consequently, by learning from an example (x_t, y_t) the local error on that example with the new parameter vector ω_{t+1} is less than before with ω_t. Thus learning converges locally with respect to the example.

The amount of local convergence depends on the stiffness σ_t. For $\sigma_t \to \infty$, a learning step as shown in (35), results in

$$\boldsymbol{\omega}_{t+1} = A^{-1} A \boldsymbol{\omega}_t = \boldsymbol{\omega}_t \tag{40}$$

and hence keeps the old parameter vector and the local error is the same afterwards. On the other hand, for $\sigma_t \to 0$ (35) takes the form

$$\begin{aligned} B(\mathbf{x}_t)\boldsymbol{\omega}_{t+1} &= \boldsymbol{\phi}(\mathbf{x}_t) y_t \\ &\Leftrightarrow \\ \sum_{i=1}^{n} \phi_j(\mathbf{x}_t)\phi_i(\mathbf{x}_t)\omega_{t+1,i} &= \phi_j(\mathbf{x}_t) y_t \quad \forall\, j \in \{1,\ldots,n\} \\ &\Leftrightarrow \\ \boldsymbol{\omega}_{t+1}{}^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{x}_t) &= y_t \end{aligned} \tag{41}$$

resulting in a new parameter vector that reproduces the example $(\mathbf{x}_t, y_t)$ exactly, i. e. decreasing the local error to zero. In between, the stiffness σ_t allows to choose how much the parameter vector and consequently the functional behavior is changed to decrease the local error.

To see the influence of a new example $(\mathbf{x}_t, y_t)$ on the parameter update $\Delta\boldsymbol{\omega}_t$, (35) can be rewritten to

$$\Delta\boldsymbol{\omega}_t = \left[A^{-1} + \frac{A^{-1} B(\mathbf{x}_t) A^{-1}}{\sigma_t + \boldsymbol{\phi}(\mathbf{x}_t)^{\mathsf{T}} A^{-1} \boldsymbol{\phi}(\mathbf{x}_t)} \right] \frac{\boldsymbol{\phi}(\mathbf{x}_t)}{\sigma_t} (y_t - \boldsymbol{\omega}_t^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{x}_t)) \tag{42}$$

showing that the error on the example has a linear influence on the resulting change in parameters. The stiffness has an influence of $\frac{1}{c+\sigma_t}$ on the magnitude of a parameter update. For $\sigma_t \to 0$ the update has a defined limit

$$\lim_{\sigma_t \to 0} \Delta\boldsymbol{\omega}_t = \frac{A^{-1}}{\boldsymbol{\phi}(\mathbf{x}_t)^{\mathsf{T}} A^{-1} \boldsymbol{\phi}(\mathbf{x}_t)} \boldsymbol{\phi}(\mathbf{x}_t) \cdot (y_t - \boldsymbol{\omega}_t^{\mathsf{T}} \boldsymbol{\phi}(\mathbf{x}_t)) \tag{43}$$

which can be derived using l'Hôpital's rule. In this case, the update is similar to the update of PA shown in Table 1, except for the transformation through the matrix A^{-1}.

2.3.2 *Worst Case Minimization*

Globally the error cannot be expected to decrease with every example because a single example only gives local information at x_t. But to incorporate it properly some global change is necessary. This goes along with the *no-free-lunch theorem* for supervised learning [117, 118], stating that there are no a priori distinctions between learning algorithms regarding the off-training-set error, i. e. the predictions in the on-line case. Any learning algorithm might produce a correct lucky guess for the next prediction.

But looking at the global approximation error, for IRMA the following inequality holds:

$$\begin{aligned}
& \int_X (\omega_{t+1}^T \phi(x) - f(x))^2 dx && (44)\\
= & \int_X \left[\omega_t^T \phi(x) - f(x) + (\Delta\omega_t)^T \phi(x)\right]^2 dx && (45)\\
= & \int_X [(\omega_t^T \phi(x) - f(x))^2 + ((\Delta\omega_t)^T \phi(x))^2 \\
& +2 \cdot (\omega_t^T \phi(x) - f(x)) \cdot ((\Delta\omega_t)^T \phi(x))] dx && (46)\\
= & \int_X (\omega_t^T \phi(x) - f(x))^2 dx + \int_X ((\Delta\omega_t)^T \phi(x))^2 dx \\
& +2 \cdot \int_X (\omega_t^T \phi(x) - f(x)) \cdot ((\Delta\omega_t)^T \phi(x)) dx && (47)\\
\leqslant & \int_X (\omega_t^T \phi(x) - f(x))^2 dx + \int_X ((\Delta\omega_t)^T \phi(x))^2 dx \\
& +2 \cdot \int_X \left|\omega_t^T \phi(x) - f(x)\right| \cdot \left|(\Delta\omega_t)^T \phi(x)\right| dx && (48)\\
\leqslant & \int_X (\omega_t^T \phi(x) - f(x))^2 dx + \int_X ((\Delta\omega_t)^T \phi(x))^2 dx \\
& +2 \cdot C_m \cdot \int_X \left|(\Delta\omega_t)^T \phi(x)\right| dx && (49)\\
\leqslant & \int_X (\omega_t^T \phi(x) - f(x))^2 dx + \int_X ((\Delta\omega_t)^T \phi(x))^2 dx \\
& +2 \cdot C_m \cdot c(X) \cdot \sqrt{\int_X ((\Delta\omega_t)^T \phi(x))^2 dx} && (50)
\end{aligned}$$

Starting with the global squared error of the approximation ω_{t+1} to an unknown optimal target function $f(x)$ in (44), an expansion in (45) relates the global error at step $t+1$ to the previous error at step t. Factoring out the square in (46) and exchanging sum and integration in (47) allows to limit the global error upward in (48). If the approximation is changed in a region not supported by the current example, i. e. anywhere except at x_t, it is unknown what the true value of the target function is. A learning algorithm can only guess and depending on the unknown optimal target function $f(x)$, this is better or worse. Hence, it is not possible to be good in every case but it is possible to be safe from making it worse in every case. Thus the estimate assumes that any change from ω_t to ω_{t+1} in the worst case increases the error, i. e. the change has the same sign as the previous error of ω_t and thus is upper bounded by its absolute value.

As the absolute prior error is constant for a learning step, an upper bound can be estimated restricting it to its maximum value

$$C_m = \max_{x \in X}(\left|(\omega_t^T \phi(x) - f(x))\right|) \tag{51}$$

in (49). Additionally, as it is not always possible to minimize the absolute error and the squared error simultaneously, the absolute error

is bounded in (50) by the squared error through the *inequality of arithmetic and geometric means* with a constant $c(X)$ depending on the input space [102].

With the upper bound of (50) on the global squared error of the approximation ω_{t+1}, the first term is fixed by the previous approximation of ω_t and can thus not be influenced by any learning algorithm. The second term as well as the third term are minimized by IRMA for a given improvement on the current example (x_t, y_t) as they are equivalent to the change D of (21). Consequently, while locally decreasing the error by a certain amount depending on the stiffness (see (39)), IRMA minimizes the worst case development of the global approximation error in each step. Hence, it incorporates the new knowledge of the example as local as possible into the approximation in contrast to any other learning algorithm and is reliable in every step. Since this analysis is true for *any* model structure ϕ, IRMA is a reliable learning algorithm independent of the chosen model structure.

2.3.3 *Independence of Specific Model Structure*

With the way IRMA incorporates a new example, it gets independent of the specific formulation of the model structure. Taking different model structures that represent the same class of functions, but with different parameters to represent this class, will result in the same way of adapting the model. Other learning algorithms yield different results, as they do not respect the influence of the model structure. As IRMA incorporates knowledge about the transformation ϕ, in this case the global input-output relation is the same after adapting to an example, regardless of the specific formulation of the model structure.

An example is shown in Fig. 14 where on the one hand a GLT with linear interpolation is used as a model structure, with two parameters representing the height at -10 and 10. On the other hand, a first order polynomial is used, with two parameters representing the offset and slope. Both model structures represent the same class of functions, i. e. straight lines, on the input domain $[-10, 10]$. But, starting with a parameter vector of zero, PA yields two different results for learning on the same example whereas IRMA results in the same line in both cases. Hence, for IRMA only the input-output relation is relevant and not the inner model structure.

2.3.4 *Complexity*

With respect to the computational complexity, IRMA learning takes more effort than usual first order learning algorithms. As the matrix inverse can be obtained efficiently with the Sherman-Morrison Formula ([93], Sec. 2.7), the calculation takes $\mathcal{O}(n^2)$ steps, mainly be-

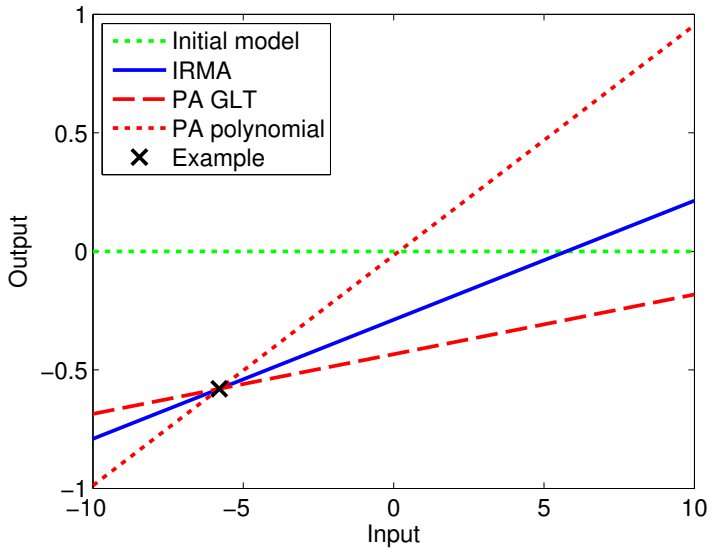


Figure 14: Comparison of the results of IRMA and PA on two model structures which represent the same class of functions. For IRMA the result is the same in both cases, whereas PA results in two different approximations.

cause of matrix multiplications. It uses $\mathcal{O}(n^2)$ amount of memory, again for storing the matrix. In comparison, state of the art first order learning algorithms like PA take $\mathcal{O}(n)$ calculation steps and $\mathcal{O}(n)$ memory. But still the complexity of IRMA is fixed depending on the complexity of the model structure with n basis functions and does not increase with increasing amounts of examples.

2.4 INVESTIGATIONS OF IRMA

2.4.1 *General Setup*

To support the principal analysis with quantifiable results and to further investigate the behavior of the presented approach, selected empirical investigations are presented in the following. Additional more specific investigations that do not fit into the focus of this chapter are presented in Appendix B. For the investigations two different model structures are used (see Appendix A.3.3 for details of the implementation). On the one hand, a GLT (see [88, Section 10.3] and [108]) with either linear or Gaussian interpolations is used. This model structure has the advantage that each parameter only influences the input-output relation locally in input space. But this local influence comes at the cost of the curse of dimensionality with respect to the number of parameters [88]. On the other hand, a polynomial [88, Section 10.2] without mixed terms, i. e. only linear combinations of different polynomial orders of single input dimensions, is used. Here the number of parameters increases only linearly with the number of dimensions.

But this comes at the cost of a complex interaction of the parameters and a global effect of each parameter on the input-output relation. Consequently, these model structures present two extremes of the design space of model structures and allow to generalize from the empirical investigations.

For evaluation of the performance of a learning algorithm, three measures are relevant. First, and most important, the cumulative loss

$$L_c(\tau) = \sum_{t=0}^{\tau-1} (y_t - \boldsymbol{\omega}_t^T \boldsymbol{\phi}(\mathbf{x}_t))^2 \tag{52}$$

on the sequence of examples $(\mathbf{x}_t, y_t)$ evaluates the predictive performance, i. e. how good the on-line learning performs on the sequence of data presented. Second, the data loss

$$L_d(\tau) = \frac{1}{\tau} \sum_{t=0}^{\tau-1} (y_t - \boldsymbol{\omega}_{\tau-1}^T \boldsymbol{\phi}(\mathbf{x}_t))^2 \tag{53}$$

evaluates the quality on all examples $(\mathbf{x}_t, y_t)$ seen up to the respective step, i. e. how well the general relationship of the examples was learned. Third, for comparison with ground truth information, additional test examples $(\widetilde{x}_i, \widetilde{y}_i)$ are directly drawn on a fine-grained regular grid covering the complete input space regardless of the density of training examples and without any disturbance. With these data the ground truth loss

$$L_g(\tau) = \frac{1}{n_g} \sum_{i=1}^{n_g} (y_i - \boldsymbol{\omega}_{\tau-1}^T \boldsymbol{\phi}(\mathbf{x}_i))^2 \tag{54}$$

evaluates the ability to generalize and cancel noise, i. e. how well the learned approximation suits undisturbed and regularly sampled examples of the optimal target. Measuring this ground truth loss is only possible for a known analytic target function.

All investigations are set up using the unified online-learning systems library (UOSLib) [8] which supports an easy comparison of learning algorithms and reproducible experiments (see Appendix A for an overview of the features). Within this framework, all experiments are uniquely described by footprints which are defined in Appendix A.

2.4.2 *Basic Empirical Investigation*

As a first example, the on-line approximation of a non-linear function is investigated, to show the behavior of IRMA throughout the learning process. A typical problem with on-line learning is the interaction of model structures with different expressiveness and a low example density, e. g. in the beginning of learning. To investigate this, for

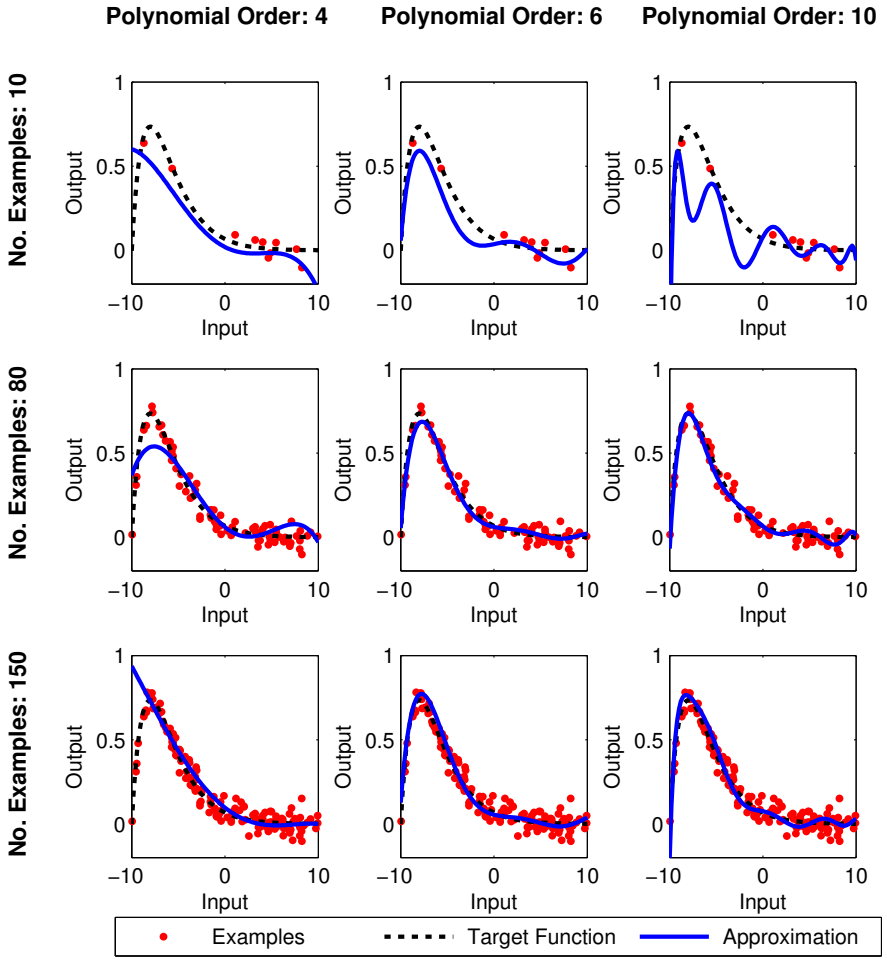


Figure 15: Resulting functional behavior of on-line learning by IRMA after different amounts of data are presented to model structures with differing complexities.

training, 150 instances x_t are drawn randomly from a uniform distribution on $[-10, 10]$. For each the target value is generated by the function $y_t = (x_t + 10) \cdot exp(-\frac{x_t+10}{2}) + \xi$ with normally distributed noise $\xi \sim \mathcal{N}(0, 0.05)$ to form an example[2]. IRMA[3] is set up with a stiffness $\sigma = 0.1$ and polynomial model structures of different orders are used[4]. With 4^{th} order, the expressiveness is too small to learn the target function properly. The 6^{th} order polynomial is appropriate for the target function, whereas the 10^{th} order normally would tend to overfitting because it is too expressive. Snapshots of the resulting functional behavior are taken after 10, 80, and 150 examples of the same sequence have been presented to see the effect of data density.

Figure 15 shows the resulting input-output relation. Additional numerical results of a ground truth comparison on 100 equally spaced,

2 UOSLib-scenario: `mode` = REG, `func` = nonlinhalf, `ND` = 150, `NG` = 100, `noise` = 0.05, `minPath` = false, `rSeed` = 1234567

3 UOSLib-learn: IRMA, `variant` = 0, `stiff` = 0.1

4 UOSLib-model: Poly, `order` = [4,6,10]

Table 2: Ground truth loss compared for on-line and batch learning with different amounts of examples on model structures with differing complexities.

No. of examples	4^{th} order	6^{th} order	10^{th} order
10 (on-line)	$9.4 \cdot 10^{-2}$	$7.8 \cdot 10^{-2}$	$1.3 \cdot 10^{-1}$
10 (batch)	$1.2 \cdot 10^{-2}$	$2.4 \cdot 10^{-1}$	$5.1 \cdot 10^{7}$
80 (on-line)	$5.6 \cdot 10^{-2}$	$1.8 \cdot 10^{-2}$	$2.1 \cdot 10^{-2}$
80 (batch)	$5.9 \cdot 10^{-3}$	$5.0 \cdot 10^{-4}$	$2.5 \cdot 10^{-4}$
150 (on-line)	$5.6 \cdot 10^{-2}$	$2.2 \cdot 10^{-2}$	$2.6 \cdot 10^{-2}$
150 (batch)	$5.5 \cdot 10^{-3}$	$3.8 \cdot 10^{-4}$	$1.7 \cdot 10^{-4}$

undisturbed test points are shown in Table 2 for the on-line learning system as well as a batch fit using the Vandermonde matrix of the complete data sequence with the Matlab® backslash operator, i. e. a least squares fit. With low data density, the higher expressiveness is used to fit to these few examples but even for the 10^{th} order polynomial which could yield values that are orders of magnitude bigger than the output range of the target function, the oscillations are only moderate and result in a low ground truth loss of $1.3 \cdot 10^{-1}$. In this case, batch learning results in a high ground truth loss of $5.1 \cdot 10^{7}$ as it is more affected by overfitting. With increasing data density, the approximation fits closely to the target function and the influence of the example noise is small. Thus a stable approximation of the target function is achieved. Though higher than for batch learning, the resulting ground truth losses of on-line learning are very low with higher data densities, but at a lower computational demand. With the low expressiveness of the 4^{th} order polynomial, the leftmost part of the target function is not met due to the low amount of data, i. e. only four of the total 150 examples.

In summary, IRMA results in a reasonable approximation in every case, regardless of the model expressiveness or the data density, without drastic influence on the approximation in areas not supported by examples, resulting in a compliant generalization without overfitting. Consequently, there are no large errors at any time during learning.

2.4.3 *Comparison of On-line Learning Methods*

To quantify the quality of learning in comparison to other on-line learning methods, a more complex analytic function with several changes of monotonicity is learned on-line. For this task, IRMA is compared with the state of the art on-line learning methods PA and RLS. Using the two model structures introduced above, the exemplary in-

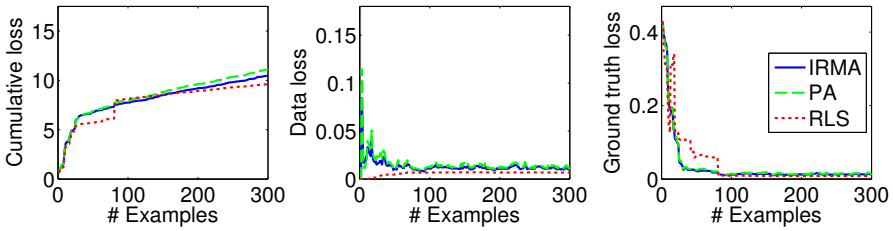


Figure 16: Comparison of IRMA, PA, and RLS on learning a sine target function without noise, using a GLT with 16 regularly distributed grid nodes.

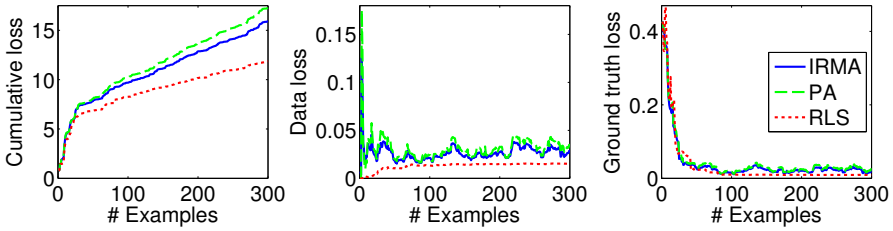


Figure 17: Comparison of IRMA, PA, and RLS on learning a sine target function with noise, using a GLT with 16 regularly distributed grid nodes.

vestigations presented here allow to generalize the results to other model structures. For training[5], $n_d = 300$ instances x_t are randomly selected from a uniform distribution on $[-10, 10]$ and the respective target value y_t is generated by a sinusoidal $y_t = \sin(x_t) + \xi$ with normally distributed noise ξ. For comparison with ground truth information, $n_g = 300$ additional equally spaced test examples are used. IRMA[6] is set up with a stiffness of $\sigma = 0.1$, PA[7] needs no hyperparameter in its basic version, and RLS[8] has an initial covariance matrix $\Sigma_0 = \mathbb{1} \cdot 10^3$ and a forgetting factor of $\lambda = 1$.

Figure 16 shows the resulting losses for a GLT model structure with Gaussian interpolation and 16 regularly distributed grid nodes[9] without noise ($\xi = 0$) and Fig. 17 with noise $\xi \sim \mathcal{N}(0, 0.1)$ on the examples. In every case, the cumulative loss increases quickly in the beginning when not enough knowledge about the target function is present. After this initialization phase, the cumulative loss increases less, depending on the amount of noise and residual approximation error. Without noise, the cumulative losses of the three algorithms are comparable and RLS is better suited in presence of noise as it has a lower slope of the cumulative loss. On the data loss, all algorithms achieve a low loss in the end which increases in presence of noise. RLS

5 UOSLib-scenario: `mode` = REG, `func` = sine, `ND` = 300, `NG` = 300, `noise` = [0, 0.1], `minPath` = false, `rSeed` = 12345

6 UOSLib-learn: IRMA, `variant` = 0, `stiff` = 0.1

7 UOSLib-learn: PA, `variant` = 0

8 UOSLib-learn: RLS, `Sinit` = 10^3, `forget` = 1

9 UOSLib-model: GLT, `num` = 16, `base` = gauss

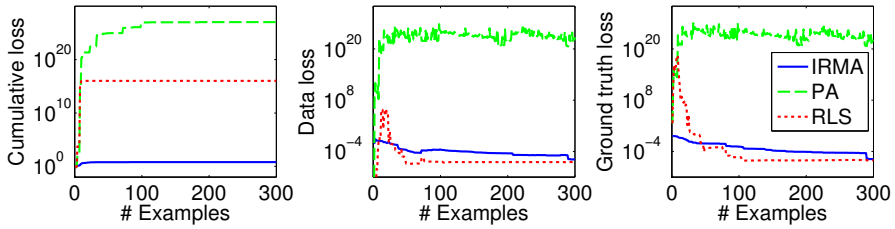


Figure 18: Comparison of IRMA, PA, and RLS on learning a sine target function without noise, using a polynomial of 15^{th} order.

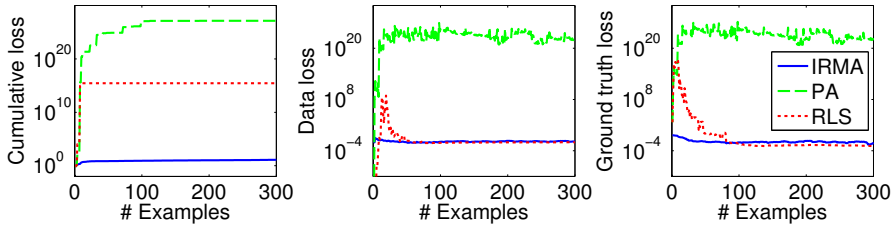


Figure 19: Comparison of IRMA, PA, and RLS on learning a sine target function with noise, using a polynomial of 15^{th} order.

is designed to minimize the data loss and thus outperforms the other methods. But as a consequence of this minimization, the ground truth loss of RLS is higher in the beginning due to overfitting. This overfitting leads to the risk of a high prediction error like the one at time step $t = 80$ in the cumulative loss. These high prediction errors are usually rare events in a learning sequence, but as it is unpredictable when a high error occurs, they are especially dangerous to the overall system behavior. Hence, the reliability of RLS is low. In comparison of the first order methods, IRMA is slightly better than PA but does not show a significantly different development.

In contrast to the local GLT model structure, Fig. 18 shows the resulting losses (mind the logarithmic scale) for a polynomial model structure of 15^{th} order[10] without noise and Fig. 19 again with noise. The polynomial model structure leads to a different behavior of the state of the art algorithms. The cumulative error of PA increases evermore and similarly the high example and ground truth losses show, that the general relationship of the examples is not learned properly. With RLS, the general relationship can be learned on the long run but the ground truth loss in the beginning is much higher with this complex model structure. This indicates overfitting and an increased risk of high prediction errors in between, therefore reducing the reliability. Thus, the cumulative error also increases much more. However, IRMA shows nearly the same results as with the GLT model structure that are orders of magnitude better than that of PA and RLS. It learns the general relationship while minimizing the risk of high prediction er-

10 UOSLib-model: Poly, `order = 15`

Table 3: Final results of the losses after all examples were presented for the comparison of IRMA, PA, and RLS on learning a sine target function. Losses $> 10^{15}$ are marked in gray.

		$L_c(n_d)$	$L_d(n_d)$	$L_g(n_d)$
IRMA	GLT	10.5	$1.0 \cdot 10^{-2}$	$1.2 \cdot 10^{-2}$
	GLT noise	15.9	$2.9 \cdot 10^{-2}$	$1.9 \cdot 10^{-2}$
	Poly	6.4	$1.3 \cdot 10^{-6}$	$1.6 \cdot 10^{-6}$
	Poly noise	12.9	$1.8 \cdot 10^{-2}$	$8.5 \cdot 10^{-3}$
PA	GLT	11.1	$1.2 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$
	GLT noise	17.3	$3.4 \cdot 10^{-2}$	$2.4 \cdot 10^{-2}$
	Poly	$1.1 \cdot 10^{27}$	$5.3 \cdot 10^{22}$	$5.6 \cdot 10^{22}$
	Poly noise	$1.4 \cdot 10^{27}$	$2.3 \cdot 10^{22}$	$2.4 \cdot 10^{22}$
RLS	GLT	9.6	$6.8 \cdot 10^{-3}$	$8.3 \cdot 10^{-3}$
	GLT noise	11.9	$1.5 \cdot 10^{-2}$	$9.2 \cdot 10^{-3}$
	Poly	$1.1 \cdot 10^{16}$	$3.3 \cdot 10^{-7}$	$8.5 \cdot 10^{-7}$
	Poly noise	$2.8 \cdot 10^{15}$	$1.0 \cdot 10^{-2}$	$1.4 \cdot 10^{-3}$

rors in each step. With additional noise, the principal behavior stays the same but all losses increase as expected.

The final losses after the complete sequence of examples was presented are shown in Table 3. The polynomial model structure allows to achieve a lower cumulative loss, if used properly. Especially for IRMA each loss decreases with this model structure, even though the expressiveness with respect to VC-dimension is the same as with the GLT model structure [107]. In contrast PA performs worse than IRMA in every case and cannot use the expressive polynomial model structure. However, RLS has low data and ground truth losses which improve as well in combination with the polynomial model structure. So it is able to learn it adequately on the long run, but its on-line performance is bad with a polynomial model structure as L_c shows.

The main advantages of IRMA shown in this investigation are twofold. On the one hand, as it minimizes the worst case error as shown in Section 2.3.2, it prevents high cumulative losses as they occur with RLS. Thus not only low data and ground truth losses are achieved but as well a low cumulative loss. On the other hand, the incorporation of knowledge about the non-linear transformation ϕ allows to learn any model structure that is linear in its parameters, even if the parameters interact globally, as demonstrated on the extreme case of a polynomial model structure.

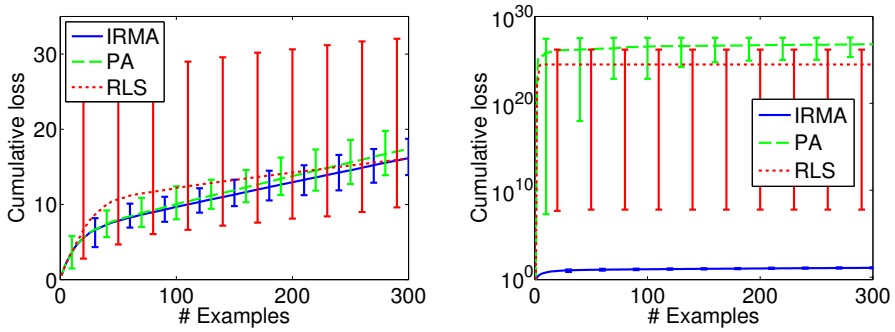


Figure 20: Comparison of IRMA, PA, and RLS on learning a sine target function with noise over 100 random sequences of examples using a GLT (left) or a polynomial (right, logarithmic scale) model structure. The average performance is shown as a line and the respective minimum and maximum as error bars.

As the chosen sequence of examples influences the results, the same experiment is repeated 100 times for different randomly drawn sequences (randomizer seeds: 12345 to 12444) to compare the variance of the results. The average cumulative loss for these sequences is shown in Fig. 20 together with its minimum and maximum as error bars. On average again all algorithms perform comparably with a GLT model structure, but RLS shows a high variance resulting in the best performance on some sequences, but also at the risk of a high loss for other sequences. With a polynomial model structure, this variability of RLS is even higher and PA even fails to learn in every case as the minimum is very high. But IRMA keeps the lowest average cumulative loss together with a low variability and achieves results comparable, albeit slightly better, to the GLT model structure. So IRMA shows a low variance of the results with both model structures, which is associated with the minimization of the worst case, and consequently provides a reliable on-line learning performance.

2.4.4 *Influence of the Stiffness*

The only hyper-parameter of IRMA is the stiffness σ, adjusting between keeping old knowledge and following a new example. Hence, to investigate the influence of the stiffness σ the previous investigation is run again with the same setup but for different values of σ. For comparison, the final cumulative loss is plotted in Fig. 21.

The results show that a single minimum for the cumulative loss, i. e. an optimal value of the stiffness, is present in every case. The polynomial model structure again results in a lower loss and especially without noise, a stiffness of $\sigma \approx 0$ is optimal, in contrast to the other cases where the optimum is different from zero. This is explained by the fact that noise on the training examples has the same influence as

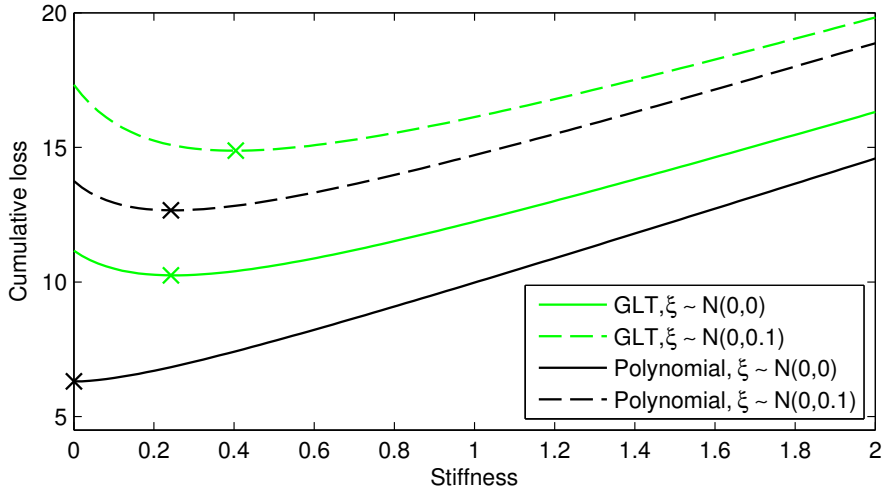


Figure 21: Influence of the stiffness on the resulting cumulative loss for different model structures and noise levels. The minimum of every case is marked by a cross.

a model structure that cannot express the optimal target function adequately. The GLT model structure has a residual error and thus even with exact examples the parameter update appears random. Thus, the optimal stiffness increases with increasing amount of noise of the examples or decreasing expressiveness. And, choosing a slightly different value for σ has only low influence on the resulting cumulative loss. Hence, the influence of the hyper-parameter σ is easy to understand and easy to adjust to a problem at hand and should be set up according to the expected noisiness, i. e. the higher the noise the higher the stiffness should be chosen.

2.4.5 *Higher Dimensional Problems*

The investigations so far dealt with one-dimensional regression problems. Thus, to investigate the results of IRMA on regression problems of higher dimensionality, the *concrete compressive strength* data set [120] obtained from the *UCI Machine Learning Repository* [16] is used. The dataset contains eight input dimensions to predict the compressive strength of concrete and consists of 1030 examples. In eight dimensions, the curse of dimensionality shows a significant disadvantage of GLT model structures. With only five grid points in each dimension, this model structure consists already of about 400,000 parameters to be learned. This is not reasonable with only 1030 examples available and the computational demand is quite high as well. But, using a polynomial model structure, the number of parameters increases only linearly, i. e. a fifth order polynomial leads to 41 parameters which is of reasonable complexity.

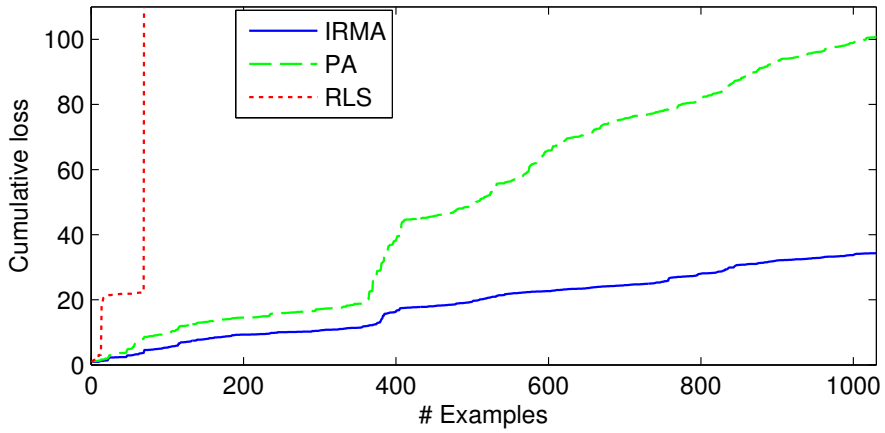


Figure 22: Comparison of IRMA, PA, and RLS on learning the higher dimensional concrete compressive strength dataset. The cumulative loss of RLS increases drastically and is thus omitted for higher values.

Table 4: Final cumulative and data loss of IRMA, PA, and RLS after learning the higher dimensional concrete compressive strength dataset.

Measure	PA	RLS	IRMA
Final cumulative loss	100.8	429.1	**34.3**
Final data loss	0.25	**0.02**	0.06

For this data set, again the three learning methods are compared. PA[11] and RLS[12] with initial covariance matrix $\Sigma_0 = \mathbb{1}$ and forgetting factor $\lambda = 1$ are compared to IRMA[13] with a stiffness of $\sigma = 2 \cdot 10^{-9}$. As the amount of functional change measured by IRMA increases with the dimensionality, this stiffness is comparable to a stiffness of $\sigma \approx 1.28$ in a single input dimension. Using a polynomial model structure of 4^{th} order[14], the results of Fig. 22 are obtained with the final losses of Table 4. The cumulative loss of IRMA increases only moderately and achieves by far the lowest result and a low data loss as well in the end. PA results in a higher cumulative and data loss, whereas RLS is able to get the lowest data loss in the end but again at a significantly higher cumulative loss. Consequently, the benefits of IRMA generalize well to higher dimensions.

In this case a GLT model structure, like any model structure with locally effective parameters, is not feasible due to the curse of dimensionality. Hence, it is especially beneficial that IRMA is able to learn polynomial model structures, or any other model structure with glob-

11 UOSLib-learn: PA, `variant` = 0
12 UOSLib-learn: RLS, `Sinit` = 1, `forget` = 1
13 UOSLib-learn: IRMA, `variant` = 0, `stiff` = $2 \cdot 10^{-9}$
14 UOSLib-model: Poly, `order` = 4

ally effective parameters, because they give a higher degree of expressiveness without being subject to the curse of dimensionality.

2.4.6 *Non-stationary Environments*

All previous investigations dealt with a fixed target function, i. e. a stationary input-output relation of the examples. But on-line learning has the advantage of a continuous adaptation to changes in this relation over time. Typically two kinds of changes are distinguished. Either a shift occurs, characterized by a sudden change, or a drift occurs, where the relation slowly changes.

For the investigation of non-stationary environments, again IRMA, PA, and RLS are compared. The GLT model structure with Gaussian interpolation and 16 regularly distributed grid nodes as well as the polynomial model structure of 15^{th} order are used again. For the shift investigation, $n_d = 300$ instances x_t are randomly selected from a uniform distribution on $[-10, 10]$ and the respective target value y_t is generated by the polynomial

$$y_t = \sum_{i=0}^{3} p_i x^i \tag{55}$$

with parameters $p = (-0.198, 0.06, 0.003, -0.0015)$ for the first 150 examples and $p = (0.198, -0.06, -0.003, 0.0015)$ for the last 150 examples[15]. For the drift investigation, the same two target polynomials are used, but the first 100 examples are drawn with the first parameter set, the last 100 examples with the second parameter set, and in between the parameter set is linearly blended[16]. To evaluate the influence of the randomly chosen sequence of examples, the experiment is run 100 times as above with different random seeds.

As the examples are not subject to noise, and the first order methods can directly deal with changes, the stiffness for IRMA[17] is set to $\sigma = 0.0$ and PA is used in its basic form[18]. RLS[19] is initialized with the covariance matrix $\Sigma_0 = \mathbb{1} \cdot 10^3$. To deal with changes, the forgetting factor is lowered to $\lambda = 0.96$ as the setting achieving the best results.

The resulting cumulative losses of the shift scenario are shown in Fig. 23. For the GLT model structure (Fig. 23 left), the performance of IRMA is nearly the same as that of PA. Both can deal with the shift at learning step 150 but the initial adaptation to the first target is quicker than relearning the second target in all three cases. RLS performs better on the first target, but consequently adapts to the second

15 UOSLib-scenario: `mode` = REG, `func` = shift, `ND` = 300, `NG` = 30, `noise` = 0.0, `minPath` = false, `rSeed` = [12345 - 12444]

16 UOSLib-scenario: `mode` = REG, `func` = drift, `ND` = 300, `NG` = 30, `noise` = 0.0, `minPath` = false, `rSeed` = [12345 - 12444]

17 UOSLib-learn: IRMA, `variant` = 0, `stiff` = 0

18 UOSLib-learn: PA, `variant` = 0

19 UOSLib-learn: RLS, `Sinit` = 10^3, `forget` = 0.96

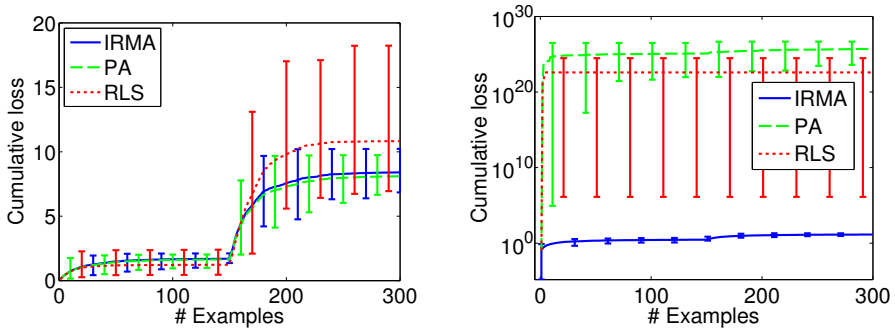


Figure 23: Comparison of IRMA, PA, and RLS in presence of a shift in the input-output relation on 100 randomly drawn example sequences using a GLT (left) or a polynomial (right, logarithmic scale) model structure. The average performance is is shown together with the respective minimum and maximum as error bars.

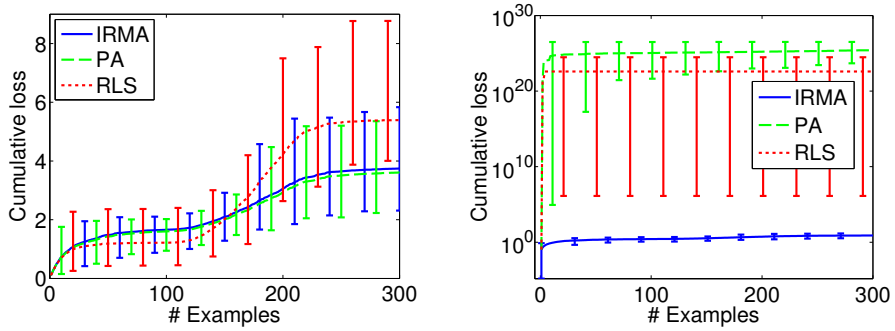


Figure 24: Comparison of IRMA, PA, and RLS in presence of a drift in the input-output relation on 100 randomly drawn example sequences using a GLT (left) or a polynomial (right, logarithmic scale) model structure. The average performance is is shown together with the respective minimum and maximum as error bars.

target more slowly and in this second phase of adaptation the performance strongly depends on the sequence of examples as the worst case is much bigger than the average. With the polynomial model structure, the known results of the more suitable learning with IRMA being orders of magnitude better can be seen as well. Especially the dependence of RLS on the sequence of examples is apparent by the high variability of its results.

The resulting losses of the drift scenario in Fig. 24 are similar to the shift scenario, but with a lower final cumulative loss. The impact on the worst case of RLS is lower in this case, but still significant.

The investigation shows that first order methods are more appropriate than second order methods to deal with non-stationary environments. Furthermore, IRMA is not only able to learn any model structure reliably, but also keeps reliable when the underlying data changes over time and learning has to adapt continuously.

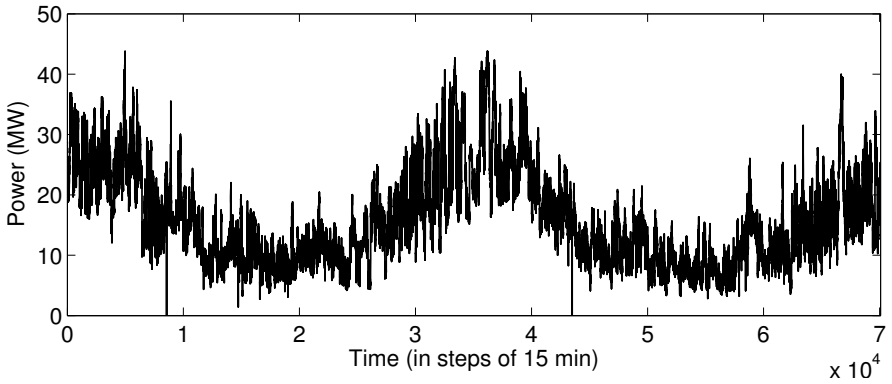


Figure 25: Power feed of the city of Kiel (Germany) on the medium voltage energy supply network level for the years 2008 and 2009.

2.5 APPLICATION TO ELECTRICITY LOAD FORECASTING

2.5.1 *Problem Domain*

Power companies rely on accurate electricity load forecasting to minimize financial risk and optimize operational efficiency and reliability. Figure 25 illustrates that such load forecasting is a typical application for continuously adapting predictive models in a non-stationary environment [15, 49, 55]. The weekly profile of electric load changes continuously, i. e. drifts, throughout a year, e. g. due to weather conditions or new small consumers or generators, on the one hand. On the other hand, rapid changes, i. e. shifts, of the conditions, e. g. due to holiday seasons or new big consumers or generators in the power grid, might occur at any time. Usually, a prediction of the next 24 hours is of interest for scheduling power plants or trading on the electricity market. The measurements as well as the predictions are done mostly on a fifteen minute basis.

2.5.2 *Investigation Setup*

The application of IRMA to load forecasting has been first published in [6]. For the investigation, data of the German city Kiel from the years 2008 and 2009 [63] is used (see Fig. 25). The recordings contain measurements of every 15 minutes for the complete two years, i. e. a total amount of 70080 examples. Based on this dataset, two scenarios are compared. First, the predictive model is required to predict the next 15 minute measurement in each step. Second, it predicts the sequence of measurements for the next 24 hours ahead, i. e. a total of 96 measurements, again in each step. Then the next measurement is used to adapt the model. In order to build a model that can learn fast and still reliably, here a rather small set of input values is chosen. De-

Table 5: Best setup of the hyper-parameter of the learning methods PA-II, RLS, and IRMA for electricity load forecasting.

	PA-II (C)	RLS (λ)	IRMA (σ)
GLT, step ahead	1.0	0.9999	0.0
GLT, 24h ahead	0.1	0.9999	2.6
Polynomial, step ahead	0.0	0.9999	0.0
Polynomial, 24h ahead	0.0	1.0	1.1

pending on the weekday and time of day as inputs the load demand is predicted as the output for different time horizons.

The investigation is done using the UOSLib as well because it simplifies the comparison of different on-line learning algorithms. As a model structure, a local GLT with Gaussian interpolation and eight nodes per dimension[20] is compared with a global polynomial structure of 7^{th} degree[21]. Both structures have a comparable expressiveness for each input dimension, i. e. eight parameters. But the GLT has a total of 64 parameters due to the curse of dimensionality whereas the polynomial has a total amount of 15 parameters. The training examples are normalized according to the UOSLib standard to have inputs in $[-10, 10]$ and the output in $[-1, 1]$. All model parameters are initialized to zero resulting in a zero prediction.

The IRMA approach is compared with PA-II as the state of the art first order learning method with adjustable aggressiveness and RLS with forgetting as the state of the art second order learning method. For each algorithm the best setup of its hyper-parameter was determined by a grid-search to achieve the lowest prediction error on the dataset (see Table 5). As a baseline comparison a naive steady prediction was performed. In this case, the last measurement is taken for the single step ahead prediction and the measurements of seven days ago, i. e. the same day of the week before at the same time, are taken to predict the next 24 hours.

2.5.3 *Results*

The resulting prediction error is measured in percent of the output range to relate the results to an accuracy of the load prediction independent of its absolute value. The initial zero prediction of both models, i. e. a medium power consumption as the output is normalized to the interval $[-1, 1]$, results in an error of 19.75% for step ahead prediction and 19.77% for the 24 hours ahead sequence. So any successful learning algorithm should at least improve beyond this performance.

20 UOSLib-model: GLT, `num` = 8, `base` = gauss

21 UOSLib-model: Poly, `order` = 7

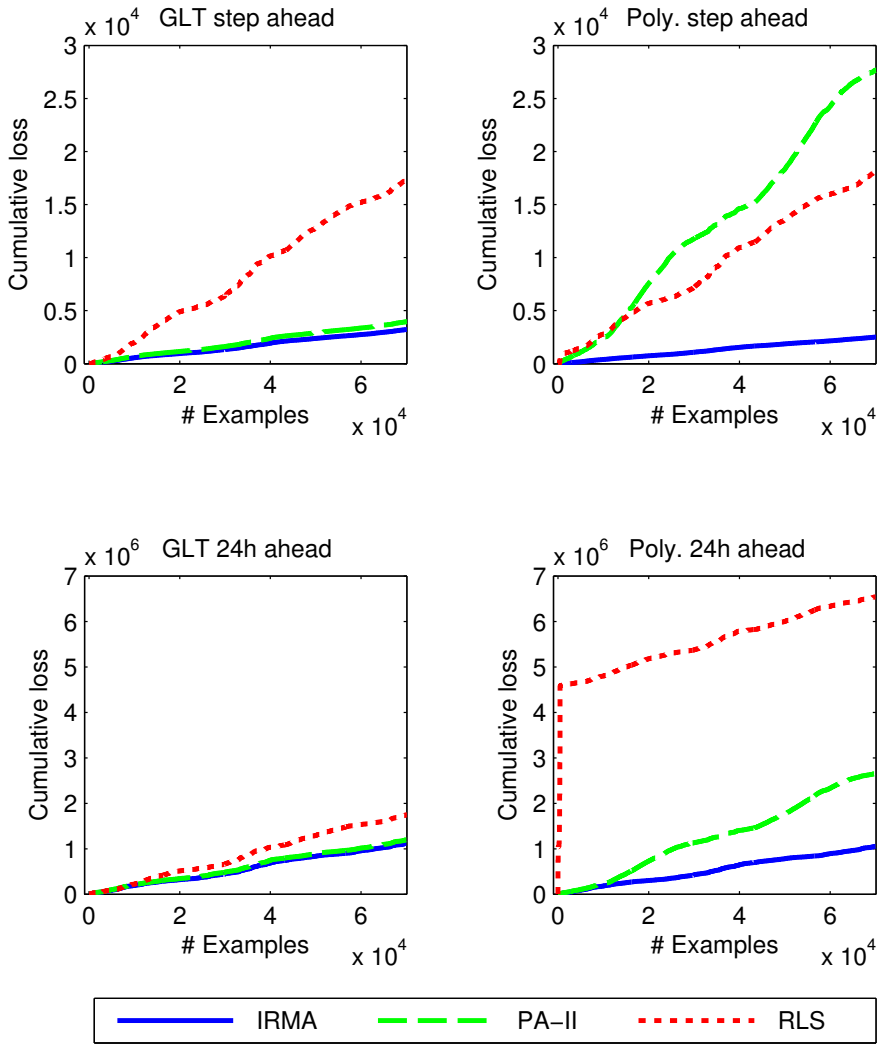


Figure 26: Comparison of the development of the cumulative loss for IRMA, PA-II, and RLS for electricity load forecasting. The left column shows the results for a GLT model structure and the right column shows results for a polynomial model structure. The upper row refers to step ahead prediction, while the lower row refers to predicting the sequence for 24 hours ahead.

Table 6: Comparison of the prediction accuracy for electricity load forecasting. The learning methods PA-II, RLS, and IRMA are compared to a steady prediction. The mean error is given in percent of the output range and best results are marked bold.

Scenario	Steady	PA-II	RLS	IRMA
GLT, step ahead	**1.04**	2.83	12.39	2.30
GLT, 24h ahead	9.66	8.92	12.97	**8.32**
Polynomial, step ahead	**1.04**	19.75	12.92	1.79
Polynomial, 24h ahead	9.66	19.77	48.69	**7.85**

Figure 26 shows the cumulative loss for the different learning algorithms over time. For step ahead prediction, IRMA and PA-II show a similar performance using the local GLT model structure, but a significant improvement is given with IRMA for the global polynomial model structure. RLS performs similarly with both model structures but yields significantly worse results. The overall relative prediction errors in percent of the output range are presented in Table 6. Using the GLT structure, IRMA achieves the best accuracy of all learning methods. But, no method is able to beat the steady prediction. The best setup here is to choose a hyper-parameter achieving a quick adaptation on each example, as it is not relevant to learn the general input-output relation, but to adapt quickly close to the next instance. This means a high aggressiveness of $C = 1.0$ for PA-II and a stiffness of $\sigma = 0.0$ for IRMA (see Table 5). For RLS the best results are achieved with a forgetting factor λ slightly below one, but it is still not able to adapt adequately to the non-stationary nature of the problem and gets the worst result. The setup of the forgetting factor is highly sensitive. Further lowering it results in very high prediction errors, decreasing the overall accuracy. Using the polynomial structure, PA-II is not able to learn at all and the best results are achieved with zero aggressiveness, i. e. no learning. However, IRMA further increases its accuracy with the polynomial model structure, again using a stiffness of $\sigma = 0.0$.

For the 24 hours ahead prediction, it is necessary to correctly learn the general input-output relation to give good long term predictions. Here, the performance in Fig. 26 with a GLT structure again is comparable for IRMA and PA-II. RLS again cannot adapt to the non-stationary data and gets instable with a lower forgetting factor. Its accuracy with a polynomial model structure even decreases for any forgetting factor below one. This shows that the stability of RLS is severely affected if forgetting is enabled at all. With a polynomial model structure the prediction of RLS suffers from severe overfitting in the beginning and has a higher slope on the long run as well. But for 24 hours ahead prediction IRMA and PA-II also outperform the steady prediction using the GLT model structure. As seen in Table 5, the optimal hyper-parameters decrease the influence of every new example by lowering the aggressiveness and accordingly increasing the stiffness. IRMA again shows a significant improvement using the global polynomial model structure achieving the lowest prediction error of all methods.

In general, the results demonstrate that RLS is not suited for non-stationary data even with forgetting as it either cannot adapt continuously or gets instable. PA-II is able to learn local model structures, but fails with the global polynomial model structure. Only IRMA is able to learn this globally active model structure adequately and achieves better results than with the GLT structure, even though much less parameters are available there. Consequently, the best prediction

is possible with a polynomial structure using IRMA as a learning method. It significantly outperforms the steady prediction even with only low input information, i. e. the daytime and weekday.

2.6 CONSEQUENCES

In a nutshell, along with the minimal local error every on-line learning approach optimizes different optimization criteria regarding the distance of the newly chosen hypothesis to the old one. While PA learning minimizes the change of the parameter vector measured by the norm of its difference, RLS learning minimizes the parameter's variance measured by its Kullback-Leibler divergence. In both cases the model structure is not considered in the parameter adaptation. Yet, a model structure is needed if the input-output relation of the presented examples is non-linear (see Section 1.1). The newly introduced IRMA minimizes the change of the global functional behavior. This measure of change of the output regarding input space and not parameter space, as done by IRMA, directly uses the information about the model structure to adjust the parameter adaptation. This way, different realizations of the same model, e. g. a first order polynomial or a GLT with two nodes, are updated the same way with a training example and not differently, as argued in Section 2.3.3.

The real-world scenario as well as the synthetic examples show that state of the art learning methods have several principal drawbacks. First order methods like PA allow a continuous adaptation on incremental data, but cannot deal with every kind of model structure and are prone to fatal forgetting. This affects especially globally active model structures, even though these structures sometimes allow to achieve better approximation results than locally active model structures. In contrast to PA, second order methods like RLS can learn such complex model structures, but only on the long run. With low data density, RLS is prone to overfitting which results in high prediction errors and consequently a high cumulative loss. Furthermore, RLS is not able to reliably adapt to continuous changes of the data as in this case there never is a "long run". With a decreased forgetting factor to achieve long term adaptability, its stability deteriorates as stated in [88] as well.

In contrast to these methods, IRMA deals reliably even with complex globally active model structures. This was demonstrated for the example of polynomial model structures. But the worst case minimization property extends this result to any model structure. IRMA preserves reliability even though it is able to incrementally adapt to non-stationary environments and results in a high prediction accuracy. The investigations on synthetic problems showed that this leads to a very different way of incorporating new examples into the parameter vector and consequently prevents overfitting and fatal forgetting

on low data densities. Hence, the advantage of IRMA is that it enforces a localized learning which does not depend on whether the basis $\boldsymbol{\phi}$ consists of localized or global functions. If the basis $\boldsymbol{\phi}$ consists of local functions that all have the same amount of influence on the global functional behavior, the result of IRMA is similar to that of PA.

Global model structures have the advantage that a higher dimensionality does not affect the number of parameters as much as with local model structures. While local model structures are not feasible in high dimensions, the example of Section 2.4.5 showed that online learning in higher dimensions is possible with IRMA using global polynomials without mixed terms. Furthermore, global model structures like polynomials generally are highly adaptive to the problem at hand, even though the model structure is fixed. At the same time, IRMA is robust against choosing an overly expressive model structure, as it systematically prevents overfitting. IRMA is proven to minimize the worst case loss in each step and is thus reliable at any time during learning while it is locally just as adaptive to new examples as other methods.

An important hyper-parameter is the stiffness of IRMA. Since it balances between quick adaptation to changes and robustness against noise, it has to be chosen depending on the problem at hand. The higher the noise level, the higher the stiffness should be set. But the more and the faster the input-output relation of the examples changes over time, the lower the stiffness should be. The investigation showed that the influence of the stiffness on the cumulative loss is smooth and has a unique minimum growing with higher noise levels. This is a consequence of the general bias-variance trade-off. So engineering this parameter is intuitively done. A more specific investigation on increasing the stiffness during learning is presented in Appendix B.2. In case of quick learning required in the beginning and if only small long term changes occur, a sigmoidally increasing stiffness is appropriate to tune this trade-off during learning. Yet, it does not have the capability to adapt to the noise level at hand in a data dependent way, like second order methods, and an ideal setup of an adequate stiffness for each step is not possible a priori. But as the power grid application shows, a fixed stiffness already yields good results. Typically, an estimation of the expected noise level and the amount of change in a non-stationary environment is possible for a given problem beforehand and the stiffness can be set up accordingly. Furthermore, the update with a zero stiffness shows a high accuracy, e. g. for the step ahead prediction in the power grid application and other investigations, again with both kinds of model structures. So even with globally active parameters learning with the highest possible adaptation is still reliable, as still the worst case is minimized.

Consequently, the performance of IRMA is not necessarily the best in every investigation presented here because any method can make

a lucky guess. But more importantly the performance does not deteriorate in any setting at all by preventing too unlucky guessing, thus making IRMA the most reliable on-line learning method. Altogether, IRMA allows on-line learning of *any* LIP model structure by respecting the influence of the non-linear transformation. The learning process is reliable in *every* step because it minimizes the worst case and thus prevents fatal forgetting. And it still has low computational and memory complexity while the resulting quality is comparable to batch learning. Hence, it is applicable for embedded systems as well as big data applications in stationary and non-stationary environments.

3 ON-LINE UNCERTAINTY ESTIMATION

In this chapter, the on-line learning setting of Chapter 2 is extended to explicitly represent uncertainties of the learning system. Thereto, the learning system is monitored to estimate its uncertainties during learning. This way, knowledge about the uncertainty of the parameter vector as well as the uncertainty of every individual prediction can be assessed.

3.1 STATE OF THE ART UNCERTAINTY ESTIMATION

3.1.1 *Basics of Uncertainty Estimation*

An on-line learning system adapts its knowledge during the ongoing operation of a system and might influence the system behavior directly again. Hence, the system safety can be affected if the resulting prediction of a learning system is incorrect. In other words, any on-line learning system can be or get uncertain and thus cannot be trusted in every situation. Knowledge about this uncertainty is a useful information for successive system modules, as it gives an estimate whether the prediction can be safely used or not. This way the total system gets more reliable and trustworthy. That is why a dynamic uncertainty estimation of an on-line learning system is necessary which covers all sources of uncertainty described in Section 1.3 at any time. It should further be easily interpretable and computationally cheap to get.

An overview of estimating the uncertainty or reliability, respectively, in machine learning is presented in [25]. The uncertainty of a prediction can be categorized into *conflict* if different labels are likely

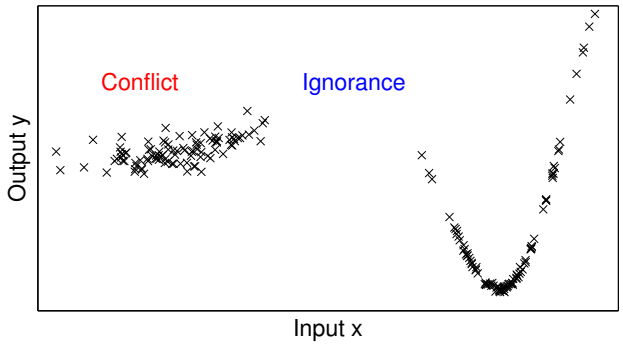


Figure 27: Illustration of the two main causes of uncertainty in a prediction, namely conflict (left) and ignorance (center).

to be true and *ignorance* if the true label is not known. These categories are illustrated in Fig. 27. If the examples are noisy like on the left side, conflict about the true label arises. But likewise in the rightmost part, the examples are subject to noise, albeit less drastically. If examples are missing in some region of the input space like in the center, ignorance about the true label arises. Again, some degree of ignorance exists wherever no example is presented like in the leftmost part. At the same time, ignorance not only arises from low example density but just as well from a too high expressiveness of the model. Likewise, conflict depends on the variance of the example's labels as well as a too low expressiveness of the model.

3.1.2 *Model Specific Batch Approaches*

The most widespread machine learning technique assessing the uncertainty of its model is the field of *Gaussian processes*, where the uncertainty is treated by normally distributed random variables [94]. Classically, Gaussian processes are learned on a batch of training examples, hence increasing the computational load with more examples, but sparse approximations have been proposed to reduce this problem [41, 89]. A Gaussian process defines a distribution over functions and inference takes place directly in function space, in contrast to second order learning algorithms like CW or AROW which also deal with a distribution over functions but in parameter space. Such a distribution is specified by a mean function and a covariance function as the prior which results in a posterior distribution conditioned by the training examples. The usual formulation with a squared-exponential covariance function allows to set parameters for input noise (signal variance), output noise (noise variance) and a length scale to adjust the algorithm [94]. But this way the uncertainties are fixed design parameters and not extracted on-line from the examples to supervise the learning process. The only data-driven uncertainty here is the scarcity of examples, hence the reflection of ignorance. Yet, Gaussian processes allow for the expression of this uncertainty for each individual prediction by a confidence interval deduced from its normal distribution.

Two uncertainty measures were introduced for the *validity index network* which learns a radial basis function network based on a set of training examples [71]. The first measure calculates the local density of training examples contributing to a particular radial basis function, giving information about its ignorance. The other measure estimates the conflict by a confidence limit of the output, given as a symmetric interval. This is derived through leave-one-out cross-validation over all examples. So conflict and ignorance can be estimated for each individual prediction, but this is not feasible for on-line estimations as the complete dataset is needed and especially cross-validation is very

time consuming. A similar extension of general multi layer perceptrons is presented in [114]. An additional output neuron is introduced which yields the variance of the prediction, i. e. the two outputs of the network are then mean and variance, similar to the output of Gaussian processes. This additional output is trained by traditional backpropagation using the local variance of a batch of training examples as its target value. Recently, this idea was generalized to the analysis of the sensitivity of a local regression model either by adding a test instance to the training set and retraining the model with perturbed labels for that instance [24], or again by leave-one-out cross-validation [23]. All of these approaches result as well in a huge computational overhead and thus are not applicable in an on-line setting.

Another approach of making uncertainties in function approximation explicit uses evidence theory, i. e. a form of imprecise probability, as a representation [44]. Here the uncertainty of a multi layer perceptron is expressed, similarly to the confidence interval of Gaussian processes, by lower and upper expectations. The width of this interval reflects the uncertainty resulting from the relative scarcity of training examples, i. e. ignorance of the learning system, but the prediction interval is not influenced by conflict. With an extension in [92] conflict of the prediction (called nonspecificity) is estimated as well. Yet, the training examples even have to be reduced to representative prototypes to accommodate the increasing computational effort with larger datasets for both approaches.

3.1.3 *Model Independent Batch Approaches*

In contrast to these methods that depend on a certain kind of model, [27] presents two model independent methods that estimate the reliability of a prediction by confidence intervals. Based on the local properties of nearby training examples, these intervals contain the true label with a certain probability. The first, called *CONFINE*, uses the mean squared error between the model and the true labels of the nearest neighbors of an evaluation instance in the training examples as an estimate of the confidence in the prediction. The higher this error is, the lower is the confidence of a prediction in that input region. The second, called *CONFIVE*, measures the variance of the labels for the same neighborhood in the training examples. Again, the higher this variance, the less confident is a prediction. Both measures reflect the uncertainty regarding conflict but not ignorance of the learned model and again depend on a full dataset.

The same holds for uncertainty representation through fuzzy instead of crisp numbers. The main application of fuzzy uncertainty representation is to transfer expert knowledge to computational methods in a fixed manner and not an on-line learning of the output fuzziness. But some work has been done on *fuzzy regression analysis* for

uncertain approximation of training data [45, 60]. These methods are again dataset based which results in a high computational demand. In addition, the uncertainty representation itself by fuzzy membership functions increases the computational demand as well.

Another recent branch of research deals with *conformal prediction* [51, 109, 110]. Using the training examples and any model that makes a point prediction, i. e. a single predicted label, conformal prediction produces a set of labels that contains the true label with some predefined probability. This allows to give growing *prediction regions* for increasing probabilities and even the influence of this probability on the growth of the set can be evaluated. It is designed for the on-line setting, but only in case of i.i.d.[1] examples. Consequently it is not suited in non-stationary environments. Additionally, it is based on the accumulated set of training examples and accordingly the computational complexity grows over time, despite the on-line setting.

3.1.4 *On-line Approaches*

All above mentioned uncertainty estimations are generated based on a complete training dataset and cannot be transfered to on-line learning with a low and fixed complexity. The problem of huge datasets is addressed only by the assumption of some data reduction to representative prototypes or sparsity in the learned model. At the same time, the uncertainty measures are not able to adapt dynamically to time variant situations with increasing and decreasing uncertainty as they always take into account the complete dataset.

A more general approach to integrate the estimation of uncertainty in each step into a learning scenario was proposed for reinforcement learning by the means of knows what it knows (KWIK) learning [73]. In this framework, a learning algorithm is allowed to opt out of predicting with "I don't know" and thus asserts its own knowledge. A KWIK algorithm for on-line linear regression was presented in [103]. It is based on least squares regression, e. g. by RLS, and measures the uncertainty of the least squares estimate due to ignorance. But again, in this framework uncertainty due to conflict is assumed to be averaged out by learning and thus not reflected anymore.

The only approach to on-line uncertainty estimation reflecting conflict and ignorance is applied for evolving Takagi-Sugeno fuzzy systems [79]. It produces local confidence levels taking into account confidence deterioration due to extrapolation and interpolation of the local models which form the consequence part of the fuzzy system. Therefor, the covariance matrix of an RLS learning algorithm is used to estimate the example density in combination with a global variance estimation based on a small regularly updated batch of data to add

1 Independent and identically distributed, i. e. each example is drawn from the same probability distribution as the others and all examples are mutually independent.

local error bars to the output. But the variance estimate poses a computational overhead and has the drawback that a global variance is used to estimate local uncertainties. Recently, this work was extended to on-line conflict and ignorance estimation but only for classification [78]. Yet, this approach requires learning by RLS and is not applicable for any other on-line learning algorithm.

3.1.5 *Consequences*

No approach is available to estimate the uncertainty of a general LIP model structure, independent of the learning algorithm used, that satisfies all requirements. Several approaches are based on one specific model structure or do not represent all sources of uncertainty and hence cannot form a general approach to the problem of uncertain function approximation. With most approaches, the computational complexity is too high or even increases with the amount of examples and thus cannot be applied in an on-line learning system. The on-line capable approaches either deal only with ignorance or depend on RLS as a learning algorithm. But still, the overview of uncertainty estimations shows that the core idea of all approaches is based on at least one of the two principles of conflict and ignorance.

3.2 TRUSTED PARAMETERS APPROACH

3.2.1 *General Approach*

An on-line estimation of a learning system's uncertainties is necessary to assess the reliability of each individual prediction at all times. As discussed before, information about the *conflict* and *ignorance* of the learning system allow to observe the uncertainty. The approach presented here is based on [10, 11] and estimates these uncertainties by monitoring the parameter vector and assigning a trustworthiness to each parameter, on-line dependent on the learning process. This way uncertainty source 4 presented in Section 1.3 is the central point to monitor all other sources of uncertainty in a learning system.

Two general principles allow to observe these uncertainties. On the one hand, the *variability* of a parameter over time shows how uncertain a parameter is regarding its conflict. The sources 1 – 3 of uncertainty affect the variability of a parameter. Input noise, output noise, or unobserved variables all result in examples that contain different labels for the same instance. Hence, the parameters are adapted all the time to incorporate these different labels. In addition, a model structure that is not expressive enough cannot incorporate all examples adequately at the same time. So here as well the parameters are adapted continuously. These influences can in principle be detected through the variability of the parameter vector, but it is important

to note that the particular cause cannot be distinguished, i. e. it is known that the prediction is uncertain but not why it is uncertain. On the other hand, the *density* of training examples gives information about the amount of knowledge substantiating a parameter. In case of a low number of presented examples, or a model structure that is too expressive, the data density per parameter is low. Thus the sources of uncertainty 1 and 3 are covered monitoring the density.

Especially for on-line learning, a probabilistic uncertainty treatment, i. e. by a statistical analysis of the examples, is inappropriate, as only one new training example is available at a time. Additionally, the example density varies significantly within the input space in practical applications and ignorance resulting from low density cannot be represented properly by probability [59, 116]. Hence, the trust management approach is more suited in this case and is computationally cheap at the same time. Based on the variability- and density-principle, three measures for a general LIP model structure are proposed here to give meta-information to every learned parameter.

3.2.2 *Ignorance Uncertainty Estimation*

First, to estimate the density of training examples for a certain parameter, its activity $\hat{\phi}_i$ for each presented example is summed up. The activity of a parameter is given by its relative contribution to the output

$$\hat{\phi}_i(\mathbf{x}) = \frac{|\phi_i(\mathbf{x})|}{\sum_{j=1}^{n} |\phi_j(\mathbf{x})|}. \tag{56}$$

Thus, for each instance $\mathbf{x}_t$ a total activity of $\sum_{i=1}^{n} \hat{\phi}_i = 1$ is distributed to the different parameters, reflecting the responsibility of a parameter for the respective prediction. With this activity $\hat{\phi}_i$, the ignorance measure Φ is updated on-line with each example's instance $\mathbf{x}_t$ to be

$$\Phi_i(T) = \sum_{t=0}^{T-1} \hat{\phi}_i(\mathbf{x}_t) \tag{57}$$

after the sequence of T examples was presented. This measure Φ increases throughout learning and the bigger its value, the more examples substantiate the parameter's setup and the less uncertainty due to ignorance about the parameter is present. Accordingly, to map this ignorance measure Φ to a trust level ϑ, a monotonically increasing function should be used. Two selected variants of this mapping are proposed. Either, a linear mapping

$$\vartheta^{I_1}(\Phi, \delta_t^\Phi, \delta_s^\Phi) = \begin{cases} 0 & \text{if } \Phi \leqslant \delta_t^\Phi \\ 1 & \text{if } \Phi \geqslant \delta_s^\Phi \\ \frac{\Phi - \delta_t^\Phi}{\delta_s^\Phi - \delta_t^\Phi} & \text{otherwise} \end{cases} \tag{58}$$

with a minimal $\delta_t^\Phi \geqslant 0$ and maximal $\delta_s^\Phi \geqslant \delta_t^\Phi$ amount of summed activity is used. This method needs two hyper-parameters to be set by the designer. But their influence is self-evident. The first (δ_t^Φ) selects how much activity is necessary to get any trust at all and the second (δ_s^Φ) selects how much activity suffices to fully trust the parameter. This way, the trust estimation ϑ^{I_l} is sensitive to the ignorance measure Φ only in this defined region and otherwise is not influenced.

As an alternative with sensitivity for any increasing Φ, a hyperbolic mapping

$$\vartheta^{I_h}(\Phi, \eta_I) = \frac{\eta_I \Phi}{1 + \eta_I \Phi} \tag{59}$$

with a single hyper-parameter $\eta_I > 0$, selecting how quickly the trust increases with the summed activity, is used. This way, a smooth dependence of the trust ϑ^{I_h} on the activity is achieved with less parameterization but full trustworthiness is only achieved in the limit.

3.2.3 *Conflict Uncertainty Estimation*

Second, the variability is estimated based on the average absolute adjustment of a parameter per activity throughout the learning process. The conflict measure $\bar{\Delta}$ is updated on-line with each example (x_t, y_t) to be

$$\bar{\Delta}_i(T) = \frac{1}{\Phi_i(T)} \sum_{t=0}^{T-1} |\Delta\omega_i(x_t, y_t)| \tag{60}$$

after the sequence of T examples was presented with the parameter adjustment $\Delta\omega_i(x_t, y_t) = \omega_{t+1,i} - \omega_{t,i}$. Thus the average amount of applied absolute adjustments for every parameter is monitored. The higher this average adjustment is, the more fluctuating is the information about the parameter and the more uncertainty due to conflict about the parameter is present. Accordingly, to map this conflict measure $\bar{\Delta}$ to a trust level ϑ, a monotonically decreasing function should be used. As the division in (60) is not defined in case of no training examples, i. e. $\Phi = 0$, the trust is then defined as zero. Two variants of the trust mapping analogous to (58) and (59) are proposed. Either, a linear mapping

$$\vartheta^{C_l}(\bar{\Delta}, \delta_t^{\bar{\Delta}}, \delta_s^{\bar{\Delta}}) = \begin{cases} 1 & \text{if } \bar{\Delta} \leqslant \delta_t^{\bar{\Delta}} \\ 0 & \text{if } \bar{\Delta} \geqslant \delta_s^{\bar{\Delta}} \\ \frac{\delta_s^{\bar{\Delta}} - \bar{\Delta}}{\delta_s^{\bar{\Delta}} - \delta_t^{\bar{\Delta}}} & \text{otherwise} \end{cases} \tag{61}$$

with a minimal $\delta_t^{\bar{\Delta}} \geqslant 0$ and maximal $\delta_s^{\bar{\Delta}} \geqslant \delta_t^{\bar{\Delta}}$ amount of average adjustment is used. Here again, the influence of these hyper-parameters is self-evident, i. e. some variability that is tolerated and an upper

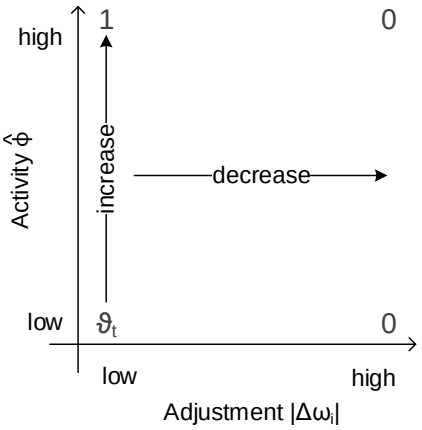


Figure 28: Principle of the direct incremental trust estimation of a parameter based on its activity $\hat{\phi}_i(\mathbf{x}_t)$ and adjustment $|\Delta\omega_i|$. The four extreme cases of the resulting new trust ϑ_{t+1} are given in the corners and in between the trust is monotone.

limit of variability that is not trustworthy at all. Again the trust ϑ^{C_l} is only sensitive within this predefined range. Alternatively, a hyperbolic mapping

$$\vartheta^{C_h}(\bar{\Delta}, \eta_C) = \frac{1}{1 + \eta_C \bar{\Delta}} \tag{62}$$

with a single hyper-parameter $\eta_C > 0$ selecting how quickly the trust decreases with average adjustment is used. This way, again a smooth dependence of the trust ϑ^{C_h} on the activity is achieved with less parameterization.

3.2.4 *Incremental Uncertainty Estimation*

Both measures Φ and $\bar{\Delta}$ take the whole sequence of examples into account incrementally. But, in a non-stationary environment, a direct trust estimation with short term memory is more appropriate as the underlying data changes over time and a region learned well before might be uncertain later. Therefor, the activity of a parameter and its adjustment through the learning algorithm in each learning step are considered to update its trustworthiness with each presented example. Three boundary cases as shown in Fig. 28 describe the behavior of this direct incremental estimation. A high activity $\hat{\phi}_i(\mathbf{x}_t)$ with low necessary adjustment $|\Delta\omega_i|$ (upper left corner) leads to full trust $\vartheta_{t+1} = 1$ as the parameter ω_i is verified as being correct, i. e. it was highly responsible for the respective prediction and no change was necessary. Whereas with a low activity and low adjustment (lower left corner) the trust is not changed ($\vartheta_{t+1} = \vartheta_t$) as the training example gives no information about the parameter's correctness, i. e. other parameters were responsible for the respective prediction. Contrariwise, if the parameter adjustment $|\Delta\omega_i|$ is high (right side), the trust is re-

duced regardless of the activity as the parameter ω_i is not confirmed yet. In between these cases, the trust is blended monotonically.

Two plausible variants of such a blending are again a linear and a hyperbolic update of the trust level. The linear update of the trust level $\vartheta^{D_l}_{t,i}$ is given by

$$\vartheta^{D_l}_{t+1,i} = \hat{\Delta}_i \cdot \left[\hat{\phi}_i(\mathbf{x}_t) + \vartheta^{D_l}_{t,i} \cdot \left(1 - \hat{\phi}_i(\mathbf{x}_t)\right)\right] \tag{63}$$

with the normalized parameter adjustment

$$\hat{\Delta}_i(\Delta\omega_i(\mathbf{x}_t, y_t), \delta^D_t, \delta^D_s) = \begin{cases} 1 & \text{if } |\Delta\omega_i(\mathbf{x}_t, y_t)| \leqslant \delta^D_t \\ 0 & \text{if } |\Delta\omega_i(\mathbf{x}_t, y_t)| \geqslant \delta^D_s \\ \frac{\delta^D_s - |\Delta\omega_i(\mathbf{x}_t, y_t)|}{\delta^D_s - \delta^D_t} & \text{otherwise} \end{cases} \tag{64}$$

parameterized by a minimal amount of adjustment $\delta^D_t \geqslant 0$ tolerated, i. e. it should not affect the trustworthiness, and a maximal amount of adjustment $\delta^D_s \geqslant \delta^D_t$ allowed, i. e. it cannot be trusted beyond, thus defining low and high adjustment in Fig. 28. Alternatively, the hyperbolic update of the trust level $\vartheta^{D_h}_{t,i}$ is given by

$$\vartheta^{D_h}_{t+1,i} = \frac{1}{1 + \eta_A \, |\Delta\omega_i(\mathbf{x}_t, y_t)|} \cdot \left(1 - \frac{1 - \vartheta^{D_h}_{t,i}}{1 + \hat{\phi}_i(\mathbf{x}_t)}\right) \tag{65}$$

with one hyper-parameter $\eta_A > 0$ weighting the influence of the parameter adjustment $|\Delta\omega_i|$.

Both approaches to a direct incremental estimate of the uncertainty result in a combined monitoring of ignorance and conflict as the activity and adjustment reflect the two uncertainty categories. A parameter can only gain trust, if it has been highly active for some training example, i. e. if there is no ignorance, and if the necessary adjustment was low, i. e. if there was no conflict.

3.2.5 *Combination*

In order to rate the total uncertainty of the on-line learning system, trust management is used to combine the different presented trust signals, regarding long term and short term aspects of ignorance and conflict, to one single trust estimation $\vartheta^\star$. The two trust signals of conflict ϑ^C and ignorance ϑ^I yield partly redundant information about the long term uncertainty of a parameter[2]. For the fusion of such partly redundant information, a compensatory operator like the average is used. In contrast, the direct estimate ϑ^D reflects the combined

2 The procedure is the same for the linear and hyperbolic variants. Thus the respective indices are omitted.

short term uncertainty of a parameter regarding conflict and ignorance which is non-redundant to the long term estimates. For the fusion of non-redundant information, a t-norm like the minimum is used. Hence, to calculate the combined trust signal a fusion according to

$$\vartheta_i^\star = \min\left\{\vartheta_i^D \ , \ \frac{\vartheta_i^C + \vartheta_i^I}{2}\right\} \tag{66}$$

is done. Any other combination of a compensatory operator and a t-norm would be possible. But the advantage of the average is that its result it is neither optimistic nor pessimistic, and any other t-norm than the minimum has the disadvantage that it quickly decreases to zero, i. e. it is pessimistic.

Accordingly, if the short term estimate assigns a low trust, the combined trust is low regardless of the long term performance, thus fitting to the influence of the most recent update of the parameter vector. Contrariwise, if the short term trust is high just because of an example that fitted to the parameter vector by chance, the combined trust is dominated by the long term estimate. Yet, the importance of short and long term estimation might vary depending on the task. For tasks in non-stationary environments with high time-variance, using only the direct estimate is more appropriate. Otherwise, without time-variance, using only the combined long term estimates suffices.

Using this monitoring of the parameter vector trustworthiness, it is possible to assess the trustworthiness of each individual prediction. Therefor, each parameter's trust influences the trust of a prediction according to its responsibility for the predicted value. This means, the more active a parameter is, the more influence its trust level has on the resulting trustworthiness. Each of the above parameter based trust estimates, i. e. ϑ^I, ϑ^C, ϑ^D, or $\vartheta^\star$, can be combined according to

$$\vartheta_{\widehat{y}_t}(\mathbf{x}_t) = \sum_{i=1}^{n} \hat{\phi}_i(\mathbf{x}_t)\vartheta_{t,i} \tag{67}$$

to give the prediction trust $\vartheta_{\widehat{y}_t}$. This way, it is possible to evaluate the total uncertainty of each prediction by using $\vartheta^\star$ or to distinguish the reason of uncertainty into ignorance and conflict on long and short term by evaluating only the respective trust signal.

3.3 FORMAL ANALYSIS OF TRUSTED PARAMETERS

3.3.1 *Influence of Hyper-Parameters*

The three measures regarding ignorance ϑ^I, conflict ϑ^C and a direct combination ϑ^D are derived from presenting a sequence of examples that are subject to some disturbance. The most significant properties of the examples for the trust estimation are given by the amount of

examples presented and the noisiness of the presented labels. Looking at the influence of the hyper-parameters on this estimation, for the linear long term variants of the trust estimation, engineering the hyper-parameters is straight forward. They directly define lower and upper bounds on the amount of examples needed or the amount of noise tolerated. The setting of the hyper-parameters of the hyperbolic variant is not as obvious. Therefore, an analysis of the expected value of the respective trust signal depending on the properties of the presented examples helps to engineer the hyper-parameters.

HYPERBOLIC IGNORANCE The hyperbolic ignorance measure ϑ^{I_h} depends on the amount of activity $\Phi_i(T)$ of a parameter ω_i by all presented examples up to time T. With no activity, the resulting trust is

$$\vartheta_i^{I_h}(\Phi_i = 0) = 0. \tag{68}$$

With increasing activity the trust is strictly increasing and its limit is given by l'Hôpital's rule as

$$\lim_{\Phi_i \to \infty} \vartheta_i^{I_h}(\Phi_i) = \frac{\eta_I}{\eta_I} = 1. \tag{69}$$

Assuming that throughout the sequence of examples the activity is a random variable $\hat{\phi}_i(\mathbf{x}_t) \sim \mathcal{D}$ drawn from some fixed distribution $\mathcal{D}$ with expected value $E[\mathcal{D}] = c_{\mathcal{D}}$, the expected value of the sum of activities after a sequence of T examples is

$$E\left[\sum_{t=0}^{T-1} \hat{\phi}_i(\mathbf{x}_t)\right] = \sum_{t=0}^{T-1} E\left[\hat{\phi}_i(\mathbf{x}_t)\right] = c_{\mathcal{D}} T. \tag{70}$$

This results in a trust of

$$\vartheta_i^{I_h} = \frac{\eta_I c_{\mathcal{D}} T}{1 + \eta_I c_{\mathcal{D}} T} \tag{71}$$

and consequently to get an ignorance based trust $\vartheta_i^{I_h} = 0.5$ after τ presented examples drawn from $\mathcal{D}$, the hyper-parameter should be chosen as

$$\eta_I = \frac{1}{c_{\mathcal{D}} \tau}. \tag{72}$$

So, the hyper-parameter η_I allows to steer the growth of the trustworthiness with the amount of examples. It increases more rapid, i. e. a higher trust with a lower amount of examples, the bigger the hyper-parameter η_I is chosen.

HYPERBOLIC CONFLICT The hyperbolic conflict measure ϑ^{C_h} depends on the variability $\bar{\Delta}_i(T)$ of a parameter ω_i resulting from all

presented examples up to time T. With no variability, the resulting trust is

$$\vartheta_i^{C_h}(\bar{\Delta}_i = 0) = 1. \tag{73}$$

With increasing variability the trust is strictly decreasing and its limit is

$$\lim_{\bar{\Delta}_i \to \infty} \vartheta_i^{C_h}(\bar{\Delta}_i) = 0. \tag{74}$$

Assuming again that the activity is a random variable $\hat{\phi}_i(\mathbf{x}_t) \sim \mathcal{D}$ drawn from some fixed distribution $\mathcal{D}$, its expected value is given by (70). Further assuming the adaptation to be a normally distributed random variable $\Delta_i(\mathbf{x}_t) \sim \mathcal{N}(0, \sigma^2)$ throughout the sequence of examples due to noise on the labels with variance σ^2, the expected value of the sum of absolute adaptations after a sequence of T examples is

$$\mathrm{E}\left[\sum_{i=0}^{T-1} |\Delta_i(\mathbf{x}_t)|\right] = \sum_{i=0}^{T-1} \mathrm{E}\left[|\Delta_i(\mathbf{x}_t)|\right] = T\sigma^2\sqrt{\frac{2}{\pi}}. \tag{75}$$

This results in a total expected value of the variability measure of

$$\mathrm{E}\left[\bar{\Delta}_i\right] = \frac{1}{c_{\mathcal{D}} T} T\sigma^2\sqrt{\frac{2}{\pi}} = \sigma^2\sqrt{\frac{2}{\pi c_{\mathcal{D}}^2}} \tag{76}$$

which is independent of the number of examples presented. Thus, to get a conflict based trust $\vartheta_i^{C_h} = 0.5$ for examples with variance σ^2, the hyper-parameter should be chosen as

$$\eta_C = \frac{1}{\sigma^2\sqrt{\frac{2}{\pi c_{\mathcal{D}}^2}}} \tag{77}$$

So, the hyper-parameter η_C allows to steer the decline of the stiffness with increasing noise. It declines more rapid, i. e. a smaller trust with a higher noisiness, the bigger the hyper-parameter is chosen.

3.3.2 *Fixed Point of Direct Estimation*

HYPERBOLIC DIRECT ESTIMATE The hyperbolic variant of the direct measure $\vartheta_t^{D_h}$ is defined recursively. So the limit of this sequence of recursive updates for a fixed activity $\hat{\phi}$ and a fixed absolute adaptation $|\Delta|$ is of interest to analyze the steady state behavior of the estimation. The fixed point $\breve{\vartheta}^{D_h}$ of the sequence is given through (65) by

$$\breve{\vartheta}^{D_h} = \frac{1}{1 + \eta_A |\Delta|}\left[1 - \frac{1 - \breve{\vartheta}^{D_h}}{1 + \hat{\phi}}\right] \tag{78}$$

resulting in

$$\breve{\vartheta}^{D_h} = \frac{1}{\frac{\eta_A|\Delta|}{\hat{\phi}} + \eta_A|\Delta| + 1}. \tag{79}$$

Abbreviating $\alpha = \frac{1}{1+\eta_A|\Delta|}$ and $\beta = 1 + \hat{\phi}$, the following holds:

$$\begin{aligned} \vartheta_{t+1}^{D_h} = \alpha - \frac{\alpha}{\beta} + \frac{\alpha}{\beta}\vartheta_t^{D_h} \geqslant \vartheta_t^{D_h} \quad &\Leftrightarrow \\ \left(1 - \frac{\alpha}{\beta}\right)\vartheta_t^{D_h} \leqslant \alpha - \frac{\alpha}{\beta} \quad &\Leftrightarrow \\ \vartheta_t^{D_h} \leqslant \frac{\alpha - \frac{\alpha}{\beta}}{1 - \frac{\alpha}{\beta}} = \breve{\vartheta}^{D_h} \end{aligned} \tag{80}$$

So, if the current value $\vartheta_t^{D_h}$ is less or equal to the fixed point $\breve{\vartheta}^{D_h}$, the sequence is monotonically increasing. Furthermore

$$\begin{aligned} \vartheta_{t+1}^{D_h} = \alpha - \frac{\alpha}{\beta} + \frac{\alpha}{\beta}\vartheta_t^{D_h} \leqslant \breve{\vartheta}^{D_h} = \frac{\alpha - \frac{\alpha}{\beta}}{1 - \frac{\alpha}{\beta}} \quad &\Leftrightarrow \\ \vartheta_t^{D_h} \leqslant \frac{\beta}{\alpha}\frac{\alpha - \frac{\alpha}{\beta}}{1 - \frac{\alpha}{\beta}} + 1 - \beta = \frac{\alpha - \frac{\alpha}{\beta}}{1 - \frac{\alpha}{\beta}} = \breve{\vartheta}^{D_h} \end{aligned} \tag{81}$$

shows that the new value $\vartheta_{t+1}^{D_h}$ is less or equal to the fixed point $\breve{\vartheta}^{D_h}$ if the current value $\vartheta_t^{D_h}$ was so. The same holds vice versa with inverted inequalities if the current value $\vartheta_t^{D_h}$ is above the fixed point. Consequently, the sequence converges to its fixed point in every case.

LINEAR DIRECT ESTIMATE The linear variant of the direct measure $\vartheta_t^{D_l}$ is defined as well recursively. So again its limit for a fixed activity $\hat{\phi}$ and a fixed normalized absolute adaptation $\hat{\Delta}$ is of interest to analyze the steady state behavior of the estimation. The fixed point $\breve{\vartheta}^{D_l}$ of the sequence is given through (63) by

$$\breve{\vartheta}^{D_l} = \hat{\Delta}\left(\hat{\phi} - \breve{\vartheta}^{D_l}\left(1 - \hat{\phi}\right)\right) \tag{82}$$

resulting in

$$\breve{\vartheta}^{D_l} = \frac{\hat{\Delta}\hat{\phi}}{1 - \hat{\Delta} + \hat{\Delta}\hat{\phi}}. \tag{83}$$

Using this fixed point, the following argumentation similar to the hyperbolic direct measure holds:

$$\begin{aligned} \vartheta_{t+1}^{D_l} = \hat{\Delta}\hat{\phi} + \vartheta_t^{D_l}\left(\hat{\Delta} - \hat{\Delta}\hat{\phi}\right) \geqslant \vartheta_t^{D_l} \quad &\Leftrightarrow \\ \hat{\Delta}\hat{\phi} \geqslant \vartheta_t^{D_l}\left(1 - \hat{\Delta} + \hat{\Delta}\hat{\phi}\right) \quad &\Leftrightarrow \\ \vartheta_t^{D_l} \leqslant \frac{\hat{\Delta}\hat{\phi}}{1 - \hat{\Delta} + \hat{\Delta}\hat{\phi}} = \breve{\vartheta}^{D_l} \end{aligned} \tag{84}$$

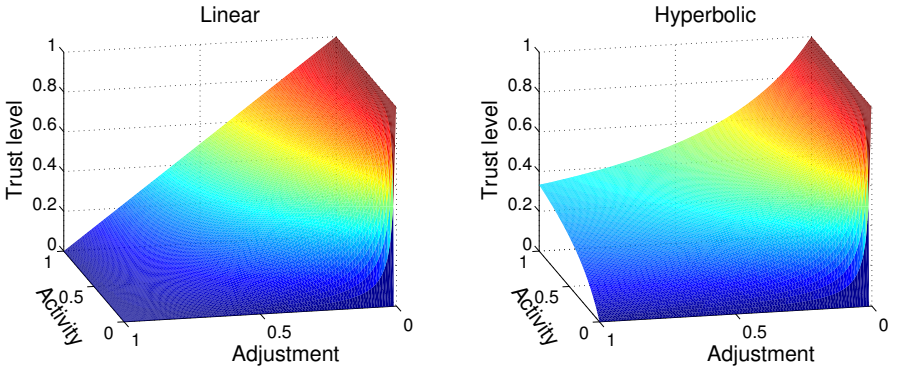


Figure 29: Limit of the direct incremental trust estimation using the linear recursive update (left) and the hyperbolic recursive update (right).

So, if the current value $\vartheta_t^{D_\iota}$ is less or equal to the fixed point $\check{\vartheta}^{D_\iota}$, the sequence is monotonically increasing. Furthermore

$$\vartheta_{t+1}^{D_\iota} = \hat{\Delta}\hat{\phi} + \vartheta_t^{D_\iota}\left(\hat{\Delta} - \hat{\Delta}\hat{\phi}\right) \leqslant \check{\vartheta}^{D_\iota} = \frac{\hat{\Delta}\hat{\phi}}{1-\hat{\Delta}+\hat{\Delta}\hat{\phi}} \quad \Leftrightarrow$$

$$\vartheta_t^{D_\iota} \leqslant \left[\frac{\hat{\Delta}\hat{\phi}}{1-\hat{\Delta}+\hat{\Delta}\hat{\phi}} - \hat{\Delta}\hat{\phi}\right]\frac{1}{\hat{\Delta}-\hat{\Delta}\hat{\phi}} = \frac{\hat{\Delta}\hat{\phi}}{1-\hat{\Delta}+\hat{\Delta}\hat{\phi}} = \check{\vartheta}^{D_\iota} \tag{85}$$

shows that the new value $\vartheta_{t+1}^{D_\iota}$ is less or equal to the fixed point $\check{\vartheta}^{D_\iota}$ if the current value $\vartheta_t^{D_\iota}$ was so. The same holds again vice versa with inverted inequalities if the current value $\vartheta_t^{D_\iota}$ is above the fixed point. Consequently, the sequence converges as well to its fixed point in every case.

The fixed points thus reflect the steady state behavior of the direct measures and show the influence of the hyper-parameters. For the hyperbolic direct measure η_A reflects the magnitude with which an adjustment of the parameter decreases the trust in (79) (similar to the conflict measure). For the linear direct measure the sensitivity to adjustments in (83) is bounded by the hyper-parameters the same way as with the linear conflict measure. The resulting fixed points of the two direct estimation variants are compared in Fig. 29. As expected, in both cases the influence of activity and adjustment on the trust estimation is monotone. An activity of zero leads to a trust of zero, independent of the adjustment. And the trust increases with increasing activity. Likewise, a zero adjustment leads to a trust of one, independent of the activity. Here it is important to note that the fixed point represents only the steady state behavior, but the convergence to this fixed point is much slower with less activity. The main difference is the clearly restricted range of adjustments where any trust is assigned at all for the linear variant which is defined by the hyper-parameters. Compared to that, the hyperbolic variant decreases to no trust only in the limit of infinite adjustment. Typically, the adjustments that occur in a real application are bounded. So the hyperbolic trust will never

decrease to zero and consequently is less expressive as it cannot use the full range of trust levels.

3.4 INVESTIGATIONS OF TRUSTED PARAMETERS

3.4.1 *Basic Empirical Investigation*

The trust estimation of a learned model should correctly reflect erroneous predictions due to conflicting information about the setup of a parameter as well as due to ignorance about the parameter's value. To demonstrate the typical behavior of the proposed method, two examples will be examined. First, an analytic function is learned on-line, using a GLT model structure[3] and then using an 8^{th} order polynomial[4]. The GLT model structure is set to have different expressiveness throughout the input range with nodes at

$$x = (-10, -5, -2, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)\,. \tag{86}$$

For training with IRMA[5], $n_d = 400$ instances x_t are randomly selected from a normal distribution $\mathcal{N}(\pi, 4)$ to get locally varying example densities. The respective target value y_t is generated by a sinusoidal function

$$y_t = \sin(x_t) + \xi$$

with normally distributed noise $\xi \sim \mathcal{N}(0, 0.1)$[6].

The parameterization of the linear ignorance trust is set to $\delta_t^{\Phi} = 0.2$, $\delta_s^{\Phi} = 10$ to require at least 10 examples for each parameter to be fully trusted. Equivalently the hyperbolic ignorance trust is parameterized with $\eta_I = 0.4$, i. e. assuming a uniformly distributed activity the trust rises to 0.5 after about 5 examples. The linear conflict trust is parameterized with $\delta_t^{\bar{\Delta}} = 0.05$ and $dsens^{\bar{\Delta}} = 0.8$ just like the linear direct estimate with $\delta_t^{D} = 0.05$ and $\delta_s^{D} = 0.8$ to tolerate very low noise and to not trust any noise above 40% of the output range. The hyperbolic conflict trust is similarly parameterized with $\eta_C = 1.5$ and the hyperbolic direct estimate the same with $\eta_A = 1.5$, i. e. it rises to 0.5 at a noise of 20% of the output range.

The resulting approximation and its corresponding trust estimation are shown in Fig. 30 for the GLT model structure. The benefit of this structure is the locality of its parameters, thus allowing for a reasonable local trust estimation as well. The conflict trust is high in the area with dense examples and a fine grained model (around $x \approx \pi$) and decreases with the coarse grid and lower example density. The ignorance trust is comparable but slightly higher in the coarse grained

3 UOSLib-model: GLT, `loc` = $(-10, -5, -2, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$, `base` = gauss

4 UOSLib-model: Poly, `order` = 8

5 UOSLib-learn: IRMA, `variant` = 0, `stiff` = 0.1

6 UOSLib-scenario: `mode` = REG, `func` = sineloc, `ND` = 400, `NG` = 100, `noise` = 0.1, `minPath` = false, `rSeed` = 12345

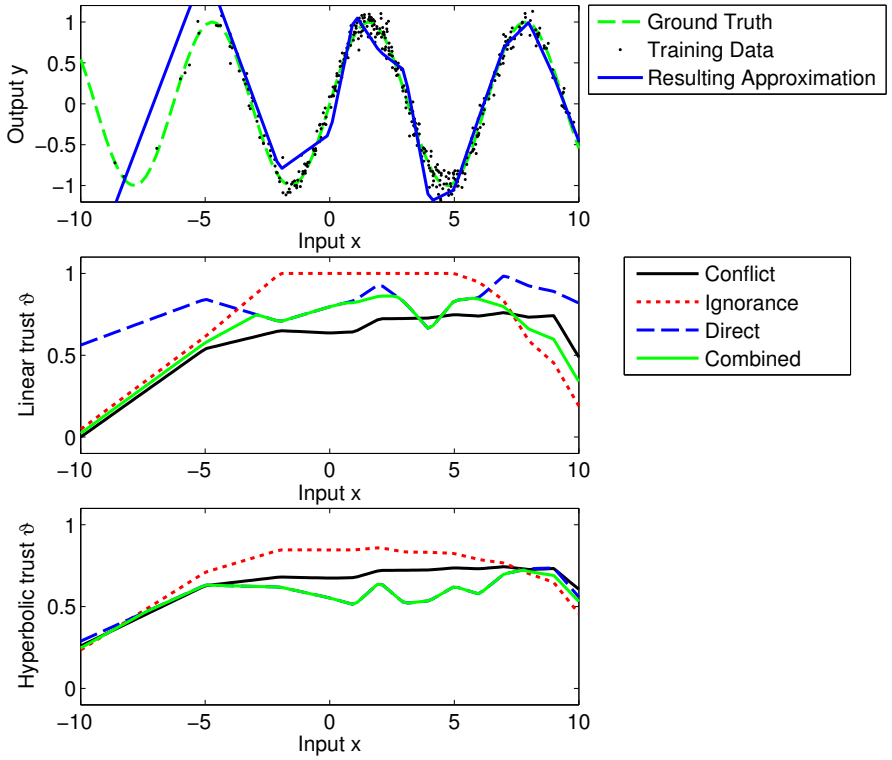


Figure 30: Resulting approximation of a GLT model structure after learning (top) and the corresponding trust estimation for the linear variant (middle) and the hyperbolic variant (bottom).

part with high example density (around $x \in [-2, 0]$) and results in full trust with dense examples regardless of the noise. The direct estimate dominates the combined trust with a slightly lower trust, e. g. at $x = -2$ and $x = 4$, where recent changes were necessary, as it is a short-term estimate. But it also results in a high trust if the last example was met quite well, e. g. around $x = -8$ even if the resulting approximation is not ideal. But the combined trust consequently reflects the total uncertainty very well. The main difference between the hyperbolic and linear variant is that a wider trust range is used by the linear mapping, as it can be set up through two parameters and thus allows to specify a narrower range of sensitivity. And, the hyperbolic estimates result in a lower trust as they are more pessimistic than the linear estimates.

In comparison, the results for the polynomial model structure are shown in Fig. 31 with the same parameterization. One big difference is that the trust estimate is symmetric due to the symmetric nature of the absolute influence of the polynomial's parameters. So the high example density on the right half increases the trust on the left half as well. Furthermore, the responsibility of each parameter varies a lot throughout the input space, resulting in much more activation of the higher order monomials. Thus the trust increases quickly to one

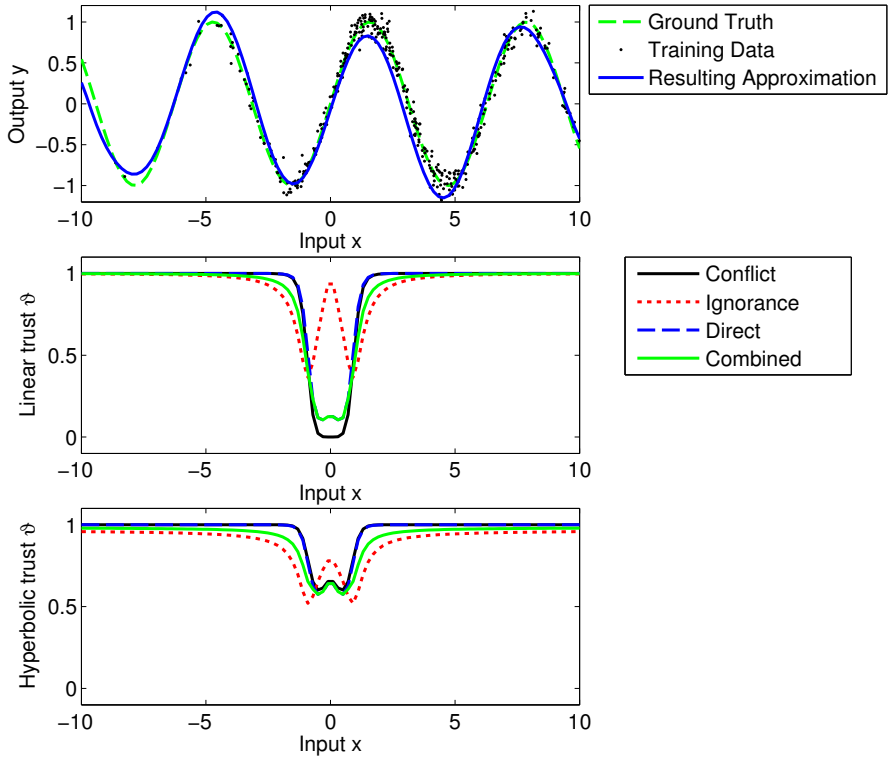


Figure 31: Resulting approximation of a polynomial model structure after learning (top) and the corresponding trust estimation for the linear variant (middle) and the hyperbolic variant (bottom).

apart from $x \approx 0$. Consequently, an appropriate trust estimate cannot be given locally with this global model structure and therefore it is less informative. So, even though the trust estimate is applicable to all kinds of LIP model structures and reflects the trustworthiness of the parameters to some degree, it quickly loses its local interpretation with more globally active model parameters. Hence, a local trust estimation necessitates a local model structure.

3.4.2 *Influence of Disturbances*

To demonstrate and investigate the different measures in a more elaborate way, the same target function is learned by IRMA using a linear GLT model structure now with 17 equally spaced nodes[7] covering the input range adequately. $n_d = 300$ instances are chosen uniformly distributed for learning to present sufficient examples throughout the input space. This way, the basic setting allows a good learning result and the influence of added disturbances can be evaluated. In the beginning of learning only sparse data are available and convergence properties can be evaluated in the end. Six scenarios, representing the first three uncertainty categories of Section 1.3, are employed for

7 UOSLib-model: GLT, `num` = 17, `base` = linear

investigation, namely the ideal case with no disturbances, four types of disturbances, and all four disturbances at once. These four disturbances are:

- Type (i): An input disturbance is emulated by a normally distributed noise $\mathcal{N}_I(0, 0.1)$ which is added to the instance given to the learning algorithm (uncertainty category 1).
- Type (ii): An output disturbance is simulated similarly by an additive normal distribution $\mathcal{N}_O(0, 0.1)$ to the target label given to the learning algorithm (uncertainty category 2).
- Type (iii): The influence of an unobserved variable is done by an additive disturbance of $0.5y^2$ to the label with random $y \sim \mathcal{U}(0, 1)$, which is not known by the learning algorithm (uncertainty category 3).
- Type (iv): Lastly, an inexact approximation is simulated by using a GLT with only seven nodes (uncertainty category 3).

All these disturbances influence the conflict of the learning system. Ignorance is covered by the varying density of training examples throughout the progress of learning as the density increases with the number of incrementally presented examples. For the evaluation, every scenario is run 20 times (randomizer seeds: 12345 to 12364) for the linear and the hyperbolic variant in order to eliminate the influence of the randomness of the examples. After every presented example, the model is evaluated at $n_t = 100$ equally distributed undisturbed test points (x_j, y_j) and compared to the target function by a weighted mean squared error e_w (87). The MSE is weighted by the respective trust, thus considering the knowledge about potential errors. Hence a low trust, does not increase e_w as opposed to the MSE. As still for a low e_w the trust should be as high as possible, additionally the mean trust $\overline{\vartheta}$ of the model is evaluated by (88). Thus for $\overline{\vartheta} = 1$, e_w is equal to the MSE.

$$e_w = \frac{1}{n_t} \sum_{j=0}^{n_t-1} \vartheta(x_j) \cdot (f(x_j) - \hat{y}_j))^2 \tag{87}$$

$$\overline{\vartheta} = \frac{1}{n_t} \sum_{j=0}^{n_t-1} \vartheta(x_j) \tag{88}$$

The trust estimation is parameterized the same way as before, i. e. $\delta_t^\Phi = 0.2$, $\delta_s^\Phi = 10$, $\eta_I = 0.4$, $\delta_t^{\bar{\Lambda}} = 0.05$, $\delta_s^{\bar{\Lambda}} = 0.8$, $\delta_t^D = 0.05$, $\delta_s^D = 0.8$, $\eta_C = 1.5$, $\eta_A = 1.5$.

The plots of all results are presented in Appendix B.3 for completeness. But, as the course of e_w and $\overline{\vartheta}$ over the number of examples for every scenario and trust estimation shows the same qualitative behavior, the results are here reduced to their characteristic properties for

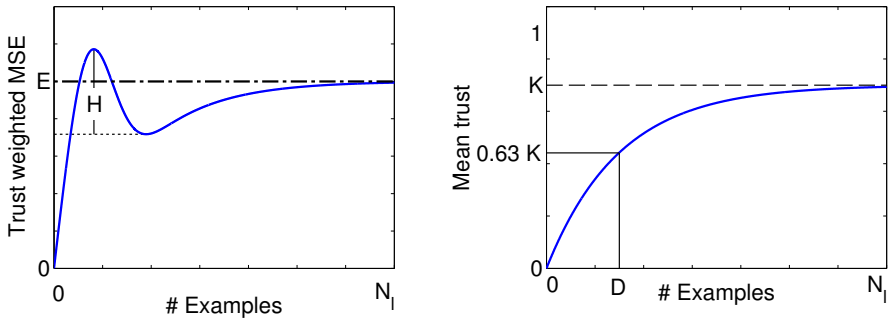


Figure 32: Qualitative development of trust weighted error (left) and mean trust (right). Characteristic numbers for these curves are shown only principally (see text for details).

ease of analysis. Figure 32 shows the principal behavior of the two measures. Because of learning from scratch, the weighted error e_w (Fig. 32 left) always starts at zero as the trust is initialized with zero. Then it increases and converges on the long run to a characteristic value E. As not every measure is able to reflect the low density of training examples in the beginning, e_w possibly overshoots indicated by the hight H which is zero in case of no overshoot. The mean trust $\overline{\vartheta}$ of the model (Fig. 32 right) starts at zero as well and converges monotonously to a boundary value K. An overall example density D, i. e. the necessary number of incremental learning steps to reach 63% of the boundary value K, characterizes the convergence speed.

Table 7 summarizes the characteristic numeric values to compare the performances in a condensed way. As a reference, the learning system principally is able to achieve a low residual error E in the undisturbed case. Accordingly, all trust levels converge to a high value of K as the learning proceeds and thus reflect the successful learning.

As a result, the conflict trust $\vartheta^{\bar{\Delta}}$ decreases for every disturbance and has a boundary value $K < 1$ of the mean trust according to the severity of its influence, e. g. the lowest trust is given for all disturbances at once. Hence, it reflects the uncertainty about the target value due to conflicts very well on the long run. But, in all scenarios the conflict trust $\vartheta^{\bar{\Delta}}$ gets the highest overshoots H of the weighted error, i. e. it is not able to recognize the ignorance due to low example density.

Expectedly, the ignorance trust ϑ^{Φ} gets low or no overshoots H, indicating that the example density is correctly recognized. On the long run, the linear ignorance trust ϑ^{Φ} converges to a mean trust of $K \approx 1.0$ and the hyperbolic variant to a high value of $K \approx 0.86$ despite the disturbances as it cannot reflect the remaining variability. Thus, it gets the highest error E.

The direct trust estimate ϑ^{D} partly reflects the density of training examples. As the model has to be adapted more in the initial learning phase depending on the difference between the target values and the initial value of the model, this lowers the estimated trust, but cannot

Table 7: Experimental results of the trust estimation for different disturbances (index h for the hyperbolic mapping variant and l for the linear mapping variant).

	Uncertainty	Linear estimate				Hyperbolic estimate			
	measure	E_l	H_l	K_l	D_l	E_h	H_h	K_h	D_h
Undisturbed	Conflict	0.04	0.03	0.82	76.00	0.04	0.05	0.79	36.00
	Ignorance	0.05	0.00	0.98	104.00	0.04	0.01	0.85	55.00
	Direct	0.05	0.02	0.98	34.00	0.04	0.01	0.82	42.00
	Combined	0.04	0.01	0.90	91.00	0.04	0.02	0.80	47.00
Disturbed input	Conflict	0.05	0.02	0.73	74.00	0.05	0.04	0.73	29.00
	Ignorance	0.07	0.00	0.98	104.00	0.06	0.01	0.85	55.00
	Direct	0.07	0.01	0.92	32.00	0.05	0.01	0.72	36.00
	Combined	0.06	0.00	0.84	90.00	0.05	0.01	0.71	39.00
Disturbed output	Conflict	0.06	0.02	0.67	75.00	0.06	0.04	0.69	27.00
	Ignorance	0.09	0.00	0.98	104.00	0.07	0.00	0.85	55.00
	Direct	0.07	0.01	0.87	31.00	0.06	0.01	0.66	32.00
	Combined	0.07	0.00	0.80	90.00	0.06	0.01	0.66	36.00
Unobserved variables	Conflict	0.10	0.02	0.56	76.00	0.12	0.02	0.64	22.00
	Ignorance	0.18	0.00	0.98	104.00	0.16	0.00	0.85	55.00
	Direct	0.14	0.00	0.78	29.00	0.10	0.00	0.58	29.00
	Combined	0.13	0.00	0.72	88.00	0.10	0.00	0.58	31.00
Inexact approx.	Conflict	0.11	0.05	0.50	36.00	0.14	0.05	0.62	7.00
	Ignorance	0.24	0.00	1.00	56.00	0.22	0.00	0.92	34.00
	Direct	0.16	0.01	0.72	13.00	0.12	0.00	0.55	13.00
	Combined	0.15	0.00	0.67	46.00	0.12	0.00	0.55	15.00
All disturb.	Conflict	0.09	0.04	0.32	34.00	0.17	0.03	0.55	6.00
	Ignorance	0.31	0.00	1.00	56.00	0.29	0.01	0.92	34.00
	Direct	0.18	0.02	0.61	13.00	0.14	0.01	0.47	12.00
	Combined	0.17	0.00	0.56	46.00	0.14	0.01	0.47	12.00

reflect the density directly. Hence, it results in low overshoots H and as well reflects the severity of disturbances by a lowered trust $K < 1.0$. As this measure is incrementally estimated, it gets to its boundary value the fastest as seen by the low values of D.

Summing up, the ignorance trust ϑ^{Φ} reflects the long-term local density very well, whereas the conflict trust $\vartheta^{\bar{\Delta}}$ reflects the long-term variability. The direct trust estimate ϑ^{D} allows a combined, yet less accurate, short-term estimation for both effects with quick adaptation. Consequently, the combined trust estimate $\vartheta^{\star}$ covers all cases and gives an overall informative measure of the local trustworthiness of the model. It results in nearly no overshoot H, a low trust weighted error E, and a final trust K reflecting the severity of the disturbances. In comparison between the linear and hyperbolic variant, again the main difference is that the resulting trusts of the hyperbolic variant are in a narrower range and are overall more pessimistic.

3.4.3 *Application to Electricity Load Forecasting*

As a real-world application the electricity load forecasting of Chapter 2.5 is extended with trust estimation. The previous investigations showed that the most suitable model structure for trust estimation is a GLT structure[8]. Hence, this investigation focuses on this case using the same optimal hyper-parameters of IRMA as before, i. e. either a stiffness of $\sigma = 0.0$ for the step ahead prediction or a stiffness of $\sigma = 2.6$ for the 24h ahead sequence prediction according to Table 5.

The trust estimation is done using the linear variant, as it results in a more comprehensible setup and allows a wider spread of the resulting trust levels with principally similar behavior. For step ahead prediction, the ignorance estimation is parameterized by $\delta_t^{\Phi} = 0.2$, $\delta_s^{\Phi} = 25$. This way, the ignorance trust rises to one after about half a month of examples is presented because in this time $24 \cdot 4 \cdot 16 = 1536$ examples are presented which is about 25 per parameter. The conflict trust tolerates some low long-term noise by parameterization $\delta_t^{\bar{\Delta}} = 0.02$, $\delta_s^{\bar{\Delta}} = 0.3$, whereas the direct estimate is more sensitive to low short-term noise with $\delta_t^{D} = 0.0$, $\delta_s^{D} = 0.3$.

In consequence, using the combined trust estimate of (66) for the trust evaluation of (67), every prediction is assessed with a trust level. One possibility to deal with this trust level is to discard every prediction below a certain trust threshold similar to KWIK learning. In these cases for example the steady prediction could be used as a fallback. So measuring the prediction error only for the remaining predictions with a trust level above the threshold, the resulting accuracy can be adjusted by this trust threshold.

Figure 33 shows the result of this experiment. In the upper plot, the step ahead prediction error of trusted predictions is plotted de-

8 UOSLib-model: GLT, `num` = 8, `base` = gauss

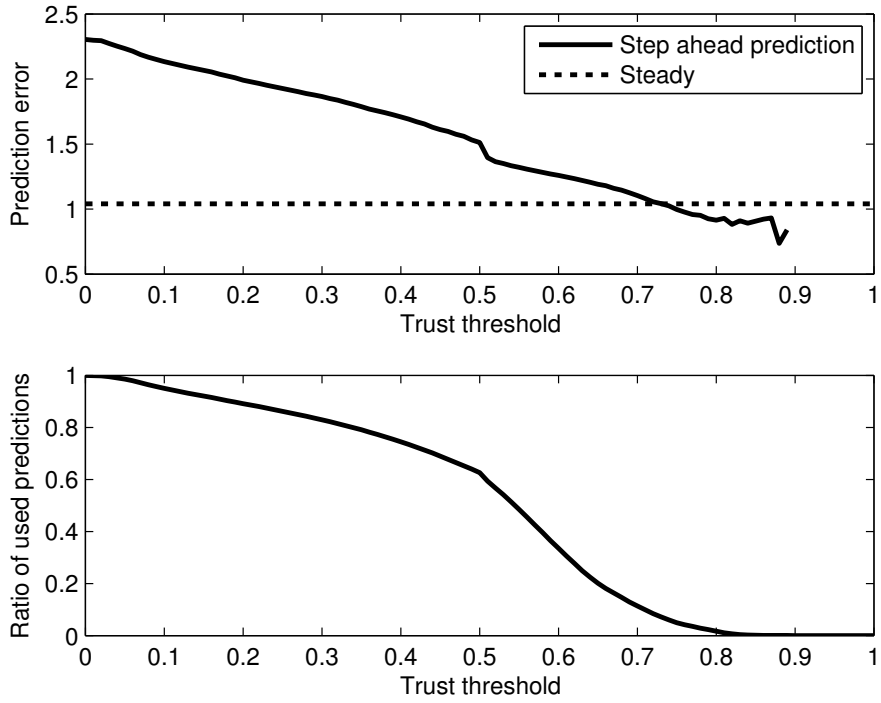


Figure 33: Improvement of the step ahead prediction accuracy by rejection of predictions below a given trust threshold (top) and the respective remaining ratio of predictions made (bottom).

pending on the threshold. The error decreases monotonically with an increasing trust threshold. Only with a very high threshold the influence is not monotone due to the small sample size influencing the averaged prediction error. Thus, as expected, the trust correlates with the accuracy of the predictions and a higher threshold adequately discards the more uncertain predictions. Additionally, with a high enough threshold of about 0.74, the accuracy exceeds that of the steady prediction. But in this case more than 92% of the predictions are rejected. So in most cases, the learned model cannot give better predictions about the load development than a steady prediction.

Still, this investigation shows that the learned model successfully knows what it does not know, hence preventing erroneous predictions. And with the supplementary trust estimation even the single step ahead prediction can be improved to outperform the steady prediction baseline. The correct threshold can be adapted dynamically on-line as well. After the true label of a prediction is revealed, it is possible to evaluate if the steady prediction was better than the learned one. In combination with the prediction trust, the threshold can be chosen to include only predictions which enhanced the accuracy in the past.

Looking at the 24h ahead sequence prediction of the electric load, the ignorance estimation is parameterized the same way as for step ahead prediction by $\delta_t^\Phi = 0.2$, $\delta_s^\Phi = 25$. The conflict trust still toler-

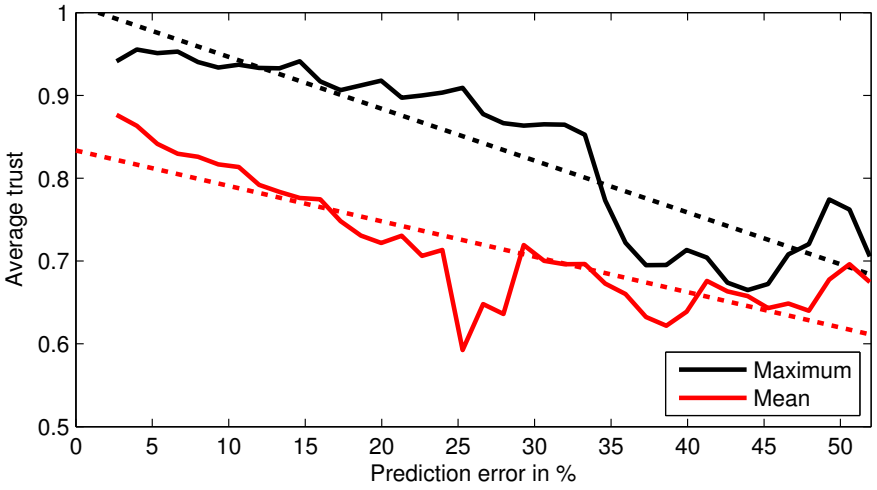


Figure 34: Mean and maximum average trust assigned to 24h ahead sequence predictions depending on their respective prediction error. A linear fit (dotted) shows the corresponding trend.

ates some low long-term noise but is more sensitive to lower noise levels due to the increased stiffness in this setup by parameterization $\delta_t^{\bar{\Delta}} = 0.02,\ \delta_s^{\bar{\Delta}} = 0.1$. The direct estimate again does not tolerate low short-term noise and is as well more sensitive to lower noise levels with $\delta_t^{D} = 0.0,\ \delta_s^{D} = 0.1$. As a result of the trust estimation, every sequence can be assigned with an average trustworthiness of its single predictions.

To investigate the effectiveness of this trust estimation, the prediction error of the total sequence is divided into 40 equally spaced bins. For each bin, the maximum and average trust assigned to the respective predicted sequences is calculated. So this trust should decrease with increasing prediction error if the trust is meaningful.

The results of the investigation are shown in Fig. 34 together with a linear fit of the data. Both curves are roughly decreasing as expected. So the more inaccurate predicted sequences are assigned with a lower trustworthiness and its upper bound by the maximum also decreases. Yet, the mean trust underestimates the predictive quality for a medium prediction error. Therefore, Fig. 35 shows the results partitioned into the first and second year of the investigation period. In the first year, the prediction accuracy is underestimated mainly due to ignorance. So even as the predictive quality is sufficient, the model is not sure about this estimate yet. In contrast, the results of the second year show a clear nearly linear tendency of the assigned trust to the respective prediction error. Furthermore, no errors above 39% occur at all in the second year.

With the additional trust information, e. g. the decision for buying additional energy on the energy market or powering up additional power stations is enhanced. Either the prediction about future loads

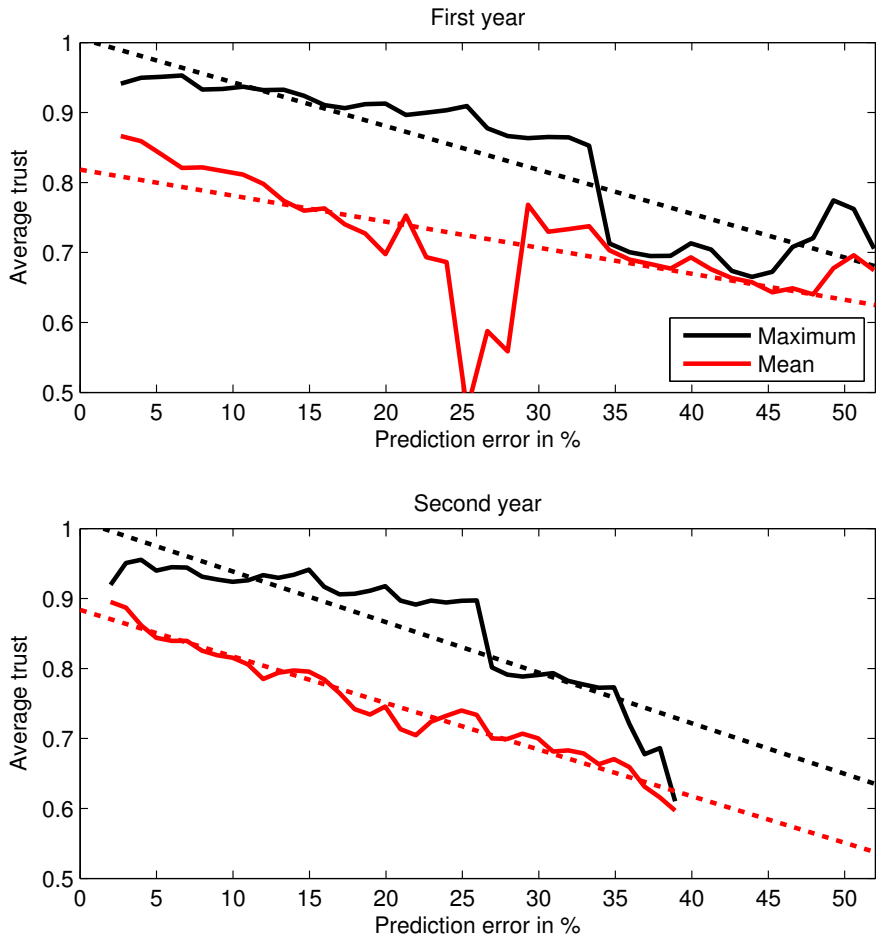


Figure 35: Partition of the mean and maximum trust assigned to 24h ahead sequence predictions depending on their respective prediction error into the two years of the investigation period. The first year is shown in the upper plot and the second year in the lower plot. A linear fit (dotted) shows the corresponding trend.

made by the learning system can be trusted and allows an optimized decision, or the prediction is not trustworthy and a safer, less optimal decision can be made. In conclusion, the real-world application shows as an example that the trust estimation is appropriate and serves as an adequate reflection of the predictive quality. This way it allows to reject predictions with a too low trustworthiness and a safe fallback strategy can be used to ensure the overall system's safety with an increased performance at the same time.

3.5 CONSEQUENCES

Using a learning system can be enhanced by monitoring its uncertainty and thus providing an uncertainty estimate for each individual prediction as well as for the system in total. The hypothesis in form of the parameter vector is central to estimating this uncertainty as it

is affected by the learning process as well as the setup of the set of hypotheses, i. e. the model structure. The approach presented here assigns each parameter of the parameter vector a trustworthiness. Both, uncertainty due to ignorance and due to conflict are monitored and represented in a unified manner through trust signals. In addition to the long term estimation of uncertainty which is more robust to singular lucky guessing of a correct prediction, a short term incremental estimate reflects the most recent uncertainty of an on-line learning system. The combination of these different trust signals through trust management yields a single uncertainty estimation for each individual prediction subsuming all kinds of uncertainty regardless of their source. So the complexity for other system modules reacting to the uncertainty of the learning system can be kept low, i. e. they get to know that the learning system's prediction can or cannot be trusted but not why. So they cannot and do not have to diagnose the source of uncertainty but know when to use a more safe fallback.

The formal analysis of the trust estimations showed how to choose the hyper-parameters. The linear estimation of uncertainty is based on a lower and an upper limit defining the extreme cases of no or full trust. This way the designer of an on-line learning system can set for the ignorance based trust how much training examples are necessary to gain any trust and how much are sufficient to fully trust the result. For the conflict based trust, the setup of the hyper-parameters defines how much variance can be tolerated and how much variance is too much to trust the result at all. The direct incremental estimate of uncertainty is set up similarly to rate the necessary adjustment of a parameter. Using the smooth hyperbolic trust estimation, only one parameter adjusts the in- or decrease rate of the trust regarding the summed activity or the average adjustment of a parameter, respectively. But despite its smoothness, this restriction limits the expressiveness. Hence, the linear variant turned out to be the more adequate choice.

As the interpretability of each parameter's influence is better for a model structure with local influence of the parameters, it is also more suited to assign a local trustworthiness to each individual prediction. This was also demonstrated in the basic empirical investigations. Otherwise, the trust estimate can only be as local in input space as the basis functions are. Additionally, a different parameterization of the trust estimation for each parameter of global model structures is more adequate, to reflect the different influences of the parameters. But this would drastically increase the number of hyper-parameters to be set up by the designer and thus is not feasible. Yet, for a local GLT model structure, the trust estimation successfully reflected all sources of uncertainty which were introduced into the learning system. Furthermore, the application of this approach to electricity load forecasting demonstrated an improvement of the prediction quality

in two ways. The on-line learning system knows when its predictions are accurate. And using a threshold on the prediction's trust level, the part of predictions which is better than some reliable but suboptimal fallback can be identified. Consequently, the learning system can be used when its predictions are accurate to increase the performance and otherwise the fallback, like the steady prediction in this case, is taken to be safe. In this manner an on-line learning system can be integrated into any system using the trust management approach to define fallbacks and further increase the overall reliability and safety of the system while enhancing the performance when possible. Likewise, the trust estimation of the 24 hours ahead prediction correlates with the accuracy of the respective prediction thus allowing a better informed decision.

In a nutshell, the meta-information regarding the trustworthiness of each individual prediction opens up the possibility for advanced integration of an on-line learning system into a system architecture. Hence, more application areas of learning systems are possible where otherwise an uncertain prediction would endanger the system safety, as this case can be intercepted at a higher level. Yet, the approach for producing an uncertainty estimation for each individual prediction presented here is sensible only for local model structures and a suitable approach for global model structures would be of interest.

An information about the certainty of a learning system could be useful in other learning scenarios, not presented here, as well. As discussed in [73], such an enhancement is beneficial for *active learning* where the learning algorithm gets to choose an instance to be labeled for training. Here it could choose an instance for which its uncertainty is high to better train its prediction for this case. Furthermore, *reinforcement learning* needs to know which situations should be explored, as they are uncertain, and which are already known or certain, respectively. Or in *anomaly detection* where a distinction between new data that the algorithm knows nothing about, i. e. it has high uncertainty of the prediction, and abnormal data that the algorithm expected to be different has to be made. Thus, an application of the presented approach in these areas might provide further fruitful insights.

4

ON-LINE LEARNING – THE UNCERTAIN CASE

In this chapter, the on-line learning setting of Chapter 2 is extended to deal explicitly with uncertainties. On-line learning depends on the current parameter vector and the presented example, both of which may be more or less certain. Accordingly, the extension is twofold. First, the learning algorithm is extended to incorporate knowledge about the uncertainties of the parameter vector from Chapter 3. This yields a second order IRMA algorithm (SIRMA) which adapts itself to the situation at hand. Second, a general extension to incorporate explicit knowledge about the uncertainty of each training example is introduced.

4.1 UNCERTAINTY IN INCREMENTAL RISK MINIMIZATION

4.1.1 *Trustworthiness of the Parameter Vector*

The stiffness of IRMA reflects the adaptiveness to new examples. The more examples already have been presented, the less important a new example is. Thus, the stiffness of IRMA should be chosen to reflect the amount of examples already subsumed in the current approximation, i. e. its uncertainty due to ignorance. As different amounts of examples accumulate over time in different parts of the input space, the stiffness of IRMA should be a local influence on the learning algorithm as well. The basic idea as presented in [10] is pictured in Fig. 36. Here, for simplification, a linear approximation with two reference nodes is learned. If the example density is equal (a), both parameters should

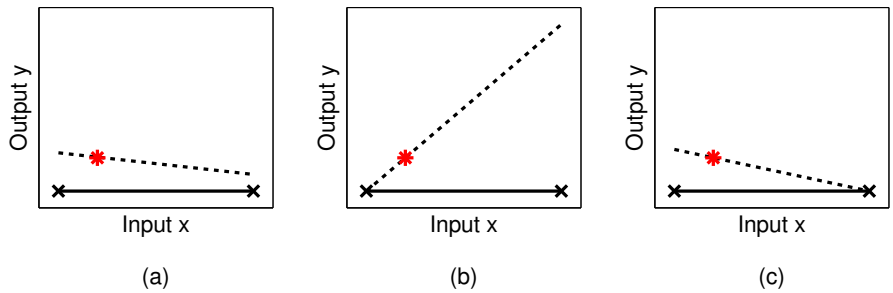


Figure 36: Three cases for learning a simple linear approximation (black line) on the same example (red asterisk) but with different data densities to give a new approximation (dotted line). Either both parameters have the same data density (a), the left density is higher than the right (b), or the right density is higher than the left (c).

be adjusted similarly, but the left one more due to its closeness to the example. If the right node has a low and the left a high example density (b) the current setup of the later is less trustworthy and thus should be adapted more. Contrariwise, if the left node has a much lower example density (c), it should be adapted more. These different situations can be reflected using a local stiffness regarding the local ignorance.

Hence, the incremental risk functional is extended with a local stiffness $\sigma_t(\mathbf{x}) > 0$ to

$$R_{inc2}(\boldsymbol{\omega}) = \frac{1}{2} \cdot \int_X L(\boldsymbol{\omega}_t^T \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\omega}^T \boldsymbol{\phi}(\mathbf{x})) \cdot \sigma_t(\mathbf{x}) d\mathbf{x} + \frac{1}{2} L(y_t, \boldsymbol{\omega}^T \boldsymbol{\phi}(\mathbf{x}_t)) \tag{89}$$

so as to change the functional behavior less in regions where a lot of examples already have been presented [4, 5].

To get the local stiffness regarding the example density the ignorance measure $\Phi_i(t)$ of (57) can be used. Every example increases the ignorance measure more for parameters that have a higher influence in the respective part of the input space. The local density $\zeta_t(\mathbf{x})$ is then derived from the parameter's ignorance $\Phi_i(t)$ through a linear combination regarding each parameter's activity

$$\zeta_t(\mathbf{x}) = \sum_{i=1}^{n} \Phi_i(t) \cdot \hat{\phi}_i(\mathbf{x}). \tag{90}$$

So the local density estimate depends on the parameter's densities in the same amount that the output depends on the parameters. This local density $\zeta_t(\mathbf{x})$ then influences the local stiffness of the model by increasing it linearly from an initial stiffness σ_0 with growth rate τ through

$$\sigma_t(\mathbf{x}) = \sigma_0 + \tau \cdot \zeta_t(\mathbf{x}) \; . \tag{91}$$

This way, the initial stiffness σ_0 allows to express how much the model adapts to an example initially and the growth rate can be used to steer how fast the stiffness increases with an example. So for SIRMA the adaptation for an example is changed throughout the learning process data dependently and the amount of adaptation is based on the example density in a local way. Consequently, if some region A in the input space has a high and another region B a low density, a new example will be incorporated into the model in such a way that basis functions with big influence in region A will be used less than basis functions with a bigger influence in region B (compare Fig. 36). This way, fatal forgetting is penalized more where the example density is high. With this approach the minimization of the incremental risk functional comes even closer to minimizing the risk functional of the batch version of (18), as the influence of the data distribution

is reflected, and a more robust and data dependent behavior can be expected.

For the minimum of the second order risk functional (89) follows $\forall\, i \in [1; n]$

$$\begin{aligned}\frac{\partial}{\partial \omega_i}\frac{1}{2}\cdot\int_X(\sigma_0+\tau\boldsymbol{\Phi_t}^T\hat{\boldsymbol{\phi}}(\boldsymbol{x}))((\boldsymbol{\omega}_t-\boldsymbol{\omega})^T\boldsymbol{\phi}(\boldsymbol{x}))^2dx+\\ \frac{1}{2}(y_t-\boldsymbol{\omega}^T\boldsymbol{\phi}(\boldsymbol{x}_t))^2\\ =\int_X(\sigma_0+\tau\boldsymbol{\Phi_t}^T\hat{\boldsymbol{\phi}}(\boldsymbol{x}))(\boldsymbol{\omega}_t-\boldsymbol{\omega})^T\boldsymbol{\phi}(\boldsymbol{x})\phi_i(\boldsymbol{x})dx+\\ (y_t-\boldsymbol{\omega}^T\boldsymbol{\phi}(\boldsymbol{x}_t))\phi_i(\boldsymbol{x}_t)=0\end{aligned}\tag{92}$$

which can be rewritten in vector form:

$$\begin{aligned}\int_X(\sigma_0+\tau\boldsymbol{\Phi_t}^T\hat{\boldsymbol{\phi}}(\boldsymbol{x}))(\boldsymbol{\omega}_t-\boldsymbol{\omega})^T\boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})dx+\\ (y_t-\boldsymbol{\omega}^T\boldsymbol{\phi}(\boldsymbol{x}_t))\boldsymbol{\phi}(\boldsymbol{x}_t)=0\end{aligned}\tag{93}$$

Using the definition of the matrix A

$$(A_t)_{i,j}=\int_X\phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x})\cdot\left(\sigma_0+\tau\cdot\left[\sum_{m=1}^{n}\Phi_m(t)\cdot\hat{\phi}_m(\boldsymbol{x})\right]\right)dx\tag{94}$$

as well as (29), this gets

$$A_t(\boldsymbol{\omega}_t-\boldsymbol{\omega})+(\boldsymbol{\phi}(\boldsymbol{x}_t)y_t-B(\boldsymbol{x}_t)\boldsymbol{\omega})=0\tag{95}$$

$$A_t\boldsymbol{\omega}_t-A_t\boldsymbol{\omega}+\boldsymbol{\phi}(\boldsymbol{x}_t)y_t-B(\boldsymbol{x}_t)\boldsymbol{\omega}=0\tag{96}$$

$$A_t\boldsymbol{\omega}_t+\boldsymbol{\phi}(\boldsymbol{x}_t)y_t=A_t\boldsymbol{\omega}+B(\boldsymbol{x}_t)\boldsymbol{\omega}\tag{97}$$

$$A_t\boldsymbol{\omega}_t+\boldsymbol{\phi}(\boldsymbol{x}_t)y_t=(A_t+B(\boldsymbol{x}_t))\boldsymbol{\omega}\tag{98}$$

as a condition for the critical point. Furthermore, as

$$\begin{aligned}\frac{\partial^2}{\partial \boldsymbol{\omega}^2}\frac{1}{2}\cdot\int_X(\sigma_0+\tau\boldsymbol{\Phi_t}^T\hat{\boldsymbol{\phi}}(\boldsymbol{x}))((\boldsymbol{\omega}_t-\boldsymbol{\omega})^T\boldsymbol{\phi}(\boldsymbol{x}))^2dx+\\ \frac{1}{2}y_t-\boldsymbol{\omega}^T\boldsymbol{\phi}(\boldsymbol{x}_t))^2\\ =\int_X(\sigma_0+\tau\boldsymbol{\Phi_t}^T\hat{\boldsymbol{\phi}}(\boldsymbol{x}))\boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{\phi}(\boldsymbol{x})dx+\boldsymbol{\phi}(\boldsymbol{x}_t)^T\boldsymbol{\phi}(\boldsymbol{x}_t)>0\end{aligned}\tag{99}$$

the critical point minimizes the incremental risk functional.

Hence, the resulting update of the parameter vector is given by

$$\boldsymbol{\omega}_{t+1}=(A_t+B(\boldsymbol{x}_t))^{-1}\left[A_t\boldsymbol{\omega}_t+\boldsymbol{\phi}(\boldsymbol{x}_t)y_t\right].\tag{100}$$

It is similar to that of IRMA in (35) but with a different, now time-dependent matrix A_t. Practically, this matrix is given by the linear combination

$$A_t=A^{(0)}+\sum_{m=1}^{n}\tau\cdot\Phi_m(t)\cdot A^{(m)}\tag{101}$$

with an initial matrix

$$(A^{(0)})_{i,j} = \sigma_0 \cdot \int_X \phi_i(x)\phi_j(x)dx \tag{102}$$

and a linear combination according to the parameter ignorance $\Phi_m(t)$ of matrices

$$(A^{(m)})_{i,j} = \int_X \phi_i(x)\phi_j(x)\hat{\phi}_m(x)dx,\ m = 1,\dots,n\,. \tag{103}$$

With $\sigma(x) > 0\ \forall\ x \in X$ the functions $\{\sqrt{\sigma(x)}\phi_i(x)\}_{i=1}^n$ are linearly independent for linearly independent $\phi_i(x)$ and form a basis in the function space $L^2(X)$. Hence, A_t is again the *Gramian matrix* given by their standard inner product and the matrix A_t is positive definite and has an inverse A_t^{-1}.

All matrices can be computed off-line in advance. But the linear combination has to be done on-line. Hence, the resulting algorithm for on-line learning by SIRMA is as follows:

Listing 2: Second Order Incremental Risk Minimization Algorithm

```
Parameter: initial stiffness σ_0, growth rate τ,
           initial parameter vector ω_0
for t = 0 to n_d − 1 do{ //i.e. for each learning step
  receive instance x_t
  predict label ŷ_t = ω_t^T φ(x_t)
  receive true label y_t
  suffer loss L(y_t, ŷ_t)
  build matrix A_t = A^(0) + Σ_{i=1}^n Φ_i(t) · A^(m)
  update parameter vector ω_{t+1} = (A_t + B(x_t))^{-1} · (A_t ω_t + φ(x_t) y_t)
  update parameter density ζ_{t+1} = ζ_t + φ̂(x_t)
}end
```

As the ignorance measure Φ is a strictly increasing function, the stiffness of SIRMA increases as well. In order to keep some adaptability in non-stationary environments, a trade-off to the robustness has to be made. Therefor, a mapping of the ignorance measure $\psi(\Phi)$ is used with the properties

$$\psi(\Phi) \leqslant \psi(\Phi') \quad \text{for} \quad \Phi \leqslant \Phi' \tag{104}$$

$$\psi(\Phi) \geqslant 0 \quad \forall\ \Phi \geqslant 0. \tag{105}$$

In this notion, the approach introduced above used $\psi(\Phi) = \tau \cdot \Phi$. As an alternative with a saturating stiffness, a sigmoidal increasing stiffness

$$\psi(\Phi) = \frac{\hat{\sigma}}{2}\left[1 + \cos\left(\min\left(1, \frac{\Phi}{\tau}\right)\pi + \pi\right)\right] \tag{106}$$

is used with a growth constant τ and a maximal stiffness of $\hat{\sigma}$. This is better suited for non-stationary environments and allows for continuous adaptation.

4.1.2 *Trustworthiness of the Example*

The uncertainty of an example influences how important the loss L on the current example is for minimization. Both, the information where and to what value the model should be trained are important and hence non-redundant. So, assuming a trust annotation of each example $(\mathbf{x}_t, y_t)$ with $\vartheta_{\mathbf{x}_t}$ and ϑ_{y_t} the total trustworthiness of the example is given by a t-norm $\vartheta_{\mathbf{x}_t,y_t} = T(\vartheta_{\mathbf{x}_t}, \vartheta_{y_t})$. If no information about the example's uncertainty is available, it is treated as fully trustworthy, i. e. $\vartheta_{\mathbf{x}_t,y_t} = 1$.

This total trustworthiness is then used to linearly blend between the contribution of the example loss and the change of the output in the risk functional according to

$$R_{inc3}(\boldsymbol{\omega}) = \frac{\sigma_0 + 1 - \vartheta_{\mathbf{x}_t,y_t}}{2} \cdot \int_X L(\boldsymbol{\omega}_t^T \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\omega}^T \boldsymbol{\phi}(\mathbf{x})) d\mathbf{x} + \frac{\vartheta_{\mathbf{x}_t,y_t}}{2} L(y_t, \boldsymbol{\omega}^T \boldsymbol{\phi}(\mathbf{x}_t)) \tag{107}$$

with an additional offset of the weight for the change by the stiffness σ_0. With this blending, (107) reduces to the normal incremental risk functional of (22) in case of full trustworthiness $\vartheta_{\mathbf{x}_t,y_t} = 1.0$, a risk functional only minimizing the change of the functional behavior for $\vartheta_{\mathbf{x}_t,y_t} = 0.0$, and a linear influence of the trustworthiness in between. This extension results in the IRMA update with a modified stiffness of

$$\sigma = \frac{\sigma_0 + 1 - \vartheta_{\mathbf{x}_t,y_t}}{\vartheta_{\mathbf{x}_t,y_t}}. \tag{108}$$

Consequently, with a vanishing trust $\vartheta_{\mathbf{x}_t,y_t} \to 0$ the stiffness increases to infinity and thus no adaptation is done if the example is not trusted. On the other hand, a fully trusted example is incorporated with stiffness σ_0, i. e. normal learning is performed. Thus, the more an example cannot be trusted, the lower is its influence on learning. So with this extension, the trustworthiness of an example can be readily used in each learning step. The trustworthiness can be incorporated into the SIRMA update presented in the previous section the same way by modifying the stiffness σ_0 in the initial matrix of (102) according to (108).

4.2 FORMAL ANALYSIS OF SIRMA

4.2.1 *Worst Case Minimization*

With the extension of a local data dependent stiffness, most basic properties of IRMA still hold. The local convergence proof of (39) is satisfied by SIRMA as well. But while locally contracting, the influence on the different parameters varies depending on the local stiffness.

Regarding the minimization of the worst case development of the global approximation error, with SIRMA a variation considering the example density can be found. Assuming that all examples are drawn independently and are identically distributed (i.i.d.) according to a density $\mathbf{x} \sim \rho(\mathbf{x})$, the global error of (44) can be rewritten as

$$\int_X (\boldsymbol{\omega}_{t+1}^T \boldsymbol{\phi}(\mathbf{x}) - f(\mathbf{x}))^2 \cdot \rho(\mathbf{x}) d\mathbf{x} \tag{109}$$

in compliance with the well known batch risk functional [107]. Here the prediction error at an input $\mathbf{x}$ is weighted by the respective density as an error for predictions that are less likely to occur is less important to be minimized. For this density weighted global error, the argumentation of (44) to (50) still holds and it can be upper bounded by

$$\begin{aligned} & \int_X (\boldsymbol{\omega}_{t+1}^T \boldsymbol{\phi}(\mathbf{x}) - f(\mathbf{x}))^2 \cdot \rho(\mathbf{x}) d\mathbf{x} && (110) \\ \leqslant\ & \int_X (\boldsymbol{\omega}_t^T \boldsymbol{\phi}(\mathbf{x}) - f(\mathbf{x}))^2 \cdot \rho(\mathbf{x}) d\mathbf{x} + \int_X ((\Delta\boldsymbol{\omega}_t)^T \boldsymbol{\phi}(\mathbf{x}))^2 \cdot \rho(\mathbf{x}) d\mathbf{x} \\ & + 2 \cdot C \cdot c(X) \cdot \sqrt{\int_X ((\Delta\boldsymbol{\omega}_t)^T \boldsymbol{\phi}(\mathbf{x}))^2 \cdot \rho(\mathbf{x}) d\mathbf{x}} && (111) \end{aligned}$$

Again, the first term is fixed by the previous approximation of $\boldsymbol{\omega}_t$ and can thus not be influenced by any learning algorithm. As the local stiffness reflects the example density, the second term as well as the third term are minimized by SIRMA for a given improvement on the current example $(\mathbf{x}_t, y_t)$ which depends on the overall stiffness $\overline{\sigma}$, if the local stiffness is proportional to the density, i. e. $\sigma(\mathbf{x}) = \overline{\sigma} \cdot \rho(\mathbf{x})$.

So looking at the expected value of the local stiffness

$$\begin{aligned} E[\sigma(\mathbf{x})] &= E\left[\left(\sum_{i=1}^{n}\left(\hat{\phi}_i(\mathbf{x}) \cdot \sum_{j=1}^{T} \hat{\phi}_i(\mathbf{x}_j)\right)\right) \cdot \tau + \sigma_0\right] && (112) \\ &= \left(\sum_{i=1}^{n}\left(\hat{\phi}_i(\mathbf{x}) \cdot \sum_{j=1}^{T} E\left[\hat{\phi}_i(\mathbf{x}_j)\right]\right)\right) \cdot \tau + \sigma_0 && (113) \\ &= \left(\sum_{i=1}^{n}\left(\hat{\phi}_i(\mathbf{x}) \cdot T \int \hat{\phi}_i(\boldsymbol{\nu})\rho(\boldsymbol{\nu}) d\boldsymbol{\nu}\right)\right) \cdot \tau + \sigma_0 && (114) \\ &\stackrel{!}{=} \overline{\sigma} \cdot \rho(\mathbf{x}) && (115) \end{aligned}$$

the appropriateness of the density estimation depends on the normalized basis functions $\hat{\phi}_i$. If these basis functions are in the form of *window functions* [91], they perform a blurring of the true density and thus yield an adequate approximation. This is for example the case for GLT model structures or polynomials on the positive half space. In these cases, the density estimation is equivalent to the well known *Parzen window* approach [91].

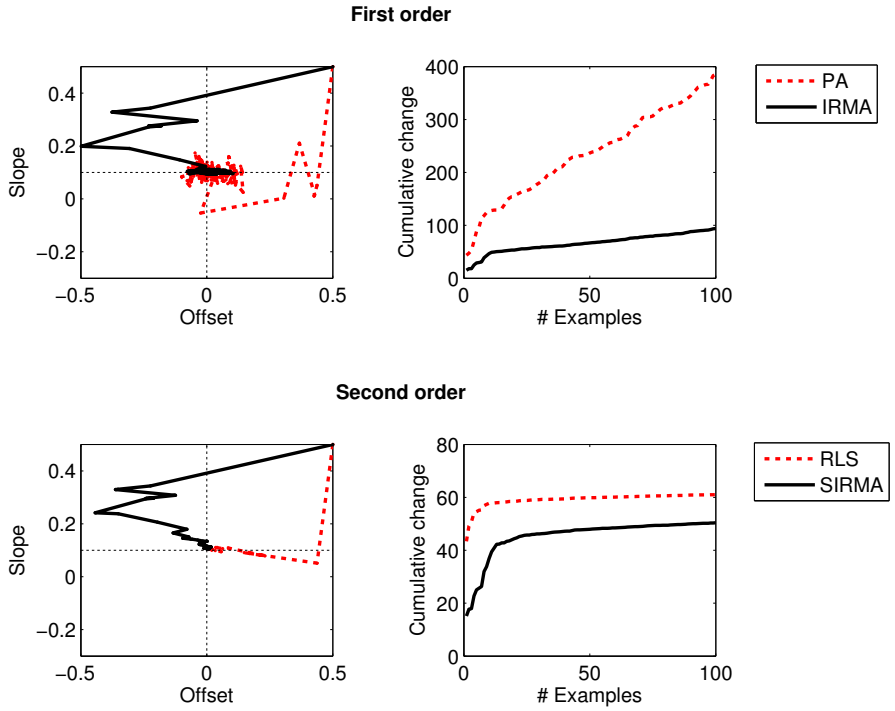


Figure 37: Comparison of different learning algorithms in parameter space (left; start at (0.5, 0.5); target at (0, 0.1)) and the resulting cumulative change of the output (right). Top shows first order algorithms PA and IRMA. Bottom shows second order algorithms RLS and SIRMA.

4.2.2 *Complexity*

With respect to the computational complexity, the extended learning algorithm of SIRMA takes $\mathcal{O}(n^3)$ steps, because of building the sum of matrices. It uses as well $\mathcal{O}(n^3)$ amount of memory, which is dominated by storing the n matrices. In comparison, state of the art second order learning algorithms take $\mathcal{O}(n^2)$ calculation steps and $\mathcal{O}(n^2)$ memory, but still the complexity of SIRMA is fixed depending on the complexity of the model structure with n basis functions and does not increase with increasing amounts of examples.

4.3 INVESTIGATIONS OF SIRMA

4.3.1 *Comparison of Learning in Parameter Space*

As a first basic investigation, a simple setup shows how the learning process progresses in parameter space to demonstrate the differences between IRMA and SIRMA and state of the art methods. In one input

dimension $x \in [-10, 10]$ $n_d = 100$ examples are generated equally distributed on the input of a linear target function[1]

$$y = 0.1 \cdot x + \xi$$

. The additive noise ξ is drawn according to a normal distribution with standard deviation 0.1. Using these examples a first order polynomial[2], with the parameters offset ($\omega_{t,1}$ with $\phi_1(x) = 1$) and slope ($\omega_{t,2}$ with $\phi_2(x) = x$), is learned starting with $\omega_{0,1} = \omega_{0,2} = 0.5$.

Figure 37 shows the progress of the parameter vector (left) and the cumulative global change of the output it causes (right) for different learning algorithms based on the same sequence of randomly chosen examples. As state of the art first and second order algorithms PA[3] and RLS[4] were chosen. With these mainly the slope is adapted at first and only then the offset is adapted to yield the correct parameter vector. As expected, the first order algorithm is prone to noise and cannot find a stable solution whereas RLS converges quickly despite the noise. In contrast to this, IRMA[5] does not adapt the slope that much as it has more impact on the resulting output, but adapts the offset as well in the beginning, thus resulting in a completely different trajectory in parameter space. Likewise, this preference is reflected in the remaining random adaptation of the first order algorithms around the target in parameter space. PA especially adapts the slope with every new noisy example and IRMA uses the offset to follow the noise. Similarly to RLS, SIRMA[6] increases the robustness to noise and thus converges to the target despite of the noise. The steps taken in parameter space by IRMA and SIRMA are bigger than for PA and RLS. But in output space these bigger steps still result in less adaptation. This is reflected by the cumulative change which is lower for IRMA as well as for SIRMA.

Therefore, the incorporation of the model structure into the risk minimization results in a different preference for parameters that are used to adapt to an example. In contrast to the state of the art methods which mainly adapt the parameters with high local influence irrespective of the global influence, the methods proposed here mainly adapt the parameters with low global influence. Obviously, the finally resulting parameter vector is the same and SIRMA successfully increases the robustness to noise.

1 UOSLib-scenario: `mode` = REG, `func` = lin, `ND` = 100, `NG` = 10, `noise` = 0.1, `minPath` = false, `rSeed` = 12345
2 UOSLib-model: Poly, `order` = 1
3 UOSLib-learn: PA, `variant` = 0
4 UOSLib-learn: RLS, `Sinit` = 10^5, `forget` = 1
5 UOSLib-learn: IRMA, `variant` = 0, `stiff` = 0.1
6 UOSLib-learn: SIRMA, `variant` = 1, `stiff` = 0.1, `growth` = 0.035

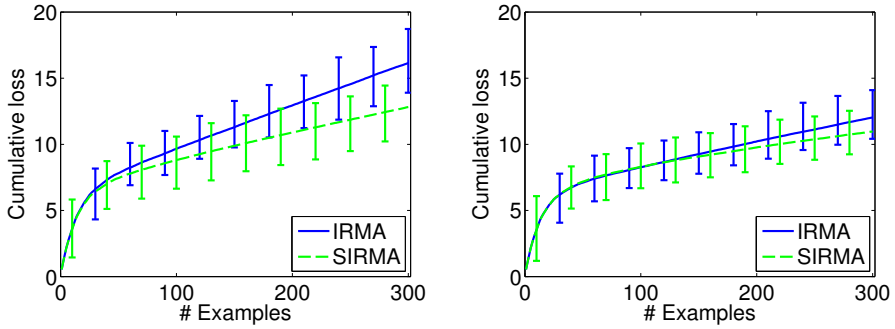


Figure 38: Comparison of IRMA and SIRMA on learning a simple target function with noise over 100 random sequences of examples using a GLT (left) or a polynomial (right) model structure. The average cumulative loss is shown with the respective minimum and maximum as error bars.

4.3.2 *Comparison of IRMA and SIRMA*

In a more difficult setup, the investigation of learning a sinusoidal target with additive noise $\xi \sim \mathcal{N}(0, 0.1)$, as shown in Fig. 20 of Section 2.4.3, is repeated with SIRMA[7] using a sigmoidal growth of the stiffness. The initial and maximal stiffness are chosen to be $\sigma_0 = 0.01$, $\hat{\sigma} = 5$ to start with a low stiffness for quick adaptation and get robust to noise later on. The density needed for maximal stiffness is $\tau = 10$ for the GLT model structure, as the 300 examples are equally distributed across the parameters and hence every parameter has an expected activation of $\frac{300}{16} \approx 18$. For the polynomial model structure the higher order monomials are activated much more than the lower order ones. Thus, to permit learning these parameters for a similar amount of examples, maximal stiffness is reached at a density of $\tau = 150$.

The results are shown in Fig. 38. In both cases, IRMA and SIRMA show a similar behavior but with a lower cumulative loss using the polynomial model structure (Fig. 38, right). SIRMA is just as robust as IRMA to varying sequences of examples as the error bars are quite small. In the beginning, i. e. for the first 50 examples, both approaches perform comparably but after initial learning, SIRMA is more robust to noise due to its increasing stiffness as the cumulative loss has a smaller slope.

This improved handling of noise can also be seen in Fig. 39. Here the same setup is used but with different magnitudes of normally distributed noise. The final cumulative loss after presentation of the same sequence of 300 examples (except for the noise magnitude) is shown depending on the noise level, i. e. the variance of the normal distribution. For low noise levels, IRMA gets the best result with a low

7 UOSLib-learn: SIRMA, `variant` = 2, `stiff` = 0.01, `growth` = [10,150], `maxstiff` = 5

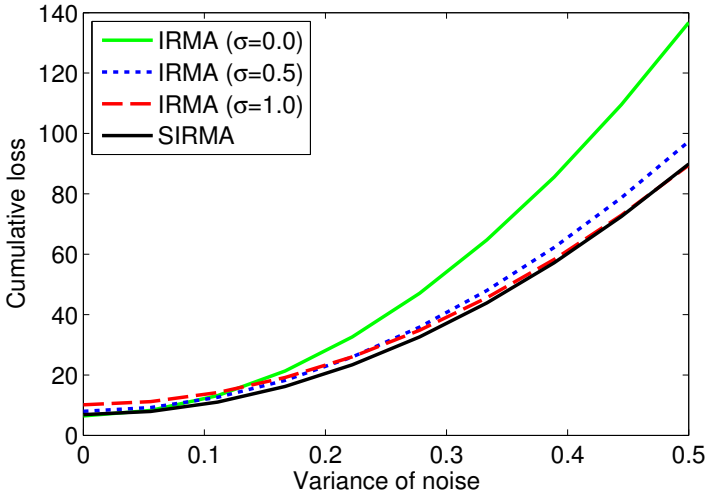


Figure 39: Comparison of the performance of IRMA with different fixed stiffnesses and SIRMA with adaptive stiffness on different noise levels.

stiffness (green solid line) but consequently worse results at higher noise levels. On the contrary, with high noise levels the best results are achieved with a high stiffness (red dashed line). With the adaptive stiffness of SIRMA (black solid line) an overall low cumulative loss can be achieved without the need to adapt the hyper-parameters to the noise level at hand. So a more robust learning is possible also with respect to engineering the hyper-parameters.

4.3.3 *Benchmark Datasets*

To cover a broader set of examples, IRMA and SIRMA are investigated in comparison to PA and RLS on several benchmarks from different domains. The benchmark datasets are part of the *NIST non-linear regression* archive [87] and the more difficult *motorcycle* dataset [54]. The selection of datasets contains different properties of training examples and thus covers typical learning situations. They differ with respect to the amount of non-linearity, how many changes in monotony occur, the overall noisiness of the examples, and locally differing noise levels and example densities. The properties are summarized in Table 8 with a minus, circle or plus indicating a low, medium or strong occurrence of the property, respectively. For comparability of the results, the datasets are scaled to the UOSLib standard, i. e. on the input to $[-10, 10]$ and on the output to $[-1, 1]$.

The basic PA[8] algorithm has no hyper-parameter to be set, RLS[9] is used with an initial covariance matrix $\Sigma_0 = \mathbb{1} \cdot 10^6$ and forgetting factor $\lambda = 1$ as the benchmarks are only static datasets, i. e. forget-

8 UOSLib-learn: PA, `variant` = 0

9 UOSLib-learn: RLS, `Sinit` = 10^6, `forget` = 1

Table 8: Properties of the benchmark datasets with a minus, circle or plus indicating a low, medium or strong occurrence of the respective property. The datasets are ordered roughly by increasing difficulty of the learning problem.

Properties	Lanczos3	Bennett5	Kirby2	Chwirut1	Hahn1	MGH17	Thurber	Chwirut2	Gauss1	Motorcycle
No. of Examples	24	154	151	214	236	33	37	54	250	133
Non-linearity	o	-	+	o	+	+	+	o	+	+
Non-Monotony	-	-	-	-	-	o	o	-	+	+
Noisiness	-	-	-	o	-	-	-	o	o	+
Local noisiness	-	-	-	-	-	-	-	o	o	+
Local sparsity	-	+	-	-	o	-	+	+	-	-

ting is not necessary. IRMA[10] uses a fixed low stiffness $\sigma_t = 0.01$ and SIRMA[11] a stiffness increasing sigmoidally from $\sigma_0 = 0.01$ to $\hat{\sigma} = 2$ with a growth of $\tau = 20.0$. For the approximation again a Gaussian GLT with 16 nodes[12] and a 15^{th} order polynomial[13] starting at $\omega_{0,i} = 0 \; \forall \; i \in [1;16]$ are used.

The important features of the learning algorithms investigated here are on the one hand the quality of on-line prediction, i. e. the cumulative loss of (52), and on the other hand as the datasets are static the correct representation of all examples by the final parameter vector, i. e. the mean data loss of (53).

The results of the experiments with the local GLT model structure are shown in Table 9 for the cumulative loss and in Table 10 for the mean data loss. Regarding the cumulative loss, PA, IRMA, and SIRMA perform quite similarly in this case. RLS sometimes achieves high losses due to overfitting but also may be lucky resulting in low losses. So its predictions are not reliable. And regarding the data loss (Table 10), a clear ranking of the methods is given even though the performance is quite similar as well. PA results in the highest data loss, followed by the same but sometimes lower data loss of IRMA, then SIRMA and RLS outperforms the other methods on every dataset. Yet, with this local model structure no significant advantage of any method can be seen on the data loss.

The results of the experiments with the global polynomial model structure are shown in Table 11 for the cumulative loss and in Ta-

10 UOSLib-learn: IRMA, `variant` = 0, `stiff` = 0.01

11 UOSLib-learn: SIRMA, `variant` = 2, `stiff` = 0.01, `growth` = 20, `maxstiff` = 2

12 UOSLib-model: GLT, `num` = 16, `base` = gauss

13 UOSLib-model: Poly, `order` = 15

Table 9: Final cumulative loss of different learning algorithms on the benchmark datasets using a GLT model structure. The lowest error for each dataset is marked in bold.

Dataset	PA	RLS	IRMA	SIRMA
Lanczos3	**6.4**	51.9	7.2	6.9
Bennett5	3.9	**3.7**	4.1	3.9
Kirby2	5.0	17.2	5.3	**4.8**
Chwirut1	8.4	190.0	8.9	**7.8**
Hahn1	6.0	11.2	6.4	**5.9**
MGH17	**5.4**	53.5	5.7	5.7
Thurber	**6.6**	15.8	7.1	6.8
Chwirut2	6.1	**6.0**	6.5	6.4
Gauss1	6.0	90.1	6.3	**5.5**
Motorcycle	16.3	158.0	16.4	**14.1**

Table 10: Final mean data loss of different learning algorithms on the benchmark datasets using a GLT model structure. The lowest error for each dataset is marked in bold.

Dataset	PA	RLS	IRMA	SIRMA
Lanczos3	$5.1 \cdot 10^{-2}$	$\mathbf{6.2 \cdot 10^{-4}}$	$4.2 \cdot 10^{-2}$	$1.7 \cdot 10^{-2}$
Bennett5	$1.8 \cdot 10^{-3}$	$\mathbf{7.3 \cdot 10^{-4}}$	$1.7 \cdot 10^{-3}$	$8.6 \cdot 10^{-4}$
Kirby2	$1.7 \cdot 10^{-3}$	$\mathbf{6.6 \cdot 10^{-4}}$	$1.7 \cdot 10^{-3}$	$9.7 \cdot 10^{-4}$
Chwirut1	$8.1 \cdot 10^{-3}$	$\mathbf{5.3 \cdot 10^{-3}}$	$8.1 \cdot 10^{-3}$	$6.6 \cdot 10^{-3}$
Hahn1	$1.8 \cdot 10^{-3}$	$\mathbf{1.1 \cdot 10^{-3}}$	$1.8 \cdot 10^{-3}$	$1.2 \cdot 10^{-3}$
MGH17	$1.4 \cdot 10^{-2}$	$\mathbf{1.1 \cdot 10^{-3}}$	$1.4 \cdot 10^{-2}$	$3.7 \cdot 10^{-3}$
Thurber	$5.0 \cdot 10^{-3}$	$\mathbf{1.0 \cdot 10^{-3}}$	$3.9 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$
Chwirut2	$1.9 \cdot 10^{-2}$	$\mathbf{4.3 \cdot 10^{-3}}$	$1.9 \cdot 10^{-2}$	$8.1 \cdot 10^{-3}$
Gauss1	$5.4 \cdot 10^{-3}$	$\mathbf{3.2 \cdot 10^{-3}}$	$5.3 \cdot 10^{-3}$	$3.9 \cdot 10^{-3}$
Motorcycle	$6.6 \cdot 10^{-2}$	$\mathbf{4.4 \cdot 10^{-2}}$	$6.6 \cdot 10^{-2}$	$5.3 \cdot 10^{-2}$

Table 11: Final cumulative loss of different learning algorithms on the benchmark datasets using a polynomial model structure. The lowest error for each dataset is marked in bold.

Dataset	PA	RLS	IRMA	SIRMA
Lanczos3	$1.0 \cdot 10^{23}$	$4.6 \cdot 10^{19}$	**8.6**	9.9
Bennett5	$6.2 \cdot 10^{23}$	$2.0 \cdot 10^{15}$	**6.6**	7.4
Kirby2	$6.9 \cdot 10^{23}$	$7.7 \cdot 10^{17}$	**9.4**	12.7
Chwirut1	$1.4 \cdot 10^{27}$	$1.5 \cdot 10^{22}$	**13.7**	14.6
Hahn1	$7.0 \cdot 10^{25}$	$7.1 \cdot 10^{16}$	**9.0**	9.5
MGH17	$4.4 \cdot 10^{23}$	$2.6 \cdot 10^{10}$	**10.0**	10.4
Thurber	$2.1 \cdot 10^{25}$	$1.7 \cdot 10^{19}$	20.3	**17.4**
Chwirut2	$1.3 \cdot 10^{26}$	$1.1 \cdot 10^{22}$	7.2	**7.1**
Gauss1	$5.0 \cdot 10^{26}$	$8.6 \cdot 10^{15}$	**6.7**	8.0
Motorcycle	$3.0 \cdot 10^{25}$	$1.2 \cdot 10^{13}$	15.0	**11.2**

Table 12: Final mean data loss of different learning algorithms on the benchmark datasets using a polynomial model structure. The lowest error for each dataset is marked in bold.

Dataset	PA	RLS	IRMA	SIRMA
Lanczos3	$3.4 \cdot 10^{24}$	1.1	$\mathbf{3.4 \cdot 10^{-2}}$	$4.8 \cdot 10^{-2}$
Bennett5	$5.2 \cdot 10^{19}$	$\mathbf{8.8 \cdot 10^{-6}}$	$1.5 \cdot 10^{-4}$	$3.3 \cdot 10^{-3}$
Kirby2	$4.4 \cdot 10^{22}$	$\mathbf{4.8 \cdot 10^{-8}}$	$4.8 \cdot 10^{-4}$	$2.7 \cdot 10^{-3}$
Chwirut1	$4.2 \cdot 10^{23}$	1.1	$\mathbf{7.5 \cdot 10^{-3}}$	$1.0 \cdot 10^{-2}$
Hahn1	$8.2 \cdot 10^{22}$	$\mathbf{6.0 \cdot 10^{-5}}$	$1.4 \cdot 10^{-3}$	$1.4 \cdot 10^{-3}$
MGH17	$2.9 \cdot 10^{22}$	$\mathbf{1.6 \cdot 10^{-2}}$	$3.5 \cdot 10^{-2}$	$4.2 \cdot 10^{-2}$
Thurber	$7.9 \cdot 10^{23}$	1.2	$\mathbf{1.9 \cdot 10^{-2}}$	$5.3 \cdot 10^{-2}$
Chwirut2	$6.4 \cdot 10^{23}$	5.4	$6.2 \cdot 10^{-2}$	$\mathbf{2.6 \cdot 10^{-2}}$
Gauss1	$1.0 \cdot 10^{20}$	$\mathbf{1.6 \cdot 10^{-3}}$	$2.8 \cdot 10^{-3}$	$1.8 \cdot 10^{-3}$
Motorcycle	$4.6 \cdot 10^{24}$	$\mathbf{4.2 \cdot 10^{-2}}$	$8.1 \cdot 10^{-2}$	$4.8 \cdot 10^{-2}$

ble 12 for the mean data loss. Here, IRMA and SIRMA successfully prevent high cumulative losses for low data densities even if such global model structures are used and thus can achieve a low cumulative loss. Especially on the *motorcycle* dataset a significantly lower cumulative loss is achieved by SIRMA. This dataset has locally differing noise levels and thus the robustness of SIRMA to the need of a fittingly selected stiffness helps to deal with the data. But the increasing stiffness of SIRMA is not always beneficial. RLS and PA have a high cumulative loss for all datasets and cannot handle the global model structure well, conforming previous investigations.

The first order method PA also gets high data losses on every dataset. Hence, it is not suitable for global model structures. But the data loss indicates that RLS, IRMA, and SIRMA converge and learn an adequate representation of the datasets. So the high cumulative loss of RLS is a result of aggressive parameter adaptation in between. For several datasets RLS even achieves a lower data loss, hence a better approximation in the end, but for some datasets like *Lanczos3*, *Chwirut1*, *Thurber*, and *Chwirut2* the amount of examples were not sufficient for RLS to converge. Even though *Chwirut1* consists of 214 examples in contrast to 54 for *Chwirut2*, these are only noisy examples for the same discrete instances. With these, RLS cannot converge as fast. Hence, RLS again cannot handle every dataset reliably, but IRMA and SIRMA are reliable in every case regarding as well the final hypothesis as every learning step in between and always achieve low losses.

4.3.4 *Uncertain Examples*

Lastly, the influence of respecting the trustworthiness of each training example is investigated. Therefor, again the sine function is used as a scenario[14]. It is learned by a 15^{th} order polynomial model structure[15] using IRMA as the basic learning algorithm[16] to investigate the influence of the trust without additional stiffness adaptation. Based on this setup, different disturbance cases of the instance x_t and the label y_t are simulated.

Two basically different kinds of disturbances are simulated in this setting. Either, the data are subject to noise with zero-mean. This means that even though the data are disturbed, their mean value provides information about the true value. Or, the data are subject to noise with nonzero-mean. This means that the disturbance has some unknown bias. Both disturbances are applied either to the instance x_t or the label y_t. Yet, it should be noted that from the perspective

14 UOSLib-scenario: `mode` = REG, `func` = sine, `ND` = 300, `NG` = 20, `noise` = 0.0, `minPath` = false, `rSeed` = [12346:12376]

15 UOSLib-model: Poly, `order` = 15

16 UOSLib-learn: IRMA, `variant` = 0, `stiff` = 0.01

of the learning system, the effect of instance and label noise is the same. But for most input-output relations, i. e. non-linear ones, zero-mean noise on the instance results in nonzero-mean noise of the label. The trustworthiness of the instance or label is lowered according to the severity of the disturbance to simulate knowledge about the example's trustworthiness. This simulates ideal knowledge about the example's trustworthiness which might not be given in a real application where the trustworthiness has to be determined somehow and might be subject to disturbances on its own. Yet, this allows to analyze how the approach can perform in the best case.

For all experiments each example is disturbed with a 50% chance or otherwise left exact so that about half of the examples can be used for normal learning. First to simulate zero-mean noise, the instance x_t is disturbed with an additive normally distributed noise. Its magnitude is altered by the variance. A zero variance corresponds to full trust, a variance of 0.3 corresponds to no trust and in between the trust is blended linearly. Second to simulate nonzero-mean noise, the instance x_t is disturbed by randomly drawing an input value form $[-10, 10]$ to replace the correct input value of the example. The magnitude of this disturbance is altered by the probability with which a random value replaces the correct input. A zero probability is assigned with full trust in the example, a 25% probability leads to zero trust and in between the trust scales linearly. The other two scenarios disturb the label y_t the same way by an additive normally distributed noise or by randomly drawing a replacement form $[-1, 1]$. Finally, after the total sequence of 300 training examples was presented, the ground truth loss of the learned model is measured, i. e. how well the true target was learned.

This experiment was repeated with 30 different random seeds to analyze the robustness and 30 different disturbance magnitudes for each and accordingly selected trust levels. For comparison, learning was done either with IRMA and full trust in every example, i. e. without respecting the trustworthiness, or with the extended influence of the trustworthiness on learning where every example that might be disturbed is assigned with the corresponding trust level and every undisturbed example is fully trusted.

The results, shown in Fig. 40, present the average ground truth loss and its variance over the thirty different random seeds as error bars. The disturbance significantly increases the ground truth loss for normal learning without trust. With respecting the trust it increases only a little or even is about the same. Thus, the advantage of respecting the trustworthiness of each example at learning is clearly apparent. The performance is expected to decrease with increasing disturbance, but as the examples with a lower trust influence the learning less, the learning system is still able to achieve a good ground truth loss just on the 50% trustworthy training examples. Furthermore, the vari-

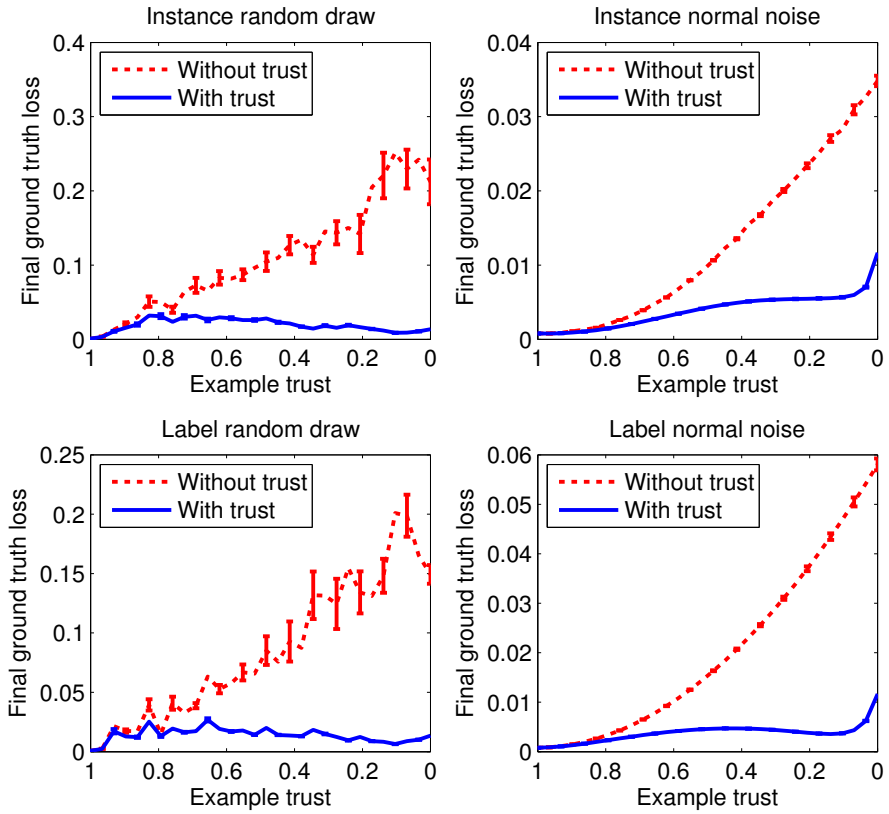


Figure 40: Comparison of the performance of IRMA with and without respecting each example's trustworthiness. The two disturbance cases of random values (left column) or additive noise (right column) are simulated for the instance uncertainty (upper row) and label uncertainty (lower row). The scales of the y-axis differ due to different impacts on the resulting quality.

ance of the results without respecting the trust is very high in case of nonzero-mean noise. This variance is significantly lower when the trust is used, i. e. no variance can be seen at all. On average over all four cases, it is only 30% of the variance without respecting the example's trust. Hence, it is robust to different sequences of training data and different disturbances. The influence of nonzero-mean noise on the ground truth loss of normal learning is significantly higher than that of zero-mean noise as seen by the different scales of the y-axis. But with respecting the trustworthiness, both disturbances are handled equally well.

Consequently, influencing the stiffness with the trustworthiness of the presented example helps to make the learning system more robust. But this approach has its limits. If all examples cannot be trusted, obviously there is not enough information to learn at all. Furthermore, as usually the correct trust is not known but itself only an estimate, it may underestimate or overestimate the true trustworthiness. If the trustworthiness is underestimated, i. e. the example could

be trusted more, learning will be slower but it will still work on the long run. On the contrary, if the trustworthiness is overestimated, i. e. the example should be trusted less, the robustness deteriorates to normal learning as the disturbance is not known. But in this case it never gets worse than normal learning. Therefore, the improvement through respecting the trust depends on an adequate estimation of the trustworthiness of an example, but in the worst case it only slows down learning.

4.3.5 *Application to Electricity Load Forecasting*

The electricity load forecasting task presented in Section 2.5 is non-stationary. Thus, the learning algorithm needs to be adaptive throughout the whole learning process which is achieved by SIRMA with a sigmoidally increasing stiffness. So the same investigations on step ahead and 24 hours ahead prediction with GLT and polynomial model structure as in Section 2.5 are repeated with SIRMA. It is set up to start with full adaptivity using an initial stiffness of $\sigma_0 = 0.0$ and grows up to a maximal stiffness that is equal to the optimal fixed stiffness of IRMA in Table 5. Hence, it only is used to locally improve initial learning.

For step ahead prediction with both model structures and 24 hours ahead prediction with a GLT model structure, the results are the same as for IRMA with a fixed stiffness. An improvement of the load prediction is achieved by SIRMA only for 24 hours ahead prediction with the polynomial model[17]. In this case, SIRMA achieves a prediction error of 7.79 in contrast to IRMA with 7.85.

With the polynomial model structure, each parameter has a global influence. As SIRMA increases the stiffness locally where more examples have been presented, it adapts the globally effective parameters such that fatal forgetting is prevented in the beginning even better than with IRMA. That is why there is no significant improvement when using a GLT on the same task, as the model structure itself helps to prevent fatal forgetting through its locally effective parameters.

An increase time of the stiffness $\tau = 168$ yields the best results. This equals about one month of training examples with the 15 parameters ($168 \cdot 15 = 2520 \approx 2688 = 4 \cdot 7 \cdot 24 \cdot 4$). As this first month accounts for only about 4% of the whole data sequence, the overall improvement through SIRMA is quite low in this case.

4.4 CONSEQUENCES

In the uncertain case, the on-line learning algorithm typically is subject to an uncertain hypothesis as well as uncertain training examples. To deal with the uncertainty of the hypothesis, the trust estimation

17 UOSLib-learn: SIRMA, `variant` = 2, `stiff` = 0.0, `growth` = 168, `maxstiff` = 1.1

of the parameter vector presented in Chapter 3 was included in the learning algorithm. The extension of IRMA to incorporate knowledge about the uncertainty of the parameter vector by trust levels for each parameter prevents fatal forgetting dynamically. The same learning situation with different underlying data densities, and hence uncertainties due to ignorance, is thus resolved differently. Moreover, the density increases the stiffness in such a way that at first with a low number of examples a quick adaptation is possible to gain knowledge and reduce ignorance. And when more examples are presented, the adaptiveness decreases yielding a higher robustness to noise and to reduce conflict. But this higher robustness leads to a lower adaptiveness in non-stationary environments, which is a principal trade-off. Therefor, a sigmoidal mapping of the ignorance measure makes it possible to combine the initialization performance of SIRMA locally and data dependent with the long-term adaptiveness of IRMA which also further increased the performance of electricity load forecasting, albeit slightly.

But the investigations showed that the improvement of SIRMA is comparably low. At the same time, the additional computational complexity is high even though it is still limited. Hence, it should be considered if SIRMA is necessary when setting up an application. The main benefits of SIRMA come into play when the target function is static and a high noise level is present, the noise level varies significantly throughout the input space, or the data density in input space varies significantly during the learning process. Otherwise IRMA already provides a good approach with lower complexity. Further improvement on choosing an adequate local stiffness throughout the learning process would be of interest for future developments. Ideally, the stiffness should not only increase to be robust to noise, but it should also be capable of decreasing again if a shift or drift occurs in non-stationary environments. This way the trade-off between adaptability and robustness could be treated dynamically.

To deal with uncertain examples that are known to be uncertain, learning by IRMA and SIRMA was extended to incorporate the trust level of each example. Incorporating this information makes the learning algorithm more robust to disturbances. The presented approach significantly improves the robustness not only to zero-mean noise but also to the more severe disturbance of nonzero-mean noise. It was shown that the performance deteriorates only gently with increasing uncertainty of the training examples when these uncertainties are known. Otherwise the disturbances severely affect the predictive quality and hence the learning system's reliability. But this improvement is only possible, if a good estimate of the uncertainty of examples is available. Yet, a wrong uncertainty estimation is not perilous for the learning system. If the uncertainty estimation is wrong, the robust-

ness just may not improve as much or the learning progress may be slowed down.

So with this extension, the learning system is able to use trust signals of the training examples. This way, it can be integrated better into a system wide trust management architecture where the trustworthiness of every information is provided.

5 CONCLUSION

5.1 DISCUSSION

5.1.1 *On-line Learning by IRMA*

On-line machine learning in general and especially regression help to make modern systems intelligent. The new approach to on-line learning, called IRMA, presented in this work, improves the state of the art with a reliable learning behavior. It respects the locality of a single example, on the one hand in time by adapting to the most recent examples presented, and on the other hand in input space by changing the global model as little as possible. Therefore the influence of a non-linear model structure is directly incorporated into the learning algorithm. As all state of the art on-line learning algorithms only minimize the change of the parameter vector, this influence has not been studied before, even though it has an important impact on the predictive quality of an on-line learning system. In contrast to state of the art learning algorithms the resulting behavior with IRMA depends less on the specific chosen model structure but only on its expressiveness. Thus, as shown by the worst case analysis in Section 2.3.2, it can be applied reliably to *any* model structure suited to the data, whereas other learning algorithms have significant drawbacks especially with globally effective parameters.

Hence, IRMA meets all the requirements of Section 1.4. The investigations showed that the cumulative loss of IRMA is low in every setting. Therefore, IRMA always learns quickly and reliably a good approximation (Requirement *a*) and achieves a good predictive quality. Its cumulative loss is not necessarily minimal in every case, because a "lucky guess" is always possible with any learning algorithm in the predictive on-line setting. Yet, in addition to lucky guesses, unlucky guesses are just as possible with other algorithms. But in contrast to that, IRMA is proven to minimize the worst case loss in every step. This way, it prevents the learning system from too unlucky guessing which means fatal forgetting is inhibited in contrast to state of the art algorithms (Requirement *c*). That is why IRMA never shows a bad performance in any investigation. However, other learning algorithms are prone to fatal forgetting and result in a high cumulative loss at least in some settings.

The main drawback of RLS as the best representative of second order learning is that it is impossible to know a priori if a setting will result in high prediction errors and thus in a high cumulative loss.

Especially on low data densities, i. e. in the beginning of learning or when new regions of the input space are reached due to shifts or drifts in the underlying data, RLS is not reliable. On the other hand, the main drawback of PA as the best representative of first order learning is that it can learn only local model structures adequately and otherwise is prone to fatal forgetting. In these cases, IRMA prevents overfitting and is inherently robust against overly expressive model structures (Requirement *g*), thus easing the design of a learning system. At the same time, a continuous adaptation to non-stationary environments is provided as the learning does not decrease over time (Requirement *b*). This allows learning in situations where no fixed optimal knowledge is available and consequently a learning system is inherently necessary.

The only hyper-parameter that has to be set up is the stiffness. It has a smooth influence on the predictive quality with a distinct optimal setting which is proportional to the noise level present in the examples. So engineering of this hyper-parameter is straightforward as opposed to the forgetting factor of RLS. It is comparable, albeit inverse, to the aggressiveness of PA. Besides, the complexity of the algorithm is low which makes it on the one hand easy to understand and on the other hand easy to apply to big data or in systems with low computational power like embedded systems (Requirement *d*).

Consequently, IRMA can be applied in a wide range of settings. With IRMA, it is possible to choose the model structure depending on the task and not depending on the restrictions imposed by the learning algorithm because IRMA has no restrictions. It allows to learn non-linear input-output relations with the simplicity of a linear learning scheme. Thus it can deal with tasks, e. g. the higher dimensional concrete dataset, that needed more elaborate non-linear learning systems like multi layer perceptrons or support vector regression before. Furthermore, it enhances the reliability, as an adequate recall of presented examples is ensured, i. e. no fatal forgetting occurs, as far as the model structure is capable of representing them, and generalization is compliant due to robustness to overfitting.

5.1.2 *Trust Estimation*

Learning from examples is inherently uncertain. Hence, knowledge about this uncertainty is important to increase the reliability of a learning system and its integration into a system architecture. With monitoring the parameter vector throughout learning, the inherent uncertainties of the learning system can be estimated. No general uncertainty estimation for on-line learning of LIP model structures has been proposed before. The uncertainty measures introduced here estimate epistemic uncertainties through ignorance of the learning system as well as aleatoric uncertainties through conflict within the

learning system. For both influences, short term and long term monitoring is provided.

This way, every parameter of the parameter vector is attributed by a trust signal reflecting its trustworthiness at each learning step (Requirement *i*). This allows a designer of a learning system to supervise the learning process and to adjust its setup. If the parameter vector is uncertain, the expressiveness of the model structure can be changed (Requirement *g*). In case of too much ignorance, the model structure might be chosen too complex and should be reduced. Contrariwise, in case of too much conflict, the model structure might not be expressive enough and it should be increased. But the cause of conflict might not be the model complexity but just as well the noise on the examples. Then an increased stiffness of IRMA stabilizes the learning system against the noise.

Moreover, the trust estimation of the parameter vector allows to reflect the trustworthiness of each individual prediction (Requirement *j*). So in addition to the prediction itself, a trust level is provided as a meta-information for other processing modules or a supervisor. Yet, the approach is limited as the trustworthiness of each individual prediction is most meaningful only for local model structures like a GLT. With global model structures, the trust estimation of individual predictions is less sensible.

Such an extended on-line learning system can be integrated with other processing modules according to the trust management approach, e. g. in the COBRA-architecture [3]. This further increases the reliability of the overall system as uncertainties of the on-line learning system can be dealt with at higher system levels or subsequent system modules. If the prediction is too uncertain, a fallback strategy can be applied as demonstrated in the electricity load prediction. Then the resulting total system performance is enhanced when possible but still safe in presence of uncertainties. With this extension, on-line learning can be considered even in safety critical applications.

5.1.3 *On-line Learning by SIRMA*

To deal with known uncertainties at learning, the algorithm should be able to adequately handle uncertain information. With SIRMA, the information about the uncertainty of the parameter vector due to ignorance is used to influence the learning algorithm (Requirement *f*). It thus increases the stiffness locally the more, the lower the ignorance is. Thereby, the learning algorithm adapts itself to the data seen so far and fatal forgetting is prevented even better than with IRMA in case of varying data densities. This advantage is particularly significant with global model structures whereas local model structures are better in dealing with varying data densities by construction. This way,

the predictive quality increases throughout learning as the influence of noise decreases.

Additionally, the increasing stiffness of SIRMA enhances initial learning. It enables a quick adaptation in the beginning and is robust to varying noise levels on the long run, similar to other second order learning algorithms like RLS. But the growing stiffness comes at the cost of a declining adaptiveness. This is a principal trade-off that has to be considered when solving a task with an on-line learning system. When a shift or drift occurs, forgetting is necessary which contradicts robustness to noise. So in a non-stationary environment the designer needs to adjust the learning system to this trade-off. This dilemma of non-stationarity and noise is also addressed in RLS through the forgetting factor, but the setup of this hyper parameter is very sensitive making RLS instable.

Moreover, incorporating the trustworthiness of an example to lower its influence if it is uncertain further increases the reliability (Requirement *e*). In case of a system that is enhanced by trust signals, the information about an example's trustworthiness can be respected by the learning algorithm. Thus the learning system is more robust to disturbed examples that are known to be uncertain and the knowledge is protected. Consequently, the predictive quality is not harmed.

5.1.4 *Consequences*

A guideline for choosing an on-line learning algorithm from the set of PA, RLS, IRMA, and SIRMA is presented in Fig. 41. This guideline aims at good predictive quality and low computational complexity. The main distinction is whether the target function is static or time variant. If in case of a static target function a reliable prediction is not necessary at every step, i. e. only the final result is important, RLS is the best suited algorithm. Otherwise, if a time variant target function is learned with a local model structure with roughly equal influences of all parameters, PA is suitable due to its low computational demand, but IRMA might further improve the performance. In all other cases, IRMA provides a better solution as long as the noise level is not too high and the examples are more or less uniformly distributed. In presence of higher noise or varying example densities, SIRMA with linearly increasing stiffness for a static target or sigmoidally increasing stiffness for a time variant target enables further improvement. So RLS is a good solution in a special case and otherwise the other algorithms meet increasing demands but with increasing computational complexity at the same time. Yet in all cases, IRMA and SIRMA are sure not to fail as the worst case is minimized in each learning step which is not guaranteed with other learning algorithms.

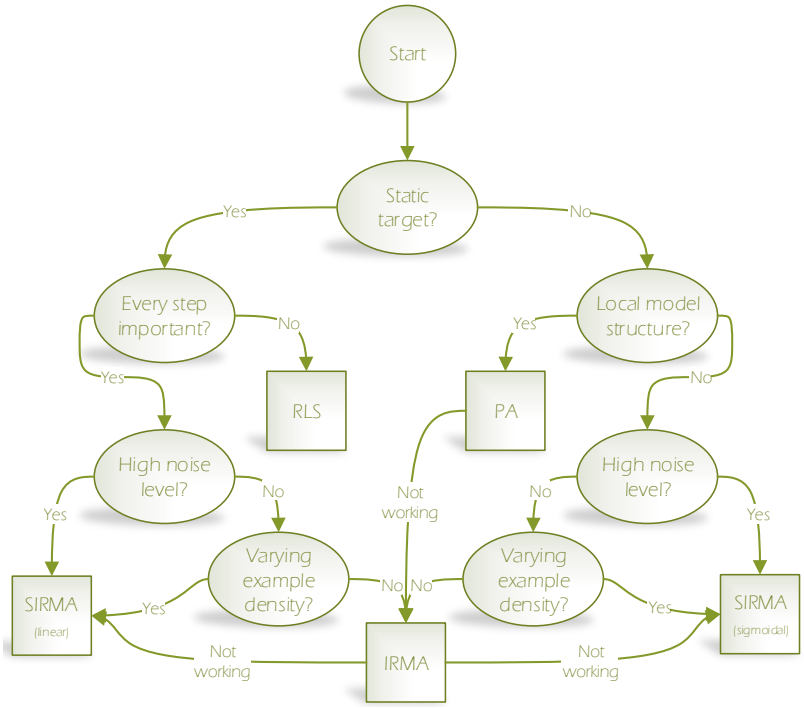


Figure 41: Flow chart for selecting the most suitable on-line learning algorithm for LIP regression.

5.2 SUMMARY

This thesis deals with the on-line learning problem as an important research field of machine learning. The challenges of on-line learning were discussed and a categorization of uncertainties influencing a learning system was presented. With IRMA a novel approach to on-line learning was introduced, resulting in a plausible and reliable update of the model with every presented example regardless of the specific model structure used. With this approach, the local validity of each example is respected, even with global model structures, by keeping the global change of the input-output relation as low as possible. Hence, fatal forgetting due to overfitting is prevented inherently.

An extension to reflect the uncertainties due to ignorance and conflict of such a learning system on-line by trust levels enhances the interpretability. It also assesses the uncertainty of each individual prediction. This way, it enables the safe integration of an on-line learning system into a system architecture that deals with uncertainties explicitly. At the same time, this mechanism is used to improve the learning algorithm by incorporating knowledge about ignorance into the update of the learning system to reduce fatal forgetting. Furthermore, to increase the robustness to disturbed training examples the trustworthiness of each presented example is incorporated into learning and reduces its influence according to its uncertainty.

All contributions were analyzed theoretically and their properties were demonstrated in empirical investigations and in a real world application on electricity load forecasting. The new approach to on-line learning clearly outperforms state of the art methods in several situations. A guideline for choosing a learning algorithm clarifies when state of the art methods might be more suited or when the presented learning algorithms should be used. Furthermore, the analysis of the trust estimation shows that ignorance and conflict are adequately reflected. As shown in the step ahead electricity load forecasting, this additional trust information enhances the applicability of learning on a system level. No other approach for on-line uncertainty estimation is known that can be applied to LIP model structures with any learning algorithm.

Altogether, the main contribution of this work are the on-line learning algorithms IRMA and SIRMA together with the trust estimation of the parameter vector and following from this a trust estimation for each individual prediction. They yield a reliable on-line learning system for LIP model structures with more insight for an engineer and a reliable prediction assessing its own uncertainty to react accordingly on a system level.

5.3 OUTLOOK

Not all aspects of on-line learning could be handled within this work. The solution to robustness against uncertain instances at prediction (Requirement *h*), was not presented in this thesis. A first approach to deal with this problem has been investigated in a masters thesis [98] and first results were published in [3]. The basic idea is to fade out the influence of an uncertain input dimension by averaging the predicted label along the uncertain dimension. This way, the influence of the uncertain value diminishes until a totally uncertain dimension has no influence on the prediction at all. The influence of this fading on the trustworthiness of the prediction is also accounted for by consideration of the conflict it causes. But an extensive analysis of this approach is still pending. The same work presented an approach for reliable interpolation and extrapolation by blending out the influence of uncertain parameters of the parameter vector. So far this approach is only computationally feasible for GLT model structures.

Some further interesting future directions can be derived from the presented work. One remaining problem that was discussed is the trade-off between robustness and adaptiveness in on-line learning. One recently presented approach uses an adaptive learning rate for a stochastic gradient descent in multi layer perceptron learning [97]. There the learning rate decreases in presence of noise to improve the robustness and it increases again in presence of shifts or drifts in order to readapt. A similar adaptive stiffness for IRMA would be

possible. Likewise, the second order version SIRMA might be extended to incorporate not only the ignorance measure but also the conflict measure into its update, to achieve a similar result.

Moreover, the basic IRMA principle presented in Section 2.2.1 is even more widely applicable. In this work it was applied to regression with LIP model structures. It could also be used the same way for other tasks like classification or density estimation or for other model structures that are non-linear in the parameters. But it is not clear whether an analytical solution of the minimization of the risk functional exists in this case or if a suitable approximation could be found. Yet, such a system would provide the same benefits of preventing fatal forgetting and overfitting, but in other domains.

The trust estimation presented here has limited validity for trust prediction with polynomial model structures. Hence, an extension of this approach that is meaningful for any LIP model structure, e. g. polynomials, would expand the applicability. Furthermore, such a trust estimation of on-line learning systems poses research questions on an architectural level as well. Here, an incorporation of a learning system with a trust attribute for its prediction into the system with a safe fallback strategy is of interest. A first idea of such an application was presented with the step ahead electric load forecasting. But the integration of the predicted trust on an architectural level could also influence higher decision levels that trigger specific countermeasures instead of just resorting to a fallback. This way the presented on-line learning system could be used reliably even in safety-critical applications.

A

UOSLIB

A.1 MOTIVATION

In the raise of this work, the unified online-learning systems library (UOSLib) [8] was developed. It is an open source library of on-line learning algorithms for Matlab®. The intention of this library is twofold. On the one hand, it is possible to compare different approaches to on-line learning within a common framework, e. g. to see how new approaches rank in comparison to the state of the art. On the other hand, a task at hand can be solved easily with different learning algorithms to find the most suitable approach. The UOSLib is focused on the two tasks of generalized linear *classification* and *regression* which are two of the most important learning tasks. It contains several learning algorithms and provides different model structures.

It features a suitable common framework which is easily extensible with new learning algorithms and model structures in order to systematically evaluate new methods. Furthermore it allows an easy setup of investigations with different properties of a learning scenario, e. g. degree of data noisiness, data linearity, or independence of consecutive data. Just as well, the model structure is exchangeable to evaluate its impact on the resulting performance.

Depending on the learning task the evaluation of the output and the loss function differ. For regression the evaluation is given by

$$\hat{y}_t = \boldsymbol{\omega}_t^T \boldsymbol{\phi}(\mathbf{x}) \tag{116}$$

with $\hat{y}_t \in \mathbb{R}$. A commonly used loss function is the *squared loss*

$$L_r(\hat{y}_t, y_t) = (\hat{y}_t - y_t)^2. \tag{117}$$

For binary classification the evaluation is given by

$$\hat{y}_t = \operatorname{sgn}(\boldsymbol{\omega}_t^T \boldsymbol{\phi}(\mathbf{x})) \tag{118}$$

with $\hat{y}_t \in \{-1, +1\}$. Here a commonly used loss function is the *hinge loss*

$$L_h = \begin{cases} 0 & \text{if } \hat{y}_t \cdot y_t \geqslant 1 \\ 1 - \hat{y}_t \cdot y_t & \text{otherwise .} \end{cases} \tag{119}$$

Within this scope of the on-line learning setting the UOSLib provides a basis to compare a variety of model structures $\boldsymbol{\phi}(\mathbf{x})$ and learning algorithms $\boldsymbol{\omega}_t \times (\mathbf{x}_t, y_t) \to \boldsymbol{\omega}_{t+1}$ together with a collection of different learning scenarios for both tasks.

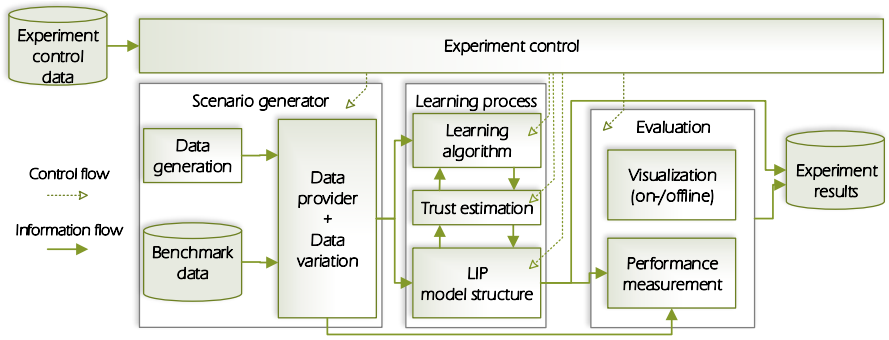


Figure 42: Block diagram of the UOSLib modules.

A.2 UOSLIB ARCHITECTURE

To support the development of new learning algorithms with the UOSLib, Matlab® was chosen as a widespread basis on the one hand to allow for rapid prototyping and easy analysis with onboard tools for plotting and evaluation and on the other hand because it is optimized for linear algebra operations which is the core of LIP model structures. The concept of the UOSLib is based on three modules as shown in Fig. 42. The *scenario generator* provides the examples that are presented to the learning algorithm. The *learning process* consists of the learning algorithm with its model structure and the trust estimation. Finally, an automatic *evaluation* measures the performance and visualizes the results. Most importantly, a setup can be uniquely specified by a footprint consisting of one tuple specifying the scenario, one specifying the model structure, and one specifying the learning algorithm, thus making it reproducible as well by other researchers. These footprints will be introduced in the following sections.

The *scenario generator* can be set up to generate synthetic data sets in a reproducible way to test the algorithms in different situations based on exactly the same data. The scenario generator hence allows to adjust specific properties of the learning scenario like the target function, noise level on the target labels, example density, independence of consecutive examples, etc. This way, the influence of these properties on the different learning algorithms can be evaluated systematically. Through a fixed random seed, the complete setup is reproducible and can be applied to different combinations of learning algorithm and model structure. Additionally, an interface to load (benchmark) datasets from a file is provided for an easy junction to external data sources. For external data some properties can be adjusted systematically as well, like added noise and the order of presentation. All generated learning scenarios are easily mapped either to a regression task or to a binary classification task by taking the sign of the output.

For the *learning process* common interfaces to different model structures as well as learning algorithms are provided to easily exchange

the structure or algorithm in use for comparison. This allows to test different combinations of

- learning task: regression or classification
- learning scenario: different properties of the data
- model structure: mapping input to parameter space $\mathbb{R}^d \rightarrow \mathbb{R}^n$
- learning algorithm: update of the parameter vector

by exchanging or systematically varying one of the four parts. In all cases, the learning process is monitored by the trust estimation as it is independent of the learning algorithm and model structure.

The learning process is *evaluated* with different measures that are tracked over time. As on-line learning is a continuous process, a cumulative performance measurement in a single number cannot present all relevant information. Rather, the progress of different measures over time is important in order to analyze the behavior. Therefor, the UOSLib incorporates the same three measures as presented in Section 2.4. They are updated after every learning step t. The cumulative loss L_c in (120) corresponds to the on-line performance, i. e. how well the predictive quality is at the respective step. In contrast to that, the mean data loss L_d in (121) corresponds to the quality on all examples seen up to the respective step, i. e. how well the examples were learned. Furthermore, for synthetic data it is possible to estimate the mean ground truth loss L_g in (122), i. e. how well the learned approximation suits undisturbed and regularly sampled examples, thus covering the ability to generalize and cancel noise. To determine this, additional test examples $(\widetilde{x}_i, \widetilde{y}_i)$ are directly drawn on a fine-grained regular grid covering the complete input space regardless of the density of training data and without any disturbance.

$$L_c(t) = \sum_{\tau=0}^{t} L(\boldsymbol{\omega}_\tau^T \boldsymbol{\phi}(\mathbf{x}_\tau), y_\tau) \tag{120}$$

$$L_d(t) = \frac{1}{t} \sum_{\tau=0}^{t} L(\boldsymbol{\omega}_t^T \boldsymbol{\phi}(\mathbf{x}_\tau), y_\tau) \tag{121}$$

$$L_g(t) = \frac{1}{N_g} \sum_{i=0}^{N_g} L(\boldsymbol{\omega}_t^T \boldsymbol{\phi}(\widetilde{\mathbf{x}}_i), \widetilde{y}_i) \tag{122}$$

With this framework, it is simple to set up different investigations of on-line learning systems and to easily compare different algorithms. The three measures give deeper insight than using only the cumulative loss and help to comprehend the learning process. Moreover, each investigation is easily reproducible.

A.3 MAIN UOSLIB MODULES

A.3.1 *Scenario Generator*

The scenario generator provides a set of examples which is used for training in a sequential order. For this purpose, it is given to the model structure as well as to the learning algorithm (see Fig. 42). An according set of ground truth data is provided for the performance evaluation. All scenarios are scaled to enforce that the instances lie in the interval $[-10, +10]$ and the label is in $[-1, +1]$. By fixing the seed `rSeed` of the random number generator, the examples are fully reproducible. The generator is called through the function `icl_loadDS` which has the following interface:

```
[data groundTruth dim] =
  icl_loadDS(mode, func, ND, NG, noise, minPath)
```

The setup specifies the learning task with the parameter `mode` which selects either *regression* or *classification*. The underlying function used to generate the examples is given by the string `func` that selects one of several predefined test functions (see Tab. 13) that differ in their amount of non-linearity and changes of the monotonicity. If this string starts with `dataset`, it is interpreted as a path to a file from which the examples are loaded. Parameter `ND` specifies the number of examples to generate for learning and parameter `NG` the number of regularly sampled ground truth data per dimension for evaluation. The parameter `noise` sets the standard deviation of normally distributed noise on the training labels. Lastly, the parameter `minPath` selects whether the examples are ordered randomly or in a sequential way, resembling a continuous movement within input space. If the parameter `minPath` is `true`, the randomly generated examples are ordered in such a way that, starting at the lowest value in each dimension, i. e. -10, the next value in the sequence is chosen from the remaining set of randomly drawn examples to have a minimal distance in input space to the current value (see Fig. 43 for an example). This simulates for instance the behavior of dynamic systems, i. e. moving on a trajectory in input space.

The scenario generator function returns the examples in `data` as a two dimensional matrix with one example in each row and the ground-truth data for comparison in `groundTruth` the same way. Depending on the function selected for generation, the dimensionality of the scenario differs which is returned in `dim`. All in all, the learning scenario hence can be uniquely identified by the following footprint:

(`mode`, `func`, `ND`, `NG`, `noise`, `minPath`, `rSeed`)

Table 13: Provided scenarios of the UOSLib. The fine grained parametrization is due to the normalization of the target labels to $[-1, 1]$.

func	Description	Dim.
`linear`	straight line $f(x) = 0.1x,\ x \sim \mathcal{U}(-10, 10)$	1
`nonlin`	exponential function $f(x) = 2\,\lvert x\rvert\, e^{-\frac{\lvert x\rvert}{2}} - 1,\ x \sim \mathcal{U}(-10, 10)$	1
`nonlinhalf`	positive half-space of exponential function $f(x) = (x+10)e^{-\frac{x+10}{2}},\ x \sim \mathcal{U}(-10, 10)$	1
`poly`	polynomial $f(x) = \frac{1}{14}(4 - 0.2x + 3\cdot 10^{-2}x^2 - 1\cdot 10^{-3}x^3 + 2\cdot 10^{-4}x^4 - 4\cdot 10^{-5}x^5 - 2\cdot 10^{-6}x^6)$ $x \sim \mathcal{U}(-10, 10)$	1
`sine`	sine function $f(x) = \sin(x),\ x \sim \mathcal{U}(-10, 10)$	1
`sineloc`	sine function with local data density $f(x) = \sin(x),\ x \sim \mathcal{N}(\pi, 4)$	1
`linear2`	linear plane $f(x) = 0.03x_1 + 0.07x_2,\ x_i \sim \mathcal{U}(-10, 10)$	2
`twocircles`	minimum of distance to two corners $f(x) = \frac{1}{11.3}\left(\min((x_1+10)^2 + (x_2-10)^2, (x_1-10)^2 + (x_2+10)^2) - 11.39\right)$ $x_i \sim \mathcal{U}(-10, 10)$	2
`crossedridge`	crossed ridge function $f(x) = 1.6211 \cdot \max(\exp(-0.3x_1^2), \exp(-0.09x_2^2), 1.25\cdot\exp(-0.1(x_1^2 + x_2^2))) - 1$ $x_i \sim \mathcal{U}(-10, 10)$	2
`spiral`	spiral loop (typical classification task)	2
`highdimlin`	linear hyperplane $f(x) = \boldsymbol{w}^T\boldsymbol{x},\ w_i \sim \mathcal{N}(0, 1),\ x_i \sim \mathcal{U}(-10, 10)$	20
`highdimnonlin`	squareroot hyperplane $f(x) = \boldsymbol{w}^T\sqrt{\boldsymbol{x}},\ w_i \sim \mathcal{N}(0, 1),\ x_i \sim \mathcal{U}(-10, 10)$	20
`shift`	3d-order-polynomial changing sign after half of training data $f_t(x) = w_t \cdot (-0.198 + 0.06x + 0.003x^2 - 0.0015x^3)$ $w_t = 1 \text{ if } t < \frac{N_D}{2}, w_t = -1 \text{ if } t \geqslant \frac{N_D}{2}, x \sim \mathcal{U}(-10, 10)$	1
`drift`	7th-order-polynomial drifting after one third of training data $f(x) = w_t(-0.094187 + 0.11262x - 0.00059436x^2 - 0.00012187x^3 - 7.9019\cdot 10^{-7}x^4$ $-1.385\cdot 10^{-5}x^5 + 7.86\cdot 10^{-7}x^6)(1 - w_t)(-0.5471 - 0.069x + 0.0037x^2$ $+0.00067908x^3 + 0.00010338x^4 + 7.5788\cdot 10^{-6}x^5 - 6.0123\cdot 10^{-7}x^6)$ $w_t = \min(1, \max(0, \frac{3t}{N_D} - 1)),\ x \sim \mathcal{U}(-10, 10)$	1
`dataset...`	If the string starts with `dataset`, it will be interpreted as a relative path to a file. The file should contain arbitrary many columns of input dimensions and one last column of the target label with space-separation.	

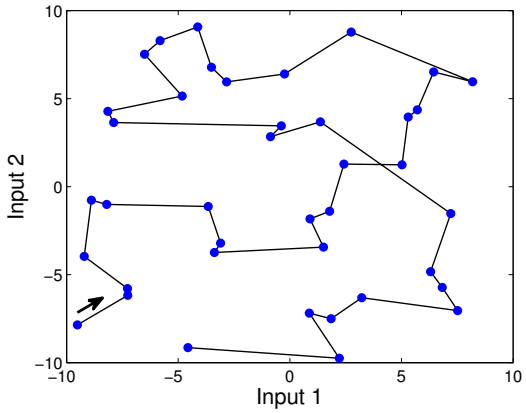


Figure 43: Example of a randomly generated set of 40 examples with its according minimal path shown by the connections reflecting the order of presentation.

A.3.2 *Learning Algorithms*

The main goal of the UOSLib is to easily implement and compare a variety of new and existing state of the art learning algorithms for regression as well as for classification. The learning algorithm receives examples from the scenario generator and the parameter vector of the model structure which then is updated (see Fig. 42). On the one hand, the learning algorithms can be divided into the groups of *first* and *second order* learning algorithms. On the other hand, they are distinguished regarding their applicability to regression and/or classification. Currently the list of implemented learning algorithms contains the following.

First order algorithms are:

- Perceptron: Classical on-line learning algorithm [95]
  Footprint: (Perceptron)
- PA: Passive-aggressive algorithm in three variants [36]:
  - PA: Parameter update fully aggressive
    Footprint: (PA, `variant=0`)
  - PA-I: Limited aggressiveness (linear slack variable)
    Hyper-parameter: aggressiveness `aggr`
    Footprint: (PA, `variant=1`, `aggr`)
  - PA-II: Limited aggressiveness (quadratic slack variable)
    Hyper-parameter: aggressiveness `aggr`
    Footprint: (PA, `variant=2`, `aggr`)
- IRMA: Incremental Risk minimization algorithm in four variants [6, 12]:
  - Fixed stiffness
    Hyper-parameter: stiffness `stiff`
    Footprint: (IRMA, `variant=0`, `stiff`)

  - Additive growing stiffness
    Hyper-parameter: initial stiffness `stiff`, addend `growth`
    Footprint: (IRMA, `variant=1`, `stiff`, `growth`)
  - Multiplicative growing stiffness
    Hyper-parameter: initial stiffness `stiff`, factor `growth`
    Footprint: (IRMA, `variant=2`, `stiff`, `growth`)
  - Sigmoidal growing stiffness
    Hyper-parameter: initial stiffness `stiff`, growth constant `growth`, maximum stiffness `maxstiff`
    Footprint: (IRMA, `variant=3`, `stiff`, `growth`, `maxstiff`)

Second order algorithms are:

- CW: Confidence weighted learning [47]
  Hyper-parameter: probability of correct classification `probab`
  Footprint: (CW, `probab`)
- AROW: Adaptive regularization of weight vectors [38, 105]
  Hyper-parameter: aggressiveness `r`
  Footprint: (AROW, `r`)
- GH: Gaussian herding in four variants [39]:
  Hyper-parameter: aggressiveness `aggr`
  - Full matrix
    Footprint: (GH, `variant=0`, `aggr`)
  - Exact diagonal matrix
    Footprint: (GH, `variant=1`, `aggr`)
  - Drop off-diagonal elements of matrix
    Footprint: (GH, `variant=2`, `aggr`)
  - Project diagonalized inverse matrix
    Footprint: (GH, `variant=3`, `aggr`)
- RLS: Recursive least squares [21]
  Hyper-parameter: factor of initial covariance matrix `Sinit`, forgetting factor `forget`
  Footprint: (RLS, `Sinit`, `forget`)
- SIRMA: Second order incremental risk minimization algorithm in two variants [4]:
  - Linear growing stiffness
    Hyper-parameter: initial stiffness `stiff`, addend `growth`
    Footprint: (SIRMA, `variant=1`, `stiff`, `growth`)
  - Sigmoidal growing stiffness
    Hyper-parameter: initial stiffness `stiff`, growth constant `growth`, maximum stiffness `maxstiff`
    Footprint: (SIRMA, `variant=2`, `stiff`, `growth`, `maxstiff`)

Of these algorithms Perceptron, PA, AROW, and GH are applicable to regression and classification, IRMA, RLS, and SIRMA are applicable for regression, and CW is applicable for classification.

As a common interface, two interface function are necessary for every learning algorithm. The first provides algorithm specific initializations, e. g. the covariance matrix of RLS, based on the algorithm setup `algSetup` containing the above hyper-parameters of the footprints. It therefor receives the structure `ILS` as it is generated by the model structure setup and the input dimensionality. Any information to be saved is changed within the returned `ILS` structure.

```
ILS = icl_initILS(ILS, dim, algSetup)
```

The second function is used to update the `ILS` structure, i. e. its parameter vector and if necessary further information, with one single example incrementally. It receives again the `ILS` structure, as well as the example (`x`, `y`), the label `yp` predicted beforehand, and the `mode` reflecting whether the task is regression or classification.

```
ILS = icl_learn(ILS, x, y, yp, mode)
```

A.3.3 *Model Structures*

The model structure is used to keep the parameter vector and perform predictions. Therefore, it receives the examples and reports the results to the performance measurement (see Fig. 42). It is determined by the vector of basis functions $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^n$. The UOSLib provides three different model structures. First, an additive polynomial of the form

$$f(\mathbf{x}, \boldsymbol{\omega}) = \omega_0 + \sum_{k=1}^{d} \sum_{m=1}^{N_o} \omega_{(k,m)} \cdot (x_k)^m \tag{123}$$

is given as an example for model structures with *globally* effective parameters by the function `icl_genPoly`. It produces an additive combination of polynomials of order N_o in every component x_n of the input vector (see Fig. 3 for an example). This model structure has the advantage of increasing only linearly in complexity with the scenario dimensionality at the cost of highly interacting parameters. The according function has the interface

```
ILS = icl_genPoly(order, dim)
```

receiving the polynomial order and the dimensionality. It returns a structure, containing the vectors of basis functions $\phi_{n,m}(x)$ and the parameter vector $\boldsymbol{\omega}$.

In contrast to that, two variants of a *local* model structure are implemented as a grid-based lookup table (GLT), one with an equally spaced grid (Fig. 44 left) and one with a spacing which can be set arbitrarily (Fig. 44 right).

In this case, the linear combination

$$f(\mathbf{x}) = \sum_{i=1}^{n} \phi_i(\mathbf{x}) \cdot \omega_i \tag{124}$$

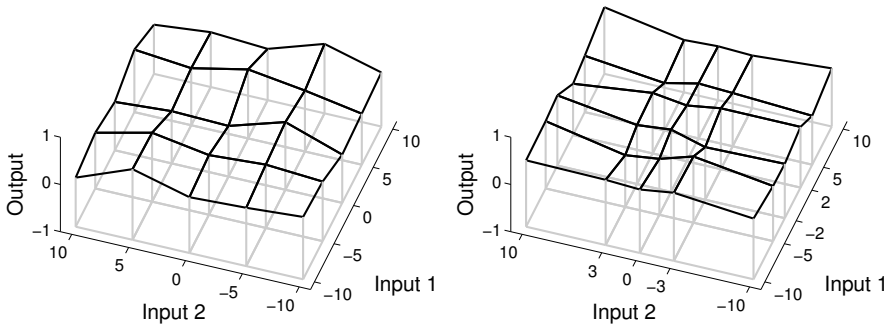


Figure 44: Example of a grid-based lookup table with linear interpolation for two dimensions. The grid-points are either equally spaced, here with five points per dimension (left), or specified dimension-wise here at $\{-10, -5, -2, 0, 2, 5, 10\}$ for the first dimension and $\{-10, -3, 0, 3, 10\}$ for the second (right).

consists of local basis functions ϕ_i which are given by the multiplicative combination

$$\phi_i(\mathbf{x}) = \prod_{j=1}^{d} \phi_{k_j(i),j}(x_j) \tag{125}$$

of their dimension-wise local functions. These are either triangular

$$\phi_{k,j}(x_j) = \max\left(0, \min\left(\frac{x - p_{k,j}}{p_{k+1,j} - p_{k,j}}, \frac{p_{k,j} - x}{p_{k,j} - p_{k-1,j}}\right)\right) \tag{126}$$

for linear interpolation or Gaussians

$$\phi_{k,j}(x_j) = \exp\left(\frac{\ln(0.5)(p_{k,j} - x_j)^2}{\sigma^2}\right) \tag{127}$$

with a width $\sigma = \frac{1}{2d}\sum_{i=1}^{d} p_{k+1,j} - p_{k-1,j}$ as the mean distance between all neighboring grid-points. The locations $p_{k,j}$ define the kth grid-point in the jth input dimension.

On such a grid in input space, the output is thus defined by the height of the model at the grid-points and either a linear or a Gaussian interpolation in-between. Linear interpolation ensures total locality of the parameter influence, whereas Gaussian interpolation results in a smooth surface. The parameters of a GLT thus only have local influence on the output and hence do not interact as much. But with such a local structure, the curse of dimensionality leads to an exponentially increasing number of parameters with increasing dimensionality which makes it infeasible in higher dimensions. For implementing this structure either `icl_genGLT` can be used with the interface

```
ILS = icl_genGLT(num, dim, base)
```

to get a regularly spaced grid in input space with the parameters `num` for the number of grid positions in each of the `dim` dimensions and `base` to choose between linear and Gaussian.

Alternatively, a more application specific setup of a GLT structure is possible with the function `icl_genGLTarb` by defining an arbitrary grid structure through

```
ILS = icl_genGLTarb(loc, dim, base)
```

where an array of locations `loc` is specified containing one location array for each input dimension. For the example of Fig. 44 (right) this array has the form $\{[-10,-5,-2,0,2,5,10]\ ,[-10,-3,0,3,10]\}$.

The used model structure is uniquely identified as well by:

(Poly, `order`) or (GLT, `num`, `base`) or (GLT, `loc`, `base`)

These model structures hence allow for an easy comparison of learning algorithms with globally or locally effective parameters. This distinction is especially significant for on-line learning as a single example only presents local information for the parameter update.

A.3.4 *Trust Estimation*

Along with the normal on-line learning, the UOSLib allows for tracking the trustworthiness of the parameter vector and evaluating the trustworthiness of each individual prediction as described in Section 3.2, regardless of the model structure and learning algorithm used. The trust estimation interacts with the model structure and the learning algorithm (see Fig. 42) and updates its information by keeping track of every update made by the learning algorithm. For that purpose, the initialization function

```
ILS = icl_initCAROLA(ILS,trustSetup)
```

receives the structure `ILS` and setup information of the trust estimation `trustSetup`. This defines either the parameters for a linear trust estimation by (58), (61), and (63)

(`Lin`, `Ilow`, `Ihigh`, `Clow`, `Chigh`, `Dlow`, `Dhigh`)

with `Ilow` and `Ihigh` as the ignorance parameters, `Clow` and `Chigh` as the conflict parameters, and `Dlow` and `Dhigh` as the adjustment parameters of the direct estimate. Or it defines the parameters for a hyperbolic trust estimation by (59), (62), and (65)

(`Sig`, `etaI`, `etaC`, `etaA`)

with `etaI` as the ignorance parameter, `etaC` as the conflict parameter, and `etaA` as the adjustment parameter of the direct estimate.

The trustworthiness is then automatically tracked during learning for supervision and trust estimation of each prediction.

B

EXTENDED INVESTIGATIONS

Some more specific investigations and detailed results are omitted from the main part of this thesis to focus on the most important results. These specific investigations are presented here for completeness.

B.1 EXPRESSIVENESS OF THE MODEL STRUCTURE

Designing a learning system poses several challenges. One is to decide on the expressiveness of the model structure to use. With a model structure that is not expressive enough, the examples cannot be learned adequately but using a too expressive model structure might result in slow convergence or bad generalization due to overfitting. To look at the behavior of the different on-line learning algorithms in these cases, the same scenario is learned here with model structures of increasing expressiveness.

For training, 300 random instances $x_t \sim \mathcal{U}(-10, 10)$ are drawn. For each the target value is generated by the function

$$y_t = (x_t + 10) \cdot \exp(-\frac{x_t + 10}{2}) + \xi$$

with normally distributed noise $\xi \sim \mathcal{N}(0, 0.05)$ to form an example[1]. Both variants, polynomial model structures[2] and GLT[3] model structures are used with one to sixteen parameters. The final cumulative loss is taken after learning with IRMA[4] using a stiffness $\sigma = 0.1$, PA[5], and RLS[6] with an initial covariance matrix $\Sigma_0 = \mathbb{1} \cdot 10^3$ and a forgetting factor of $\lambda = 1$.

The results of Fig. 45 show that with less than eight parameters the expressiveness is not sufficient which results in a high cumulative loss. As the basic PA tries to learn every example exactly, it has difficulties with too low expressiveness and gets the highest cumulative loss with the GLT model structure. RLS again has the problem that a high cumulative loss might occur as happened in this case with four or seven parameters of the GLT model structure. In this case RLS tends to overfit the model structure with low expressiveness to the

1 UOSLib-scenario: `mode` = REG, `func` = nonlinhalf, `ND` = 300, `NG` = 100, `noise` = 0.05, `minPath` = false, `rSeed` = 12345
2 UOSLib-model: Poly, `order` = [0;15]
3 UOSLib-model: GLT, `num` = [1;16], `base` = gauss
4 UOSLib-learn: IRMA, `variant` = 0, `stiff` = 0.2
5 UOSLib-learn: PA, `variant` = 0
6 UOSLib-learn: RLS, `Sinit` = 10^3, `forget` = 1

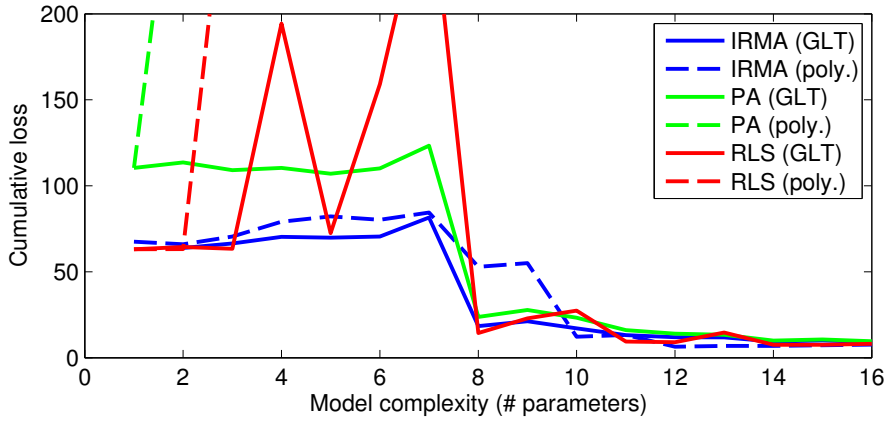


Figure 45: Comparison of IRMA, PA, and RLS on the same scenario with model structures of different expressiveness. High cumulative losses are cut off for better visibility of the lower losses.

singular examples which causes false generalization. With more than eight parameters a significantly lower loss is achieved with all learning algorithms. So the quality of learning is not hampered by a higher expressiveness. But with the polynomial model structure PA and RLS again receive high cumulative losses. On the contrary, IRMA is able to learn both model structures nearly equally well and neither suffers from too low nor from too high expressiveness.

B.2 GROWING STIFFNESS

In the IRMA investigations of Section 2.4 the stiffness was chosen to be constant throughout the sequence of examples. But if only some examples are present, i. e. at initial learning, there is no old knowledge and a low stiffness allows to learn quickly from the examples. As soon as more examples have accumulated, a higher robustness to noise and hence keeping the old knowledge is more appropriate. Consequently, the stiffness should increase with the amount of examples presented. Therefore, a time dependent stiffness σ_t should be monotonically increasing: $\sigma_t \leqslant \sigma_{t+1}$.

Three different approaches to a growing stiffness starting with an initial stiffness σ_0 are compared here. First, an additive growth

$$\sigma_t = \sigma_{t-1} + \tau_a \tag{128}$$

increases the stiffness linearly by $\tau_a > 0$ with each presented example. Second, a multiplicative growth

$$\sigma_t = \sigma_{t-1} \cdot \tau_m \tag{129}$$

increases the stiffness exponentially with a factor $\tau_m > 1$ after each example. Third, a sigmoidal growth function

$$\sigma_t = \sigma_0 + \frac{1}{2}\left[1 + \cos\left(\min(1, \frac{t}{\tau_s}) \cdot \pi + \pi\right)\right] \cdot (\hat{\sigma} - \sigma_0) \tag{130}$$

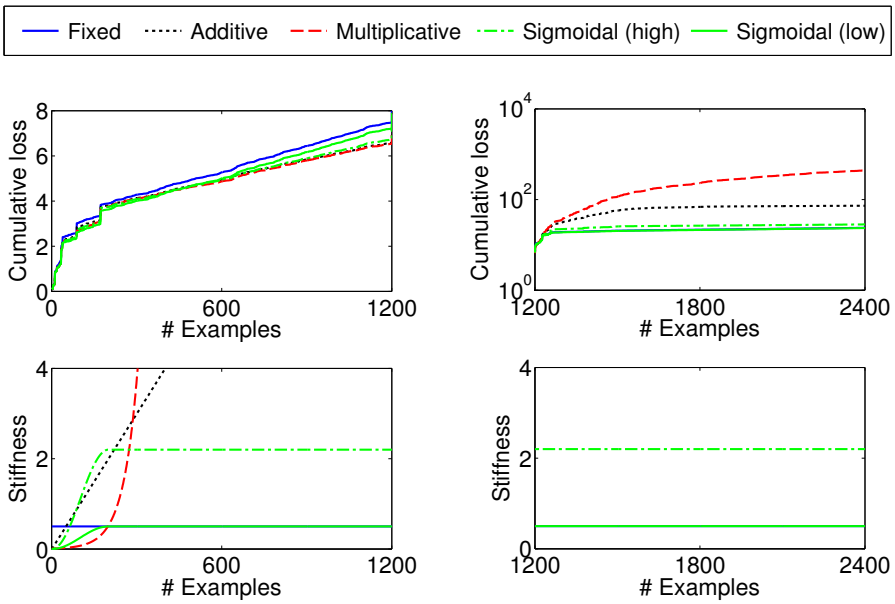


Figure 46: Comparison of different growth functions of the stiffness for the first 1200 examples (left) and after a shift for the next 1200 examples (right) of the sequence. The lower plots show the growth of stiffness along the sequence of examples (high stiffnesses are cut off for better visibility). The upper plots show the accompanying development of the cumulative loss.

increases the stiffness from σ_0 up to a maximum value $\hat{\sigma}$ in time τ_s.

To compare these growth functions of the stiffness, the shift investigation of Section 2.4.6 with polynomial model structure is performed again[7], but now with noise $\xi \sim \mathcal{N}(0, 0.05)$ and on a longer sequence of $n_d = 2400$. As a baseline, the fixed stiffness $\sigma = 0.5$ is chosen[8] to balance between adaptability and robustness to noise. The additive growth[9] starts with $\sigma_0 = 0.0$ and $\tau_a = 0.01$, the multiplicative growth[10] starts with $\sigma_0 = 0.01$ and $\tau_m = 1.02$, and the sigmoidal growth[11] starts with $\sigma_0 = 0.0$ and $\tau_s = 200$. For the maximum value, a high value of $\hat{\sigma} = 2.2$ is compared with a low value $\hat{\sigma} = 0.5$ which is equal to the fixed stiffness.

In the left plot of Fig. 46 the resulting cumulative losses and the stiffnesses are shown for the first half of the sequence of examples, i. e. the first static target function of the shift scenario. Basically, with all approaches the input-output relation is learned well. In the beginning, they perform comparably. But on the long run, the growing stiffness helps to deal with the noisy examples and decreases the cumulative loss in all cases, compared to the fixed stiffness (see also Table 14).

7 UOSLib-scenario: `mode` = REG, `func` = shift, `ND` = 2400, `NG` = 300, `noise` = 0.05, `minPath` = false, `rSeed` = 123

8 UOSLib-learn: IRMA, `variant` = 0, `stiff` = 0.5

9 UOSLib-learn: IRMA, `variant` = 1, `stiff` = 0.0, `growth` = 0.01

10 UOSLib-learn: IRMA, `variant` = 2, `stiff` = 0.01, `growth` = 1.02

11 UOSLib-learn: IRMA, `variant` = 3, `stiff` = 0.0, `growth` = 200, `maxstiff` = {2.2, 0.5}

Table 14: Cumulative loss for the first target function and after relearning for different growth functions of the stiffness.

Growth function	$l_c(1200)$	$l_c(2400)$
Fixed	7.48	23.81
Additive	6.58	72.94
Multiplicative	**6.54**	437.50
Sigmoidal (high)	6.72	28.28
Sigmoidal (low)	7.21	**23.54**

For the first 1200 examples, the upper left graph shows a bigger slope for the fixed stiffness than for all growing variants. The best performance is achieved by a multiplicative growth with $l_c(1200) = 6.54$ and nearly the same for an additive growth with $l_c(1200) = 6.58$ as they increase the robustness to noise the most. The two different maximal stiffnesses of the sigmoidal growth either just improve the initial learning (low), yielding only a slight improvement, or additionally increase robustness to noise (high) with results comparable to the ever increasing growth functions.

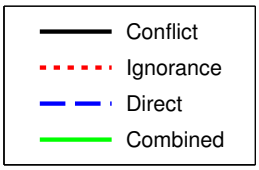
But the multiplicative and additive growth lead to a stiffness allowing for nearly no adaptation after some time, which is not suited for non-stationary environments. Consequently, they are not able to relearn when the target function shifts and the cumulative loss increases drastically. The right plots of Fig. 46 show the second half of the learning sequence, now with logarithmic scale due to highly different results. Here, the low fixed stiffness again enables to adapt quickly to the change and gets better results. The sigmoidal growth allows for a continuous adaptation as well. With a high maximal stiffness, the final performance after the shift occurred is slightly worse than with the fixed stiffness (see also Table 14). But, with a low maximal stiffness it gets the best result at the end of the complete sequence as well as an improvement for the first half of examples, but the improvement is only low.

So, the sigmoidal growth function combines the advantages. It increases quickly to stabilize against noise after the initial learning phase is done and its growth is bounded to enable a continuous adaptation throughout longer sequences of examples. Depending on the maximal stiffness a trade-off between improved robustness to noise and higher adaptivity can be achieved but always with the advantage of an improved initial learning. Consequently, an increasing stiffness is especially beneficial in presence of noise and if the initial learning requires a faster adaptation, i. e. a lower stiffness, than it is necessary

later on to adapt to changes. Yet, it is not easy to find an adequate trade-off for a given problem a priori.

B.3 DETAILED RESULTS OF TRUST ESTIMATION

In the main part of the trust estimation investigation in Section 3.4.2 the results were reduced to their characteristic properties for ease of analysis. In the following, the complete resulting progress of the trust weighted mean squared error (87) and the average trust (88) for the different disturbance types are presented. The notation of all plots is the same according to:



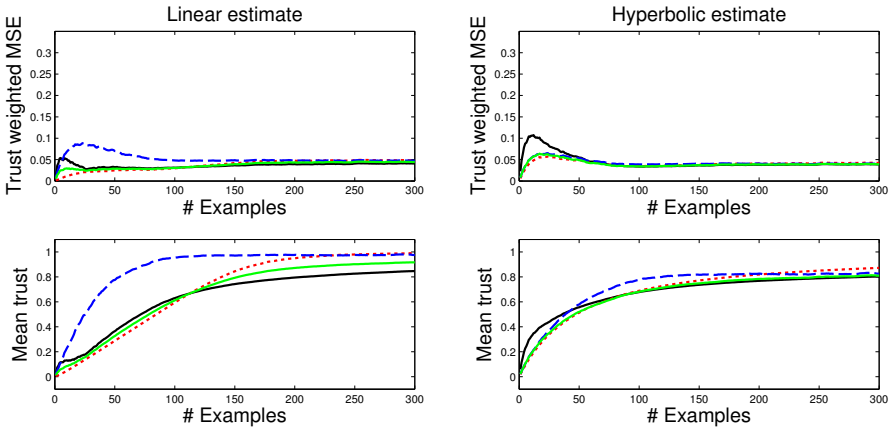


Figure 47: Trust weighted mean squared error (top) and mean trust (bottom) with linear trust estimation (left) and hyperbolic trust estimation (right) in case of no disturbance.

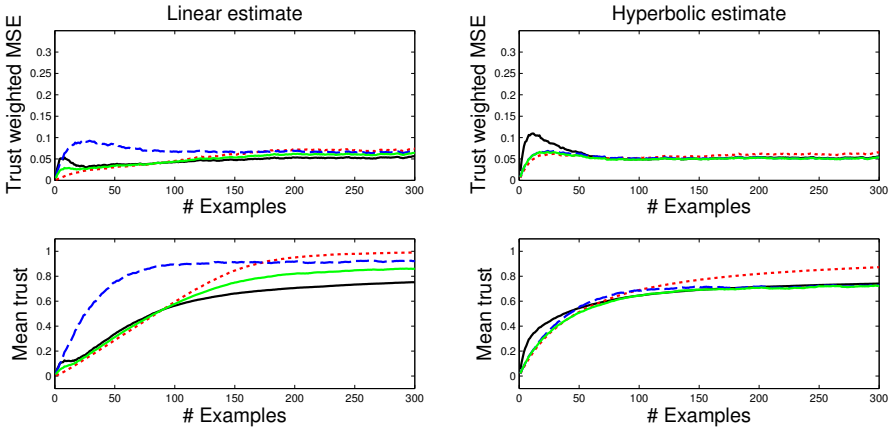


Figure 48: Trust weighted mean squared error (top) and mean trust (bottom) with linear trust estimation (left) and hyperbolic trust estimation (right) in case of input disturbance.

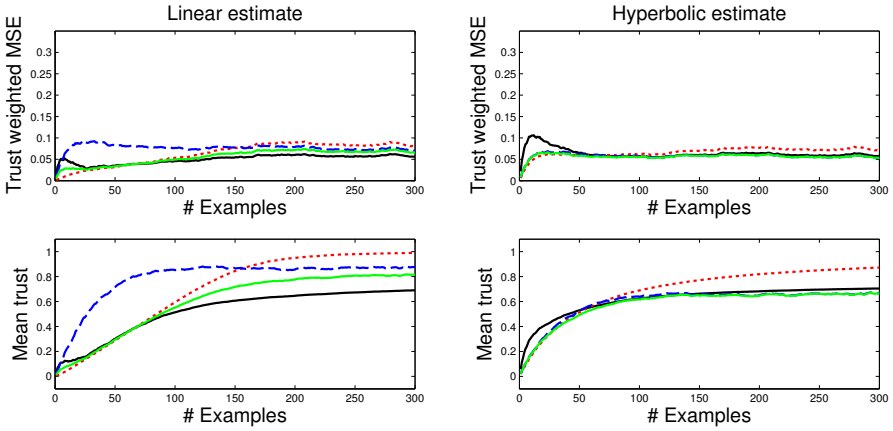


Figure 49: Trust weighted mean squared error (top) and mean trust (bottom) with linear trust estimation (left) and hyperbolic trust estimation (right) in case of output disturbance.

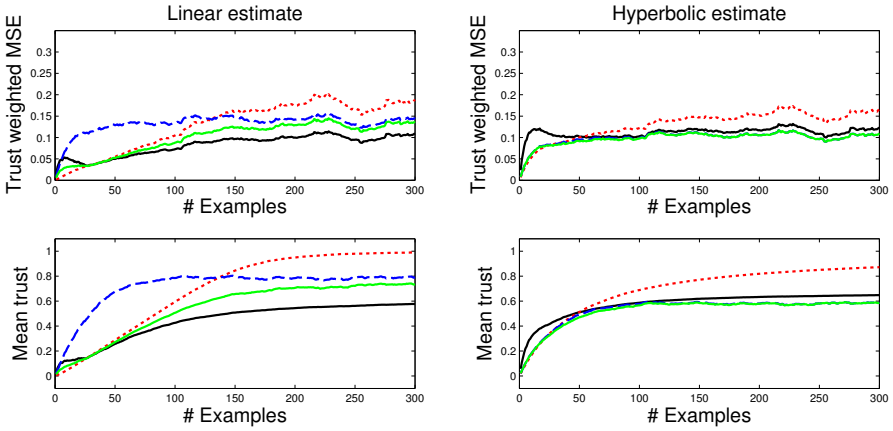


Figure 50: Trust weighted mean squared error (top) and mean trust (bottom) with linear trust estimation (left) and hyperbolic trust estimation (right) in case of unobserved variables.

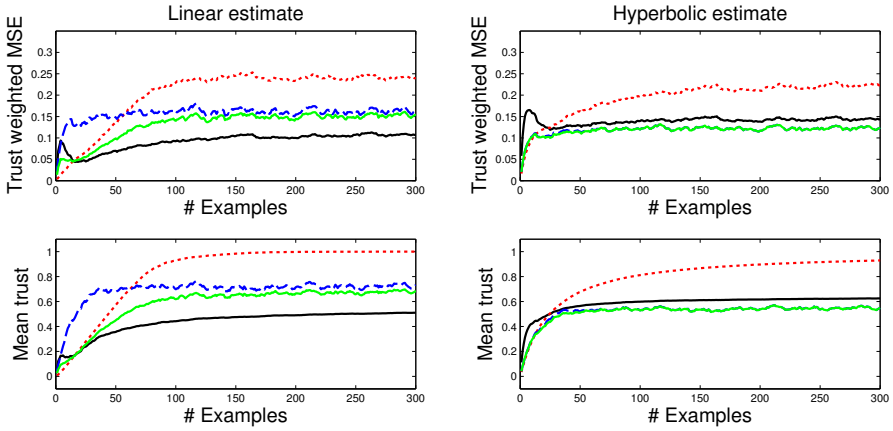


Figure 51: Trust weighted mean squared error (top) and mean trust (bottom) with linear trust estimation (left) and hyperbolic trust estimation (right) in case of inexact approximation.

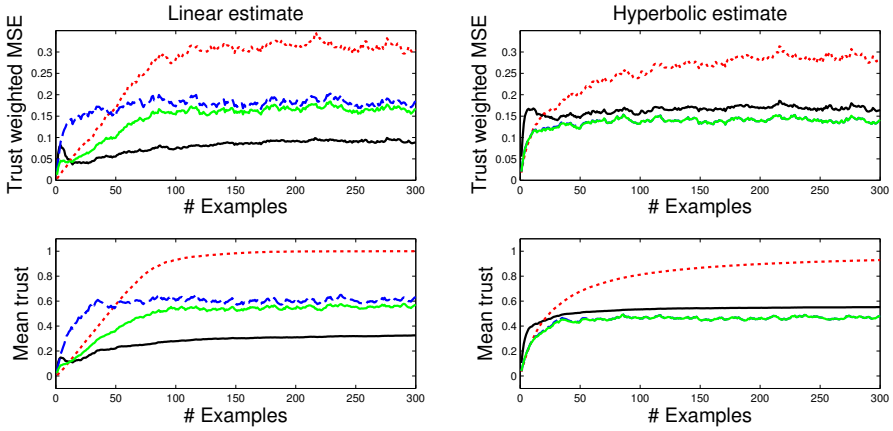


Figure 52: Trust weighted mean squared error (top) and mean trust (bottom) with linear trust estimation (left) and hyperbolic trust estimation (right) in case of all disturbances together.

C

GLOSSARY

ADAPTATION
: part of the learning process, i. e. due to a single example

ALEATORIC UNCERTAINTY
: uncertainty due to variability of the knowledge

BASIS FUNCTION
: part of the model structure, i. e. a fixed transformation from input space to one feature

BATCH LEARNING
: learning based on an (finite) set of examples

CLASSIFICATION
: estimation of an input-output relation with discrete non-ordinal output values

EPISTEMIC UNCERTAINTY
: uncertainty due to missing knowledge

FATAL FORGETTING
: change of the input-output relation apart from a presented example due to adaptation to that example

FEATURE SPACE
: space in which the examples, transformed by the basis functions, are represented linearly in the parameters

FIRST ORDER LEARNING
: direct update of the parameter vector without additional information

HYPER-PARAMETER
: user defined parameter setting up an algorithm, i. e. a parameter that is not learned

HYPOTHESIS
: one input-output relation learned from data

INPUT / INDEPENDENT VARIABLE
: input interface of the learning system

INSTANCE
: concrete value of the inputs

LABEL
: concrete value of the output

LEARNING
: overall process of forming a hypothesis from data

LIP MODEL
: model structure that is linear in the parameters

LOSS
: rating of an erroneous prediction

MACHINE LEARNING
: construction and study of systems that can learn from data

MODEL STRUCTURE
: fixed transformation from input to feature space

NON-STATIONARY
: changing input-output relation throughout learning

ON-LINE LEARNING
: learning based on a (infinite) sequence of examples

OUTPUT / DEPENDENT VARIABLE
: output interface of the learning system

OVERFITTING
: bad generalization due to a model describing only the examples instead of the underlying relation

PARAMETER VECTOR
: hypothesis of a model structure

PREDICTION
: assigned label of the learning system for an instance

REGRESSION
: estimation of an input-output relation with continuous ordinal output values

RELIABILITY
: ensuring that predictions conform with presented examples, i. e. by conforming to recall and a compliant generalization

ROBUSTNESS
: ensuring only slight changes of the output distribution for a slightly altered input distribution

SECOND ORDER LEARNING
: adaptive update of the parameter vector based on condensed information of past examples

TRAINING EXAMPLE
: representative combination of an instance and a label

TRUST LEVEL
: concrete value of a trust signal

TRUST SIGNAL
: meta-information to a normal signal reflecting the trustworthiness

BIBLIOGRAPHY

[1] W. Brockmann, A. Buschermoehle, and J. Huelsmann. A generic concept to increase the robustness of embedded systems by trust management. In *Proc. Int. Conf on Systems Man and Cybernetics*, pages 2037–2044. IEEE Press, 2010.

[2] W. Brockmann, A. Buschermoehle, J. Huelsmann, and N. Rosemann. *Trust Management – Handling Uncertainties in Embedded Systems*, pages 589–591. Autonomic Systems. Springer, 2011.

[3] W. Brockmann, A. Buschermoehle, and J. Schoenke. COBRA - a generic architecture for robust treatment of uncertain information. In *INFORMATIK 2013: Informatik angepasst an Mensch, Organisation und Umwelt*, pages 2727–2741. GI, Bonner Köllen Verlag, 2013.

[4] A. Buschermoehle and W. Brockmann. Stable on-line learning with optimized local learning, but minimal change of the global output. In *Proc. Int. Conf. on Machine Learning and Applications*, pages 21–27. IEEE Press, 2013.

[5] A. Buschermoehle and W. Brockmann. On-line learning with minimized change of the global mapping – optimized local learning by incremental risk minimization. *Evolving Systems*, 2014. (in press).

[6] A. Buschermoehle and W. Brockmann. Reliable localized online learning in non-stationary environments. In *Proc. Int. Conf. on Evolving and Adaptive Intelligent Systems*, pages 1–7. IEEE Press, 2014.

[7] A. Buschermoehle, J. Huelsmann, and W. Brockmann. A structured view on sources of uncertainty in supervised learning. In *Proc. Int. Conf. on Scalable Uncertainty Management*, volume 7520, pages 566–573. Springer, 2012.

[8] A. Buschermoehle, J. Huelsmann, and W. Brockmann. UOSLib – a library for analysis of online-learning algorithms. In *Proc. Workshop on Computational Intelligence*, pages 355–369. KIT Scientific Publishing, 2013.

[9] A. Buschermoehle, N. Rosemann, and W. Brockmann. Stable classification in enviroments with varying degrees of uncertainty. In *Proc. Int. Conf on Computational Intelligence for Modelling, Control and Automation*, pages 447–452. IEEE press, 2008.

[10] A. Buschermoehle, J. Schoenke, and W. Brockmann. Trusted learner: An improved algorithm for trusted incremental function approximation. In *Proc. Int. Symp. on Computational Intelligence in Dynamic and Uncertain Environments*, pages 16–24. IEEE Press, 2011.

[11] A. Buschermoehle, J. Schoenke, and W. Brockmann. Uncertainty and trust estimation in incrementally learning function approximation. In *Proc. Int. Conf on Information Processing and Management of Uncertainty*, pages 32–41. Springer, 2012.

[12] A. Buschermoehle, J. Schoenke, N. Rosemann, and W. Brockmann. The incremental risk functional: Basics of a novel incremental learning approach. In *Proc. Int. Conf. on Systems Man and Cybernetics*, pages 1500–1505. IEEE Press, 2013.

[13] J. Huelsmann, A. Buschermoehle, and W. Brockmann. Incorporating dynamic uncertainties into a fuzzy classifier. In *Proc. Int. Conf. of the European Society for Fuzzy Logic and Technology*, pages 388–395. EUSFLAT, Atlantis Press, 2011.

[14] N. Rosemann, A. Buschermoehle, and W. Brockmann. Beschleunigung der Selbstoptimierung durch Selbstsimulation. In *Proc. Workshop on Computational Intelligence*, pages 114–128. KIT Scientific Publishing, 2009.

[15] H. K. Alfares and M. Nazeeruddin. Electric load forecasting: Literature survey and classification of methods. *Int. J. of Systems Science*, 33(1):23–34, 2002.

[16] K. Bache and M. Lichman. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2013.

[17] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *Trans. on Information Theory*, 39(3):930–945, 1993.

[18] R. Battiti. First- and second-order methods for learning: Between steepest descent and Newton's method. *Neural Computation*, 4(2):141–166, 1992.

[19] Y. Ben-Haim. *Info-Gap Decision Theory: Decisions Under Severe Uncertainty (2nd edition)*. Academic Press, Oxford, UK, 2006.

[20] H. Block. The perceptron: A model for brain functioning. *Reviews of Modern Physics*, 34(1):123–135, 1962.

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

[22] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proc. Workshop on Computational Learning Theory*, pages 144–152. ACM, 1992.

[23] Z. Bosnić and I. Kononenko. Comparison of approaches for estimating reliability of individual regression predictions. *Data & Knowledge Engineering*, 67(3):504–516, 2008.

[24] Z. Bosnić and I. Kononenko. Estimation of individual prediction reliability using the local sensitivity analysis. *Applied Intelligence*, 29(3):187–203, 2008.

[25] Z. Bosnić and I. Kononenko. An overview of advances in reliability estimation of individual predictions in machine learning. *Intelligent Data Analysis*, 13(2):385–401, 2009.

[26] L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967.

[27] S. Briesemeister, J. Rahnenführer, and O. Kohlbacher. No longer confidential: estimating the confidence of individual regression predictions. *PLoS ONE*, 7(11):e48723, 2012.

[28] W. Brockmann, E. Maehle, K.-E. Grosspietsch, N. Rosemann, and B. Jakimovski. *ORCA: An Organic Robot Control Architecture*, pages 385–398. Autonomic Systems. Springer, 2011.

[29] W. Brockmann, E. Maehle, and F. Mösch. Organic fault-tolerant control architecture for robotic applications. In *4th IARP/IEEE-RAS/EURON Workshop on Dependable Robots in Human Environments*, 2005.

[30] E. F. Camacho and C. B. Alba. *Model predictive control*. Springer, London, UK, 2 edition, 2004.

[31] O. Castillo and P. Melin. *Type-2 Fuzzy Logic: Theory and Applications*. Springer, Berlin, Germany, 2008.

[32] N. Cesa-Bianchi. *Prediction, learning, and games*. Cambridge University Press, Cambridge, UK, 2006.

[33] N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order perceptron algorithm. *J. on Computing*, 34(3):640–668, 2005.

[34] C. Chatfield. *Time-series forecasting*. Chapman and Hall/CRC Press, Boca Raton, FL, USA, 2000.

[35] M.-S. Chen and J.-Y. Yen. Application of the least squares algorithm to the observer design for linear time-varying systems. *Trans. on Automatic Control*, 44(9):1742–1745, 1999.

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

[37] K. Crammer, M. D. Fern, and O. Pereira. Exact convex confidence-weighted learning. *Advances in Neural Information Processing Systems*, 21:345–352, 2008.

[38] K. Crammer, A. Kulesza, M. Dredze, et al. Adaptive regularization of weight vectors. *Advances in Neural Information Processing Systems*, 22:414–422, 2009.

[39] K. Crammer and D. D. Lee. Learning via gaussian herding. *Advances in Neural Information Processing Systems*, 23:451–459, 2010.

[40] M. Crowder, A. Kimber, R. Smith, and T. Sweeting. *Statistical concepts in reliability*, pages 1–11. Springer, 1991.

[41] L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

[42] E. De Vito, L. Rosasco, A. Caponnetto, U. De Giovannini, and F. Odone. Learning from examples as an inverse problem. *J. of Machine Learning Research*, 6:883–904, 2006.

[43] A. Dempster and G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, NJ, USA, 1976.

[44] T. Denoeux. Function approximation in the framework of evidence theory: A connectionist approach. In *Proc. Int. Conf. on Neural Networks*, volume 1, pages 199–203. IEEE Press, 1997.

[45] P. Diamond and H. Tanaka. *Fuzzy Regression Analysis*, pages 349–387. The Handbooks of Fuzzy Sets Series. Springer, 1998.

[46] N. Draper and H. Smith. *Applied regression analysis*. Wiley, Hoboken, NJ, USA, 3 edition, 1998.

[47] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *Proc. Int. Conf. on Machine Learning*, pages 264–271. ACM, 2008.

[48] J. Farrell and M. Polycarpou. *Adaptive approximation based control*. Wiley-Blackwell, Hoboken, NJ, USA, 2006.

[49] E. Feinberg and D. Genethliou. *Load Forecasting*, pages 269–285. Power Electronics and Power Systems. Springer, 2005.

[50] C. Ferrell. Failure recognition and fault tolerance of an autonomous robot. *Adaptive Behavior*, 2(4):375–398, 1994.

[51] A. Gammerman and V. Vovk. Hedging predictions in machine learning the second computer journal lecture. *The Computer Journal*, 50(2):151–163, 2007.

[52] C. Gentile. The robustness of the p-norm algorithms. *Machine Learning*, 53(3):265–299, 2003.

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

[54] W. Haerdle. *Applied nonparametric regression*. Cambridge University Press, Cambridge, UK, 1990.

[55] H. Hahn, S. Meyer-Nieberg, and S. Pickl. Electric load forecasting methods: Tools for decision making. *Europ. J. of Operational Research*, 199(3):902–907, 2009.

[56] F. R. Hampel. A general qualitative definition of robustness. *Ann. of Mathematical Statistics*, 42(6):1887–1896, 1971.

[57] E. Hüllermeier. Uncertainty in clustering and classification. In *Proc. Int. Conf. on Scalable Uncertainty Management*, pages 16–19. Springer, 2010.

[58] S. C. Hora. Aleatory and epistemic uncertainty in probability elicitation with an example from hazardous waste management. *Reliability Engineering & System Safety*, 54(2):217–223, 1996.

[59] W. A. Huber. Ignorance is not probability. *Risk analysis*, 30(3):371–376, 2010.

[60] E. Hüllermeier. Fuzzy methods in machine learning and data mining: Status and prospects. *Fuzzy Sets and Systems*, 156(3):387–406, 2005.

[61] E. Hüllermeier and K. Brinker. Learning valued preference structures for solving classification problems. *Fuzzy Sets and Systems*, 159(18):2337–2352, 2008.

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

[63] Kiel Netz GmbH. Stadtwerke Kiel Netzlasten (retrieved march 11, 2013). `http://www.swkiel-netz.de/index.php?id=swkielnetzgmbh__stromnetz__netzlasten`, 2013.

[64] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

[65] J. Kivinen, M. K. Warmuth, and B. Hassibi. The p-norm generalization of the LMS algorithm for adaptive filtering. *Trans. Signal Processing*, 54(5):1782–1793, 2006.

[66] D. Klaua. Ein Ansatz zur mehrwertigen Mengenlehre. *Mathematische Nachrichten*, 33(5-6):273–296, 1967.

[67] K. Kleinlützum, W. Brockmann, and N. Rosemann. Modellierung von Anomalien in einer modularen Roboter-Steuerung. In *Proc. Conf. Autonome Mobile Systeme*, pages 89–95. Springer, 2007.

[68] G. Klir. *Uncertainty and Information: Foundations of Generalized Information Theory*. Wiley, Hoboken, NJ, USA, 2005.

[69] A. N. Kolmogorov. Grundbegriffe der Wahrscheinlichkeitsrechnung. *Ergebnisse der Mathematik und ihrer Grenzgebiete*, 3, 1933.

[70] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. of Mathematical Statistics*, 22(1):79–86, 1951.

[71] J. Leonard, M. Kramer, and L. Ungar. Using radial basis functions to approximate a function and its error bounds. *Trans. on Neural Networks*, 3(4):624–627, 1992.

[72] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. In *Proc. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 298–306. ACM, 1996.

[73] L. Li, M. L. Littman, T. J. Walsh, and A. L. Strehl. Knows what it knows: a framework for self-aware learning. *Machine learning*, 82(3):399–443, 2011.

[74] Y. Li and P. M. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46(1):361–387, 2002.

[75] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

[76] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. In *Proc. Symp. on the Foundations of Computer Science*, pages 256–261. IEEE Press, 1999.

[77] L. Ljung and T. Soederstroem. *Theory and practice of recursive identification*. MIT Press, Cambridge, MA, USA, 3 edition, 1987.

[78] E. Lughofer. Single-pass active learning with conflict and ignorance. *Evolving Systems*, 3(4):251–271, 2012.

[79] E. Lughofer and C. Guardiola. Applying evolving fuzzy models with adaptive local error bars to on-line fault detection. In *Proc. Int. Workshop on Genetic and Evolving Systems*, pages 35–40. IEEE Press, 2008.

[80] H. B. McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. *J. of Machine Learning Research*, 15:525–533, 2011.

[81] H. B. McMahan and M. Streeter. Adaptive bound optimization for online convex optimization. In *Proc. Int. Conf. on Learning Theory*, pages 244–256. Omnipress, 2010.

[82] M. Mizumoto. Pictorial representations of fuzzy connectives, part i: cases of t-norms, t-conorms and averaging operators. *Fuzzy Sets Systems*, 31(2):217–242, 1989.

[83] M. Mizumoto. Pictorial representations of fuzzy connectives, part ii: cases of compensatory operators and self-dual operators. *Fuzzy Sets Systems*, 32(1):45–79, 1989.

[84] R. E. Moore. *Interval Analysis*, volume 2. Prentice-Hall, Englewood Cliffs, NJ, USA, 1966.

[85] F. Mösch, M. Litza, A. El Sayed Auf, B. Jakimovski, E. Maehle, and W. Brockmann. Organic fault-tolerant controller for the walking robot oscar. In *Architecture of Computing Systems*, pages 31–35. VDE, VDI Verlag, 2007.

[86] S. Mukherjee, R. Rifkin, and T. Poggio. Regression and classification with regularization. In *Nonlinear Estimation and Classification*, volume 171, pages 111–128. Springer, 2003.

[87] National Institute of Standards and Technology. Statistical reference datasets (strd) nonlinear regression (retrieved september 9, 2012). `http://www.itl.nist.gov/div898/strd/`, 2000.

[88] O. Nelles. *Nonlinear system identification: From classical approaches to neural networks and fuzzy models*. Springer, Berlin, Germany, 2001.

[89] D. Nguyen-Tuong, J. R. Peters, and M. Seeger. Local gaussian process regression for real time online model learning. *Advances in Neural Information Processing Systems*, 21:1193–1200, 2008.

[90] A. Novikoff. On convergence proofs on perceptrons. In *Proc. Symp. Mathematical Theory of Automata*, pages 615–622. Wiley, 1962.

[91] E. Parzen et al. On estimation of a probability density function and mode. *Ann. of Mathematical Statistics*, 33(3):1065–1076, 1962.

[92] S. Petit-Renaud and T. Denœux. Handling different forms of uncertainty in regression analysis: A fuzzy belief structure approach. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 340–351. Springer, 1999.

[93] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, 3 edition, 2007.

[94] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, 2006.

[95] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[96] M. E. Salgado, G. C. Goodwin, and R. H. Middleton. Modified least squares algorithm incorporating exponential resetting and forgetting. *Int. J. of Control*, 47(2):477–491, 1988.

[97] T. Schaul, S. Zhang, and Y. LeCun. No more pesky learning rates. In *Proc. Int. Conf. on Machine Learning*, pages 343–351. Omnipress, 2013.

[98] J. H. Schoenke. Online-evaluation of learning function approximators in presence of uncertainties (*Online-Auswertung von lernfähigen Funktionsapproximatoren unter Ungewissheiten*). Master's thesis, University of Osnabrück, 2013. Original document in German.

[99] R. Senge, S. Bösner, K. Dembczyński, J. Haasenritter, O. Hirsch, N. Donner-Banzhoff, and E. Hüllermeier. Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty. *Information Sciences*, 255(0):16–29, 2014.

[100] E. H. Shortliffe and B. G. Buchanan. A model of inexact reasoning in medicine. *Mathematical Biosciences*, 23(3):351–379, 1975.

[101] J.-J. E. Slotine, W. Li, et al. *Applied nonlinear control*, volume 199. Prentice-Hall, Englewood Cliffs, NJ, USA, 1991.

[102] J. M. Steele. *The Cauchy-Schwarz master class: An introduction to the art of mathematical inequalities*. Cambridge University Press, Cambridge, UK, 2004.

[103] A. L. Strehl and M. L. Littman. Online linear regression and its application to model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 20:1417–1424, 2007.

[104] A. N. Tikhonov. On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pages 195–198, 1943.

[105] N. Vaits and K. Crammer. Re-adapting the regularization of weights for non-stationary regression. In *Algorithmic Learning Theory*, volume 6925, pages 114–128. Springer, 2011.

[106] N. Vaits, E. Moroshko, and K. Crammer. Second-order non-stationary online learning for regression. *arXiv preprint arXiv:1303.0140*, 2013.

[107] V. Vapnik. *The nature of statistical learning theory*. Springer, Berlin, Germany, 2000.

[108] M. Vogt, N. Müller, and R. Isermann. On-line adaptation of grid-based look-up tables using a fast linear regression technique. *J. of Dynamic Systems, Measurement and Control*, 126(4):732–739, 2004.

[109] V. Vovk. On-line confidence machines are well-calibrated. In *Proc. Symp. on Foundations of Computer Science*, pages 187–196. IEEE Press, 2002.

[110] V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic learning in a random world*. Springer, Berlin, Germany, 2005.

[111] P. Walley. Towards a unified theory of imprecise probability. *Int. J. of Approximate Reasoning*, 24(2):125–148, 2000.

[112] J. Wang, P. Zhao, and S. C. Hoi. Exact soft confidence-weighted learning. In *Proc. Int. Conf. on Machine Learning*, pages 121–128. Omnipress, 2012.

[113] K. Weichselberger. The theory of interval-probability as a unifying concept for uncertainty. *Int. J. of Approximate Reasoning*, 24(2):149–170, 2000.

[114] A. S. Weigend and D. A. Nix. Predictions with confidence intervals (local error bars). In *Proc. Int. Conf. on Neural Information Processing*, pages 847–852, 1994.

[115] B. Widrow and M. E. Hoff. Adaptive switching circuits. *IRE WESCON Convention record*, 4:96–104, 1960.

[116] R. Willink. *Measurement Uncertainty and Probability*. Cambridge University Press, Cambridge, UK, 2013.

[117] D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

[118] D. H. Wolpert. The supervised learning no-free-lunch theorems. In *Proc. Online World Conf. on Soft Computing in Industrial Applications*, pages 25–42. Springer, 2002.

[119] L. Yang, R. Jin, and J. Ye. Online learning by ellipsoid method. In *Proc. Int. Conf. on Machine Learning*, pages 1153–1160. ACM, 2009.

[120] I.-C. Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808, 1998.

[121] Y. Ying and M. Pontil. Online gradient descent learning algorithms. *Found. of Computational Mathematics*, 8(5):561–596, 2008.

[122] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

DECLARATION OF ORIGINALITY

I hereby declare that this thesis was composed by and originated entirely from me without undue help or use of other than the specified resources. To the best of my knowledge and belief this thesis contains no material previously published or written by any other person. Information derived from the published and unpublished work of others has been acknowledged in the text and references are given in the list of sources. This thesis has not been previously, or concurrently, accepted for any other degree in any university.

In parts of the evaluation and selection of material, the persons listed below helped against payment as student assistants or free of charge:

- Jens Hülsmann
- Dr. Nils Rosemann
- Jan Schoenke

No further persons were involved in creating the material of this thesis. In particular, I did not take any paid help of mediation and counseling services.

Osnabrück, June 26, 2014

Andreas Buschermöhle