Herausgeber

F. HOFFMANN
E. HÜLLERMEIER
R. MIKUT

Dortmund | 23. – 24. November 2017

PROCEEDINGS **27. WORKSHOP** COMPUTATIONAL INTELLIGENCE

KIT Scientific Publishing

F. Hoffmann, E. Hüllermeier, R. Mikut (Hrsg.)

Proceedings. 27. Workshop Computational Intelligence

Dortmund, 23. – 24. November 2017

PROCEEDINGS **27. WORKSHOP** COMPUTATIONAL INTELLIGENCE

Dortmund, 23. – 24. November 2017

Herausgegeben von
F. Hoffmann
E. Hüllermeier
R. Mikut

KIT Scientific Publishing

Impressum

Scientific Publishing

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

KIT Scientific Publishing is a registered trademark
of Karlsruhe Institute of Technology.
Reprint using the book cover is not allowed.

www.ksp.kit.edu

Print on Demand 2017 – Gedruckt auf FSC-zertifiziertem Papier

ISBN 978-3-7315-0726-0
DOI 10.5445/KSP/1000074341

Inhaltsverzeichnis

Optimizing the Structure of Nested Dichotomies: A Comparison of Two Heuristics

Vitalik Melnikov[1], Eyke Hüllermeier[2]

Department of Computer Science
Paderborn University, Germany
E-Mail: [1]melnikov@mail.upb.de, [2]eyke@upb.de

Abstract

In machine learning, nested dichotomies are used to decompose a multi-class classification problem into a set of binary problems. The performance of a nested dichotomy strongly depends on the structure of such a decomposition. In this paper, we compare the random-pair heuristic, a state-of-the-art approach for optimizing the structure, with an extremely simple alternative: Leveraging a procedure for uniform sampling of dichotomies, the Best-of-K heuristic picks the (presumably) best among K randomly generated dichotomies. Interestingly, Best-of-K turns out to be highly competitive, and outperforms the random-pair heuristic in the case of simple base learners such as logistic regression.

1 Introduction

Nested dichotomies are known as models for polychotomous data in statistics and used as classifiers for multi-class problems in machine learning [4]. Based on a recursive binary partitioning of the set of classes, nested dichotomies reduce the original multi-class problem to a set of binary problems, for which any (probabilistic) binary classifier can be used. For example, the dichotomy shown in Fig. 1 decomposes a problem with four classes into three binary problems: The first classifier (C_1) is supposed to separate class 3 from the meta-class $\{1, 2, 4\}$, i.e., the union of classes 1, 2, and 4; likewise, the second classifier separates classes $\{2, 4\}$ from 1, and the third classifier the classes 2 and 4.

Once the hierarchy of classifiers required by a nested dichotomy have been trained, a new instance x can be classified in a straightforward way,

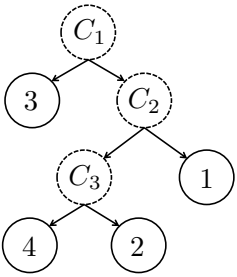


Figure 1: Example of a nested dochotomy.

namely by submitting x to the root of the dichotomy and propagating it to one of the leaf nodes, based on the decisions of the classifiers along the corresponding path. As an appealing property, we note that nested dichotomies allow for producing probability estimates (instead of hard class assignments) in a very natural way, provided each classifier predicts suitable conditional class probabilities at the inner nodes. In that case, the probability of any class y is simply given by the product of conditional probabilities on the path from the root to the leaf node marked with y.

In practice, nested dichotomies have been shown to yield superb predictive accuracy [4, 6, 8]. Yet, the performance of the multi-class classifier eventually produced may strongly depend on the structure of the dichotomy. In fact, the structure of a dichotomy specifies the subset of all binary problems that need to be solved, and some of them might be much more difficult than others. This is illustrated in Fig. 2, where the distribution of predictive accuracies (on the test data) of nested dichotomies is shown for the pendigits dataset. As can be seen, the variance is higher if logistic regression is used as a base learner, and smaller with decision trees. This observation can be explained by the fact that the latter is much more flexible than the former: A decision tree is a complex, highly nonlinear model, which can compensate a suboptimal structure much better than a simple linear model as fit by logistic regression (but of course also comes with a higher danger of poor generalization due to overfitting); or, stated differently, the choice of a suitable structure is much more critical when using simple models such as linear discriminants.

Consequently, finding a suitable structure is an important prerequisite for successful learning with nested dichotomies. Since the number of candidate structures is huge (double factorial in the number of classes),

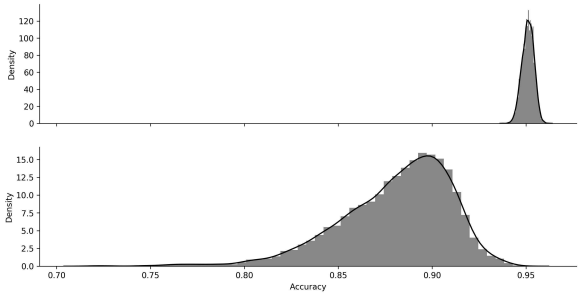


Figure 2: Gaussian kernel density estimation of accuracy distribution for the pendigits dataset based on a random sample of 10000 nested dichotomies, using decision tree (CART, in the top plot) and logistic regression (in the bottom plot) as base learner.

several heuristic methods have been proposed for constructing dichotomies specifically tailored for the data at hand [2, 6, 3]. Although some heuristics are designed to select an optimal *subset* of nested dichotomies, to be used as an ensemble of classifiers, we focus on finding a *single* dichotomy in this paper (for example because an ensemble is not desirable, due to reasons of complexity or interpretability).

According to recent empirical studies, the current state of the art is the *random-pair* selection heuristic (RPND) proposed by Leathart et al. [6]. At each inner node of a nested dichotomy, this approach obtains a split of the classes $\mathcal{Y}$ associated with that node into two meta-classes as follows: Two classes $Y_i, Y_j \in \mathcal{Y}$ are selected (uniformly) at random, and the base learner is used to train a classifier $C_{i,j}$ discriminating these two classes. Both classes define the seed for a meta-class $\mathcal{Y}_i$ and $\mathcal{Y}_j$, respectively, and each remaining class $Y \in \mathcal{Y} \setminus \{Y_i, Y_j\}$ is added to one of these meta-classes, depending on whether $C_{i,j}$ assigns the majority of instances of Y to $\mathcal{Y}_i$ or $\mathcal{Y}_j$. Once the complete structure of the dichotomy has been determined, the classifiers at the inner nodes are trained on the corresponding splits.

In this paper, we reconsider the random-pair heuristic, challenging it with the arguably most simple heuristic one may image: generating K dichotomies at random and selecting the (presumably) best one. In Section 4, we demonstrate that this simple Best-of-K heuristic is highly competitive and, depending on the value K, even able to outperform the random-pair heuristic. As a side product, the paper makes another

interesting contribution, namely an efficient algorithm for sampling nested dichotomies uniformly at random. Although sampling procedures have already been used in several papers [4, 6, 2], an explicit algorithm has never been provided; interestingly, a naive sampling procedure will yield a non-uniform distribution.

2 Uniform random sampling of nested dichotomies

On the structural side, there is a one-to-one correspondence between nested dichotomies and binary trees. Since each node in a nested dichotomy has either none or exactly two children, every nested dichotomy is a (rooted) full binary tree. In addition, each leaf node in a nested dichotomy is labeled with the corresponding class. This labeling uniquely determines the dichotomies at all inner nodes. The problem of generating a nested dichotomy can therefore be divided into two steps: (i) generation of a rooted terminally labeled full binary tree and (ii) propagation of leaf labels towards the root in order to determine the dichotomies at the inner nodes.

For the problem (i), several strategies were proposed in the literature [9, 5]. Here, we provide a two-step procedure for uniform random sampling of rooted terminally labeled full binary trees as suggested by Furnas [5]:

1. Generation of an unrooted terminally labeled full binary tree T_c. Given is a set of c terminal nodes (leafs).
   a) Create a doublet "tree" T_2 by connecting nodes 1 and 2 by a single edge.
   b) Until all c terminal nodes are connected to the tree, proceed with the following random augmentation:
      i. Given a tree T_k on $k < c$ terminal nodes, select an edge of T_k uniformly at random.
      ii. On this edge, a new internal node of degree 3 is added and the $(k+1)$st terminal node is connected. The result is a binary tree T_{k+1} on $k+1$ terminal nodes.
2. Transformation of T_c into a rooted tree.
   a) Choose an edge of T_c randomly with uniform probability.
   b) Introduce a new root node of degree 2 on this edge.

Furnas proved this procedure to provide a uniform random sampling of rooted terminally labeled full binary trees. The time complexity of the first step is linear in the number of classes c. The second step can be implemented in constant time, leading to an overall time complexity of $O(c)$.

For the label propagation problem (ii), we suggest the following solution. Every leaf node is labeled with the number $b_{c_i} = 2^{c_i}$, where c_i is the corresponding class in the original problem. The dichotomies at the inner nodes are encoded with a pair of numbers $[d_l, d_r]$, denoting the sum of all leaf labels in the left and right subtree, respectively. Since all leaf nodes are encoded uniquely (with only a single '1' in the corresponding bit string), the dichotomy encodings are also unique. To propagate the leaf labels, we traverse the nested dichotomy in postorder and encode every inner node with the sum of the dichotomies of its child nodes $[d_{l_l} + d_{l_r}, d_{r_l} + d_{r_r}]$. Since the complexity of tree traversal is linear in the number of nodes and the summation, and generating the leaf labels is nearly constant for a usual number of classes[1], step (ii) has an overall time complexity of $O(c)$. Using this approach, a single nested dichotomy can be sampled uniformly with the time complexity $O(c)$, i.e., linear in the number of classes in the original problem.

3 The Best-of-K Heuristic

Equipped with the sampling procedure for nested dichotomies from the previous section, the algorithm for the Best-of-K heuristic is quite straightforward. It is parametrized by the number K of nested dichotomies to be generated and evaluated. Given a (training) dataset with c classes and a base learner, Best-of-K consists of three steps:

1. Sample K nested dichotomies on c terminal nodes uniformly at random.
2. Train these models on the given data with the provided base learner.
3. Select the best performing model based on the validation on the training data (e.g., with the smallest training error).

[1]The length of the encoding number (as a bit string) is at most c. Both encoding operations (bit shift and summation) will have roughly constant time for up to several hundred classes.

Several remarks on this approach are in order. First, since the sampling procedure only depends on the number of classes c, and not on the data itself, it can be carried out in a preprocessing step, thereby reducing the overall execution time. Second, the training in the second step can be done independently for each nested dichotomy. Thus, Best-of-K is naturally implemented in a parallel way. If K cores are available, a speedup factor close to K can be achieved in comparison to a standard (sequential) implementation (the runtime is the maximal training time of a single nested dichotomy). Third, while being computationally cheap, the selection of the best performing model based on the training error is arguably not optimal—a better selection could probably be made on the basis of a validation error. However, our experiments in the next section suggest this strategy to be good enough in general.

4 Empirical Study

We compare the Best-of-K heuristic with the state of the art RPND on 15 multi-class datasets (Table 1) from the UCI repository [1]. On every dataset, we perform the following evaluation: For a given parameter $K \in \{1, 5, 10, 20, 50, 100\}$, we generate a single nested dichotomy using the Best-of-K heuristic and a single nested dichotomy using RPND. Both models are tested with two base learners: decision tree (CART) and logistic regression from the scikit-learn framework (ver. 0.19) [7]. The hyper-parameters of the base learners are set to default values, except for the minimal decrease of the impurity in CART, which is set to .0001 to prevent overfitting. Every generated nested dichotomy is trained multiple times by using 10-fold cross-validation, and the mean predictive accuracy is stored. Since both heuristics are randomized, we repeat this procedure 50 times for every dataset to stabilize the results. The mean and the standard deviation of predictive accuracy over these runs are given in the Table 2.

In Fig. 3, the difference $A_{diff} = A_{BoK} - A_{RPND}$ in mean accuracy is shown. Unsurprisingly, the performance of Best-of-K increases for higher values of K. For $K > 5$, Best-of-K outperforms the RPND heuristic on most of the datasets if logistic regression is used as the base learner. For example, for $K = 5$ and logistic regression as base learner, the mean time ratio over all datasets is 0.58 and the mean accuracy difference is 0.015. In the case of CART as a base learner, both heuristics perform comparable.

Table 1: The datasets used in the study. For the datasets krkopt and vowel, a one-hot encoding has been used for categorical features.

dataset	Classes	Features	Instances
LED24	10	25	5000
zoo	7	18	101
krkopt-bin	18	7	28056
segment	7	20	2310
mfeat-morphological	10	7	2000
mfeat-factors	10	217	2000
mfeat-fourier	10	77	2000
mfeat-karhunen	10	65	2000
mfeat-pixel	10	241	2000
letter	26	17	20000
optdigits	10	65	5620
page-blocks	5	11	5473
pendigits	10	17	10992
vowel-bin	11	14	990
yeast	10	9	1484

This seems plausible for the reason already mentioned: Since (unpruned) decision trees are able to generate complex models (with a tendency to overfit), they can compensate for a possibly suboptimal structure of a dichotomy. The small decrease in the performance of such models for higher values of K suggests that, at least for some datasets, the selection of a nested dichotomy based on the training error is too optimistic.

In Table 3, the mean and standard deviation of the generation time are given. The time was measured for the Best-of-K heuristic with parallelization, i.e., we assume to have K cores available and measure the maximal generation time for a single nested dichotomy. On a single core CPU, this heuristics would be roughly K times slower. The time comparison is provided in Fig. 4. The mean time ratio is defined as the ratio of mean times for generating a single nested dichotomy with both heuristics: $T = T_{BoK}/T_{RPND}$. Since the RPND heuristic depends on the size of the dataset, this ratio is lower for larger datasets such as letter or krkopt.

Table 2: Mean and standard deviation of the predictive accuracy for (a) CART and (b) logistic regression as the base learner. 'Bo' is the corresponding Best-of-K heuristic.

(a)

dataset	Bo1	Bo5	Bo10	Bo20	Bo50	Bo100	RPND
LED24	$.61 \pm .01$	$.61 \pm .01$	$.61 \pm .01$	$.60 \pm .01$	$.60 \pm .01$	$.60 \pm .02$	$.62 \pm .01$
zoo	$.92 \pm .02$	$.92 \pm .02$	$.92 \pm .02$	$.92 \pm .02$	$.92 \pm .02$	$.92 \pm .02$	$.90 \pm .01$
krkopt-bin	$.71 \pm .02$	$.72 \pm .01$	$.73 \pm .01$	$.74 \pm .01$	$.74 \pm .01$	$.75 \pm .01$	$.70 \pm .01$
segment	$.96 \pm .01$	$.96 \pm .01$	$.96 \pm .00$	$.96 \pm .00$	$.96 \pm .00$	$.96 \pm .00$	$.96 \pm .00$
mfeat-morph.	$.65 \pm .01$	$.65 \pm .01$	$.65 \pm .01$	$.65 \pm .01$	$.65 \pm .01$	$.65 \pm .01$	$.65 \pm .00$
mfeat-factors	$.87 \pm .01$	$.87 \pm .01$	$.87 \pm .01$	$.87 \pm .01$	$.87 \pm .01$	$.87 \pm .01$	$.87 \pm .01$
mfeat-fourier	$.69 \pm .01$	$.69 \pm .01$	$.69 \pm .01$	$.69 \pm .01$	$.69 \pm .01$	$.69 \pm .01$	$.70 \pm .00$
mfeat-karhunen	$.78 \pm .01$	$.78 \pm .01$	$.78 \pm .01$	$.78 \pm .01$	$.78 \pm .02$	$.78 \pm .02$	$.78 \pm .01$
mfeat-pixel	$.86 \pm .01$	$.85 \pm .01$	$.86 \pm .01$	$.85 \pm .01$	$.85 \pm .01$	$.85 \pm .01$	$.86 \pm .01$
letter	$.85 \pm .00$	$.85 \pm .00$	$.85 \pm .00$	$.85 \pm .00$	$.85 \pm .00$	$.85 \pm .00$	$.84 \pm .00$
optdigits	$.89 \pm .01$	$.89 \pm .01$	$.89 \pm .01$	$.89 \pm .01$	$.89 \pm .01$	$.89 \pm .01$	$.89 \pm .00$
page-blocks	$.97 \pm .00$	$.97 \pm .00$	$.97 \pm .00$	$.97 \pm .00$	$.97 \pm .00$	$.97 \pm .00$	$.97 \pm .00$
pendigits	$.95 \pm .00$	$.95 \pm .00$	$.95 \pm .00$	$.95 \pm .00$	$.95 \pm .00$	$.95 \pm .00$	$.95 \pm .00$
vowel-bin	$.76 \pm .01$	$.76 \pm .01$	$.76 \pm .01$	$.76 \pm .01$	$.76 \pm .01$	$.76 \pm .01$	$.77 \pm .01$
yeast	$.52 \pm .01$	$.52 \pm .01$	$.52 \pm .01$	$.51 \pm .01$	$.51 \pm .01$	$.51 \pm .01$	$.51 \pm .01$

(b)

dataset	Bo1	Bo5	Bo10	Bo20	Bo50	Bo100	RPND
LED24	$.71 \pm .02$	$.72 \pm .01$	$.72 \pm .01$	$.72 \pm .01$	$.72 \pm .01$	$.72 \pm .00$	$.72 \pm .00$
zoo	$.91 \pm .02$	$.91 \pm .02$	$.92 \pm .02$	$.91 \pm .02$	$.91 \pm .02$	$.92 \pm .02$	$.91 \pm .01$
krkopt-bin	$.29 \pm .01$	$.31 \pm .01$	$.31 \pm .01$	$.31 \pm .01$	$.32 \pm .01$	$.32 \pm .01$	$.29 \pm .01$
segment	$.89 \pm .04$	$.91 \pm .04$	$.92 \pm .03$	$.93 \pm .03$	$.94 \pm .03$	$.94 \pm .03$	$.86 \pm .01$
mfeat-morph.	$.64 \pm .04$	$.67 \pm .03$	$.68 \pm .02$	$.68 \pm .02$	$.69 \pm .02$	$.69 \pm .01$	$.59 \pm .02$
mfeat-factors	$.96 \pm .01$	$.96 \pm .01$	$.96 \pm .01$	$.96 \pm .01$	$.96 \pm .01$	$.96 \pm .01$	$.96 \pm .00$
mfeat-fourier	$.79 \pm .01$	$.79 \pm .01$	$.80 \pm .01$	$.80 \pm .01$	$.80 \pm .01$	$.80 \pm .01$	$.79 \pm .00$
mfeat-karhunen	$.91 \pm .01$	$.91 \pm .01$	$.91 \pm .01$	$.92 \pm .01$	$.92 \pm .01$	$.92 \pm .01$	$.91 \pm .00$
mfeat-pixel	$.88 \pm .02$	$.88 \pm .02$	$.89 \pm .02$	$.89 \pm .01$	$.89 \pm .01$	$.89 \pm .01$	$.91 \pm .01$
letter	$.55 \pm .03$	$.58 \pm .03$	$.59 \pm .02$	$.60 \pm .02$	$.61 \pm .01$	$.62 \pm .01$	$.58 \pm .01$
optdigits	$.93 \pm .01$	$.94 \pm .01$	$.94 \pm .01$	$.94 \pm .01$	$.95 \pm .01$	$.95 \pm .00$	$.94 \pm .00$
page-blocks	$.96 \pm .01$	$.96 \pm .01$	$.96 \pm .00$	$.96 \pm .00$	$.96 \pm .00$	$.96 \pm .00$	$.95 \pm .00$
pendigits	$.89 \pm .02$	$.91 \pm .02$	$.91 \pm .02$	$.92 \pm .01$	$.93 \pm .01$	$.93 \pm .01$	$.91 \pm .00$
vowel-bin	$.42 \pm .04$	$.47 \pm .04$	$.49 \pm .03$	$.51 \pm .03$	$.53 \pm .02$	$.54 \pm .02$	$.46 \pm .02$
yeast	$.52 \pm .01$	$.53 \pm .01$	$.53 \pm .01$	$.53 \pm .01$	$.54 \pm .01$	$.54 \pm .01$	$.52 \pm .01$

Table 3: Mean and standard deviation of the generation time in seconds with (a) CART and (b) logistic regression as the base learner.

(a)

dataset	Bo1	Bo5	Bo10	Bo20	Bo50	Bo100	RPND
LED24	.03 ± .00	.03 ± .00	.03 ± .00	.03 ± .00	.03 ± .00	.03 ± .00	.09 ± .00
zoo	.00 ± .00	.00 ± .00	.00 ± .00	.01 ± .00	.01 ± .00	.01 ± .00	.02 ± .00
krkopt-b.	.39 ± .02	.41 ± .02	.42 ± .02	.43 ± .02	.44 ± .01	.45 ± .01	2.04 ± .04
segment	.04 ± .01	.04 ± .00	.04 ± .01	.04 ± .01	.04 ± .01	.05 ± .01	.07 ± .01
mfeat-mo.	.02 ± .00	.02 ± .00	.02 ± .01	.02 ± .01	.02 ± .01	.03 ± .01	.06 ± .00
mfeat-fa.	.70 ± .05	.74 ± .05	.76 ± .05	.78 ± .04	.80 ± .04	.82 ± .03	1.03 ± .03
mfeat-fo.	.51 ± .04	.54 ± .05	.55 ± .04	.56 ± .04	.58 ± .04	.59 ± .03	.69 ± .02
mfeat-ka.	.50 ± .04	.53 ± .04	.55 ± .04	.56 ± .04	.57 ± .04	.59 ± .03	.66 ± .01
mfeat-pi.	.19 ± .01	.20 ± .01	.20 ± .01	.20 ± .01	.20 ± .01	.21 ± .01	.40 ± .02
letter	.52 ± .04	.56 ± .03	.58 ± .03	.59 ± .02	.60 ± .02	.61 ± .02	3.01 ± .04
optdigits	.23 ± .01	.23 ± .01	.24 ± .01	.24 ± .01	.24 ± .01	.25 ± .01	.45 ± .03
page-bl.	.05 ± .00	.05 ± .00	.06 ± .00	.06 ± .00	.06 ± .00	.06 ± .00	.07 ± .01
pendigits	.20 ± .01	.21 ± .01	.21 ± .01	.21 ± .01	.22 ± .01	.22 ± .01	.42 ± .01
vowel-bin	.03 ± .00	.03 ± .00	.03 ± .00	.03 ± .00	.03 ± .00	.03 ± .00	.06 ± .01
yeast	.02 ± .00	.02 ± .00	.02 ± .00	.02 ± .00	.02 ± .00	.02 ± .00	.05 ± .00

(b)

dataset	Bo1	Bo5	Bo10	Bo20	Bo50	Bo100	RPND
LED24	.23 ± .02	.25 ± .01	.25 ± .01	.25 ± .01	.26 ± .01	.26 ± .01	.42 ± .07
zoo	.03 ± .00	.03 ± .00	.03 ± .00	.03 ± .00	.03 ± .00	.03 ± .00	.05 ± .01
krkopt-b.	.97 ± .12	1.11 ± .13	1.17 ± .11	1.20 ± .10	1.25 ± .08	1.30 ± .07	5.46 ± 1.1
segment	.27 ± .04	.30 ± .03	.30 ± .03	.31 ± .03	.33 ± .03	.34 ± .03	.37 ± .09
mfeat-mo.	.04 ± .00	.05 ± .00	.05 ± .00	.05 ± .00	.05 ± .00	.05 ± .00	.10 ± .01
mfeat-fa.	.93 ± .08	1.02 ± .07	1.05 ± .06	1.07 ± .07	1.12 ± .11	1.18 ± .14	1.54 ± .19
mfeat-fo.	.27 ± .03	.29 ± .01	.29 ± .01	.30 ± .01	.31 ± .01	.31 ± .01	.41 ± .09
mfeat-ka.	.45 ± .04	.49 ± .03	.50 ± .03	.51 ± .03	.53 ± .02	.54 ± .02	.65 ± .10
mfeat-pi.	.68 ± .05	.74 ± .03	.76 ± .03	.76 ± .03	.79 ± .02	.80 ± .02	1.10 ± .15
letter	1.6 ± .21	1.89 ± .21	2.03 ± .19	2.11 ± .17	2.24 ± .16	2.32 ± .14	9.34 ± 1.6
optdigits	.74 ± .04	.78 ± .03	.80 ± .03	.82 ± .03	.83 ± .03	.85 ± .02	1.31 ± .14
page-bl.	.57 ± .12	.70 ± .06	.72 ± .05	.73 ± .04	.73 ± .03	.73 ± .03	.69 ± .13
pendigits	.95 ± .08	1.02 ± .07	1.05 ± .07	1.08 ± .07	1.11 ± .07	1.14 ± .05	1.70 ± .20
vowel-bin	.02 ± .00	.02 ± .00	.02 ± .00	.02 ± .00	.02 ± .00	.02 ± .00	.06 ± .01
yeast	.02 ± .00	.02 ± .00	.02 ± .00	.02 ± .00	.02 ± .00	.02 ± .00	.06 ± .02

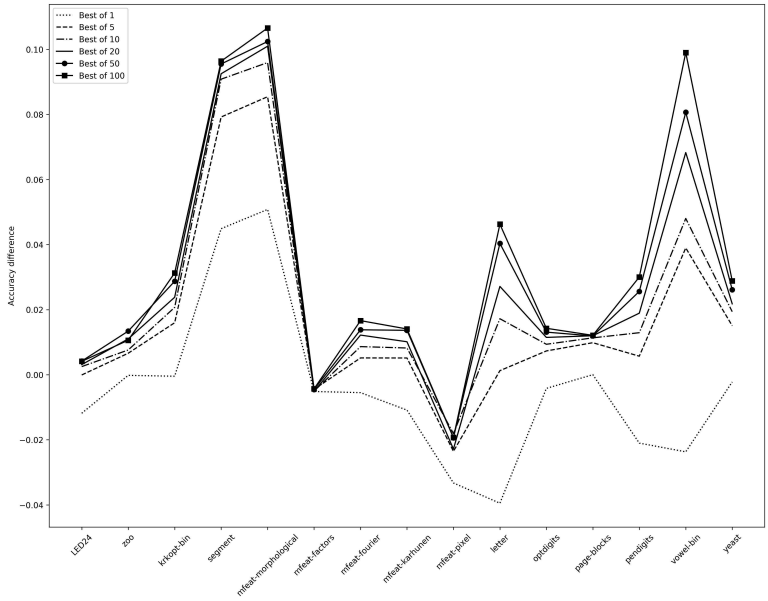


Figure 3: Difference in the mean accuracy between Best-of-K and RPND heuristic with logistic regression as the base learner.

5 Conclusion and Outlook

In this paper, we compare the state-of-the-art RPND heuristic for optimizing the structure of nested dichotomies with an extremely simple Best-of-K heuristic. To this end, an efficient algorithm for uniform sampling of nested dichotomies for a given number of classes c is also provided; the time complexity of this algorithm is $O(c)$. As the main result, we observe an increase in the predictive accuracy if logistic regression is used as a base learner. In the case of decision trees, both heuristics show similar performance.

While the focus of this study was on the optimization of a single nested dichotomy, the Best-of-K heuristic can easily be adapted for building ensembles of nested dichotomies. For example, this can be achieved by returning not only the single best performing nested dichotomy but multiple ones.

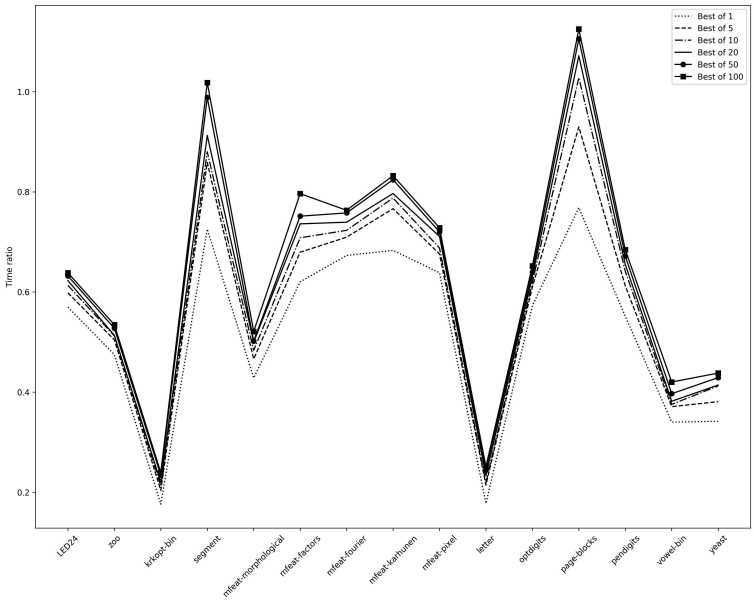


Figure 4: The ratio of mean total time (single nested dichotomy generation and training) between Best-of-K and RPND heuristic with logistic regression as base learner.

Acknowledgment

This work is part of the Collaborative Research Center "On-the-Fly Computing" at Paderborn University, which is supported by the German Research Foundation (DFG).

References

[1] D. N. A. Asuncion. UCI machine learning repository, 2007.

[2] L. Dong, E. Frank, and S. Kramer. Ensembles of balanced nested dichotomies for multi-class problems. In *Knowledge Discovery in Databases*, volume 3721 of *Lecture Notes in Computer Science*, pages 84–95. Springer, Berlin and Heidelberg and New York, 2005.

[3] M. M. Duarte-Villaseñor, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and M. Flores-Garrido. Nested dichotomies based on clustering. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: Proceedings 17th Iberoamerican Congress, CIARP 2012, Buenos Aires, Argentina*, pages 162–169. Springer, Berlin, Heidelberg, 2012.

[4] E. Frank and S. Kramer. Ensembles of nested dichotomies for multi-class problems. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, New York, NY, USA, 2004. ACM.

[5] G. W. Furnas. The generation of random, binary unordered trees. *Journal of Classification*, 1(1):187–233, 1984.

[6] T. Leathart, B. Pfahringer, and E. Frank. Building ensembles of adaptive nested dichotomies with random-pair selection. In *European Conference on Machine Learning and Knowledge Discovery in Databases, ECML PKDD 2016, Riva del Garda, Italy, Proceedings, Part II*, pages 179–194. Springer International Publishing, 2016.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] J. J. Rodríguez, C. García-Osorio, and J. Maudes. Forests of nested dichotomies. *Pattern Recognition Letters*, 31(2):125–132, 2010.

[9] F. J. Rohlf. Numbering binary trees with labeled terminal vertices. *Bulletin of Mathematical Biology*, 45(1):33 – 40, 1983.

Mining Flexibility Patterns in Energy Time Series from Industrial Processes

Nicole Ludwig, Simon Waczowicz, Ralf Mikut, Veit Hagenmeyer

Institute for Applied Computer Science, Karlsruhe Institute of Technology
E-Mail: nicole.ludwig@kit.edu

1 Introduction

The transition from traditional energy sources to renewable ones is not trivial. Various components in today's energy system are built for traditional suppliers and cannot cope with the new requirements imposed by the use of renewable energy sources (RES). The most commonly mentioned aspect differentiating renewable from traditional energy sources is the intermittent power generation, thus the power is not always generated when actually demanded. While changing the supply strategy is not an option as RES depend on e. g. wind and sunshine, changing the demand side is possible. Methods to change the energy demand behaviour are usually summarized under the term *Demand Side Management* (DSM) [16]. For these management strategies to work sufficiently, the consumer's flexibility has to be described mathematically [1] or determined from the data, as there is the possibility for usage shifts. DSM potential has been analysed in great detail for households as well as some energy-intensive industry processes (e. g. [7], [20]), however industrial batch processes have not yet been considered. As a first step in detecting demand side flexibility potential, this paper investigates the mining of patterns in energy time series data from industrial processes and introduces a new way of properly finding reoccurring motifs in industrial energy data.

Motif discovery, for example the Mueen-Keogh algorithm [13], has been applied to a great variety of problems, e. g. seismology [3], outlier detection [10], activity detection from sensor data [2], classifying heart sounds [9] and audio-based music structures [17]. In the realm of energy systems, the authors of [18] use motif discovery to reduce the dimensionality of large time series data sets and reduce the prediction errors when forecasting energy consumption. More prominently, motif discovery is used to identify individual appliances in energy consumption data. Some examples are given in [6], in which the ability is shown to disag-

gregate the energy consumption in a household from a single-point of entry, but need a training phase of one week to find the characteristics in the household. The authors of [8] give an overview over more methods for disaggregation of end-users smart meter data which is also known under the term non-intrusive load monitoring. However, all the methods mentioned work supervised and thus need labelling of the data.

Motif Discovery algorithms are very efficient at finding similar patterns in time series data. They excel if the found motifs are of approximately the same length. In energy time series this is the case for e. g. daily load curves from residential buildings. However, given industrial production data, the algorithms find proper motifs if the motifs are of approximately the same length, but they cannot identify the correct starting points if the motifs vary greatly in their length and hence cannot find the right motifs.

The main difficulty is thus the varying length of the motifs. This also leads to failure of all fixed window clustering approaches ([19], [4]). One could run the motif discovery algorithm several times with different window sizes but as we have no information about the processes possible lengths this approach seems tedious.

This paper uses a novel two-stage algorithm to correctly identify re-occurring motifs in industrial process time series data. We first use an event search algorithm to find possible starting points for the motifs and then use a standard motif discovery algorithm to identify which of those events are triggered by the same process. It is the aim to evaluate those motifs e. g. process lengths, energy intake etc. and classify the processes according to their degree of potential flexibility. The variability in the motifs is used as an indicator for potential flexibility but does not guarantee that this flexibility can also be used.

The remainder of the paper is structured as follows. We first introduces the used motif discovery and event search algorithms, before describing the found motifs and categorizing the resulting motifs in terms of their flexibility. Afterwards we discuss the implications and give a conclusion.

2 Methodology

The approach we chose can be split into four parts;

1. we discover where the *events* start with the help of a minimum search algorithm
2. we find how a usual motif looks like with the help of a univariate motif discovery
3. we compare the motif found to the processes after the events, establishing whether they are similar
4. we categorize the differences in the found motif according to their flexibility potential

In this section, we first introduce a common motif discovery algorithm before explaining our event search approach. The notion of flexibility as well as the measures we use to describe it will be introduced thereafter.

2.1 Univariate Motif Discovery

For our purpose of measuring flexibility potential, we search for cyclically reoccurring patterns with variations. Thus, we want to find subsequences in one time series that behave similarly but are not the same. The variation in the load profile can indicate a potential to shift electricity demand or find times in which we could reduce the demand through energy efficiency measures. Especially deviations from a specific pattern but also variations in the same pattern can be good indicators for these demand side management or flexibility potentials.

Hence, we want to model normal behaviour of load time series in industrial (batch) processes (e. g. power, gas, heat, steam) to be able to later detect deviations from this normal shape. A time series can summarize multiple processes being locally aggregated (e. g. in one building) or contain just one batch process. If we have no information about the number of types of the processes measured, we cannot use supervised methods to find the patterns or disaggregate the load information.

While traditional clustering approaches tend to be slow on larger data sets, the authors of [5] developed an algorithm which can detect reoccurring patterns in time series while scaling very well and being robust to noise. This so-called *motif discovery* can be applied to many applications, but

has not yet been applied to energy consumption load patterns for the purpose of finding demand side management potentials.

The algorithm is based on the *symbolic aggregated approximation* (SAX) of time series. For this approximation, the normalised time series are discretised and transformed into equal length *words* which are part of an *alphabet* with a predefined length. With the help of a *sliding window* a matrix $S^* \in (n - m + 1) \times w$ is generated, with n being the number of observations, m the number of subsequences and w the word length. The SAX representation of all subsequences, i. e. the words, are saved row-wise in this matrix. In every iteration of the algorithm, we randomly select l of the w columns of S^*, where l is a user-defined mask length and $l \leq w$. The word built with l columns is compared to all $(n - m + 1)$ rows of S^*. If there exists similarity – where similarity has to be defined beforehand and is in the following being understood as under a certain threshold distance – the corresponding entry in the so-called collision matrix is incremented. The entries with the highest values in the collision matrix are considered potential motifs. Those motifs are then iterated over the original time series and their distance is calculated to find the instances where the motif occurs. The distance measure in this paper is a simple Euclidean distance, but could also be e. g. dynamic time warping distance.

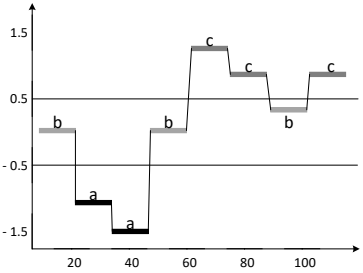


Figure 1: SAX transformation of an example time series, where each word (a-c) is a discrete part of the time series and the words are distributed based on a standard distribution.

2.2 Event Search

Motif discovery often fails if the motif is not behaving regularly, particular variations in the length of the process are hard to be detected for a fixed

word or window lengths. While industrial processes, especially chemical ones are often repeated batch processes, the batch processes do not have to be exactly the same for every iteration but can vary in length and intensity. As we struggle finding proper starting points with the fixed window approach of the motif discovery described before, we want to find the starting points separately. We chose to do a simple local minimum search in a predefined window, to find the supposed starting points for each batch process and save the time series subsequence between two minima as our batch process. For different time series, different event search methods should be used here. However, looking at industrial energy processes we expect them to have a clear minimum before they start. We treat the subsequences found equal to the motives found with the motif discovery algorithm before.

2.3 Dynamic Time Warping

Dynamic time warping (DTW) is a method to find the optimal alignment given two time-dependent sequences. Hence, given two time series x and y of length n and m, DTW will align the two series with the help of a $n \times m$ matrix. In this matrix, every element contains the distance between two points from the time series x and y, i. e. $d(x_i, y_j) = (x_i - y_j)^2$. This matrix is then used to find an optimal warping patch through the distance matrix, where we want to find the path that minimizes the warping cost:

$$\mathrm{DTW}(x, y) = \min \left\{ \sqrt{\sum_{k=1}^{K} w_k} \right. ,$$

with w_k defined as the k-th element of the warping path w. For further explanation of dynamic time warping see e. g. [14].

2.4 Flexibility

Given we found proper motifs with the above-described methods, we want to examine their potential flexibility. We work with the flexibility definition of [15]. They describe flexibility as *"the amount of energy and the duration of time to which the device energy profile (energy flexibility) and/or activation time (time flexibility) can be changed."* Based on this definition and translating it to a data-driven approach, there are several cases, for which we identify a potential degree of flexibility:

1. **Energy Flexibility**. The pattern always starts to occur at the same time and weekday but the length (duration) and intensity (power) vary. This would indicate that the process can run in different modes and there is thus some degree of flexibility in the decision on the modes.
2. **Time Flexibility**. The same pattern in the time series occurs in the same form at different e. g. times of day or days of a week, depending on the length of the pattern. For example, a process always runs Mondays, but the time varies greatly, this would indicate some level of flexibility regarding the starting time of the process on Mondays.
3. A combination of the above-mentioned cases is also possible.

It is our aim to find *potential flexibility*, neither the technical feasibility nor the economic suitability will be considered in this paper (Figure 2) as we do not have the necessary information. Regarding the economic feasibility we can note that plants are usually constructed according to their sales potential. With flexible energy consumption one might have to use over-dimensioned plants to achieve the same product output in the same amount of time, which would result in higher fix costs. To properly evaluate the real flexibility, we would thus need expert knowledge about all the technical and economic properties of the processes under investigation.

We describe the measures we use to judge the different types of flexibility in the following.

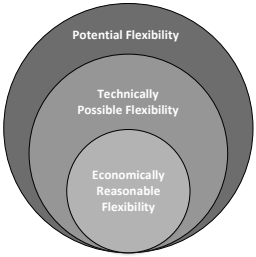


Figure 2: We differentiate between potential flexibility, which can be found via motif discovery and technical and economical flexibility, which can only be determined with the help of expert knowledge.

2.4.1 Energy Flexibility

As stated before, the energy flexibility is mainly concerned with variations in the motif itself as opposed to when the motif occurs. We have a look at three different measures: the length, intensity and ramping time of the motifs.

The length is described by the time steps between the current local minimum (i) and the next local minimum (j) found through the event search.

$$\text{length} = \text{t}(\text{localmin}[j]) - \text{t}(\text{localmin}[i]) \quad \text{for} \quad i < j \tag{1}$$

The intensity is measured as the area under the curve, thus the energy used by the process. We approximate the area under the curve (AUC) using the composite trapezoidal rule. Given an interval $[a, b]$, we split this interval in M subintervals $[x_k, x_{k+1}]$ of equal width $h = (b-a)/M$ by using the space nodes $x_k = a + kh \qquad \forall k = 0, 1, \ldots, M$. The composite trapezoidal rule for M subintervals can then be expressed as

$$T(f, h) = \frac{h}{2} \sum_{k=1}^{M} (f(x_{k-1}) + f(x_k)), \tag{2}$$

which is an approximation for the integral of $f(x)$ over $[a, b]$

$$\int_a^b f(x)dx \approx T(f, h). \tag{3}$$

We compare the ramping Δ_t of the motifs by measuring the time steps needed between the local minimum, thus starting point and the local maximum of the time series.

$$\Delta_t = t(\text{localmax}) - t(\text{localmin}). \tag{4}$$

2.4.2 Time Flexibility

In contrast to the energy flexibility described before, the time flexibility is only concerned with the starting times, days of week and length, thus end times and days of a week. We assume that if a process runs every Monday at 8 am it has to run at this specific point in time, while running

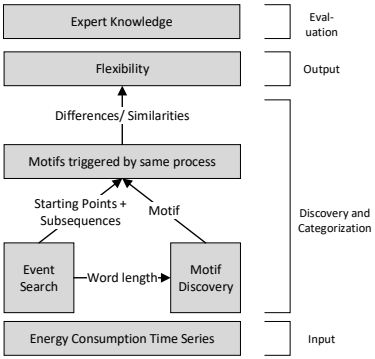


Figure 3: The framework used in this paper to extract flexibility potential from energy consumption data via motif discovery.

always at a different day of the week would indicate that we have a certain flexibility, as to when the process has to start. This is obviously a simplification as we neglect dependencies from other processes at the moment.

2.5 Framework

We use three steps to get to a categorisation of flexibility from steam consumption data before an expert lastly evaluates the potential found. The whole framework is graphically shown in Figure 3. We start with the event search algorithm, saving the subsequences between each event as possible motifs. Afterwards, we run the motif discovery algorithm using the average length of the event subsequences as the word and window length. We then measure the similarity of the found motifs with the subsequences, using dynamic time warping. The motifs which are similar enough are used to examine their differences more closely and categorize their flexibility potential. In the last step, an expert for the processes is needed to evaluate which of the potential flexibilities found can actually be used for flexibility measures and which are due to other factors.

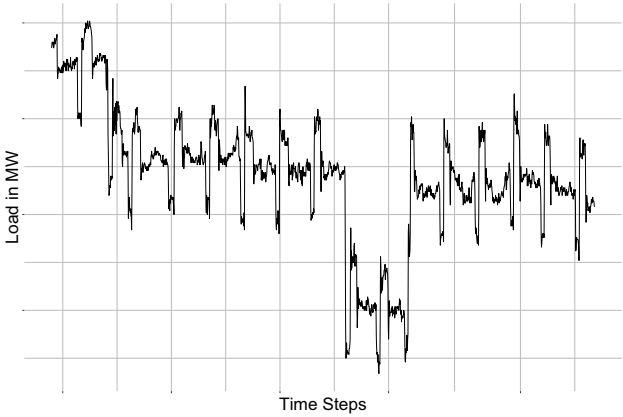


Figure 4: Original time series data of a buildings steam demand over five weeks.

3 Results

To evaluate the proposed method we have a look at the load consumption time series from a building of a chemical factory. The steam consumption is measured as hourly average in tons over a period of five weeks. Due to commercial sensitivity issues, no true measurement values or time indicators will be presented in this paper. Although we do not have information on the products manufactured, we know that there are possibly one or several repeating processes in the factory building. Given this consumption data, we want to find the shapes of those recurring processes. The time series is depicted in Figure 4. Visual analysis of the graph indicates a pattern which repeats itself on different levels of steam intake.

This section presents the individual steps described in the framework, starting with the motif discovery and event search, before analysing and categorizing the flexibility.

3.1 Motif Discovery

Following the framework we have established, we run the above described minima search algorithm with a window size of 100 and store the local minima found in a vector. We then build subsequences of the time series, starting at each of those minima. One subsequence is only five time

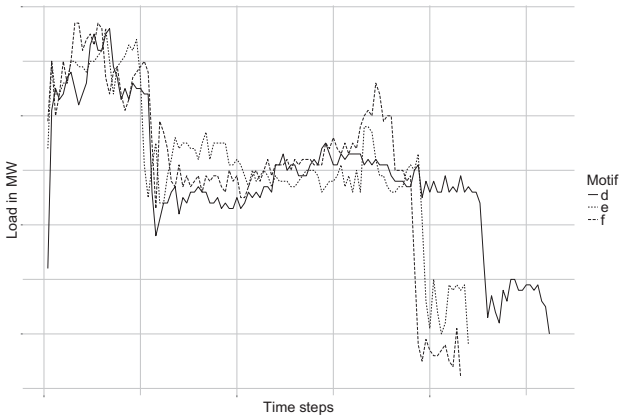


Figure 5: Exemplary subsequences of the steam consumption time series. Each subsequence starts after a minimum detected through the minima event search with a window of size 100.

steps long, as all other subsequences are substantially longer, we treated this short one as belonging to the next subsequence. Additionally the sequence until the first minimum is also treated as a subsequence.

The minima search gives a very good result to find the different time series subsequences which look similar enough to be from the same batch process. Figure 5 shows some exemplary subsequences found through the event search. The structure of the process seems to remain the same over the different instances. However, the length and intensity of the process vary, which is useful for our purpose as it could be an indicator for flexibility. Figure 6 shows a heatmap of all the subsequences, where we have normalized all subsequences with mean $\mu = 0$ and standard deviation $\sigma = 1$. The subsequences are aligned at their minimum point. We can clearly see that they exhibit a similar structure but differ in length.

Next, we run the motif discovery algorithm using the additional information we have about the processes. We chose a relatively small alphabet size of $a = 10$ as the variations in patterns do not seem to be high. Choosing the word length is more difficult as it has a great impact on the found motifs. Additional information on the *normal* length of the process running would be the obvious choice for those parameters, which we do not have in our case. However, given the subsequences found in the

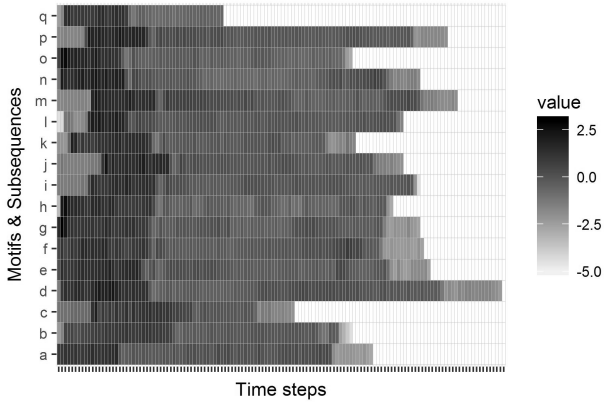


Figure 6: Heatmap of the individual subsequences found through the minima search. The subsequences start at the minimum and end at the next minimum. For a better comparison of their structure they are normalized ($\mu = 0, \sigma = 1$).

event search, we can now use the mean time distance between the local minima found (w = 94). For the moment we do not allow for overlaps between the motifs.

The algorithm finds two different motifs, whose starting points in the time series can be seen in Figure 7. The individual instances of the motifs are displayed in Figure 8. Motif 1 occurs two times in the time series, while Motif 2 occurs three times. There are variations in the motifs, which is helpful for our flexibility examination later. However, they also do not look very different, which lets us assume they might be similar enough to be the same motif if their comparison would start at the minimum as with the event search sequences. All other patterns are classified by the algorithm as not being repeated.

To further investigate this idea, we compare the DTW distances of the motifs within themselves and in between each other. To be comparable, the motifs have all been normalized with mean $\mu = 0$ and standard deviation $\sigma = 1$. The result can be found in Table 1. The distances vary between 20.96 and 76.97, with the first one being the distance between the two instances in the first motif and the second one being the distance between the third instance of the second motif and the first instance of the first motif. The third instance of the second motif seems to be the odd one out.

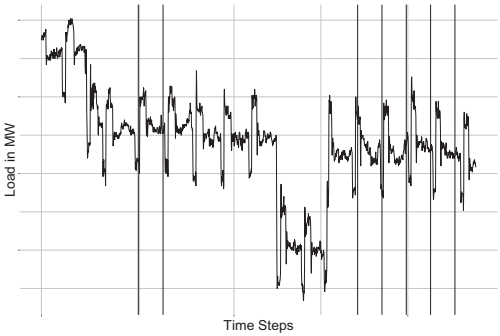
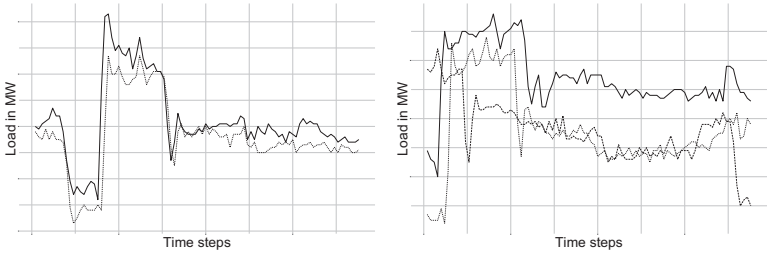


Figure 7: The steam consumption time series with the starting and ending point of the discovered motifs marked by the vertical dotted lines.



(a) The first motif with its two instances (1a – 1b).

(b) The second motif and its three instances (2a – 2c).

Figure 8: The two motifs found by the motif discovery algorithm.

This result might be to the rather large drop of load at the beginning of the instance. The heatmap depicted in Figure 9 confirms this idea, with the motif in the top row (2c), having a shorter period at a high-level energy intake before dropping. However, we will assume that the distance also for the third instance is small enough for our purposes and so assume that all five instances are triggered by the same mechanism.

As a next step, we want to have a look at the subsequences from the event search. We have seen from the heatmap before, that they all exhibit a similar pattern but differ mainly in length (see Figure 6). We now compare the motifs found through the motif discovery algorithm with the subsequences determined through the event search. It is our aim to find how similar those motifs are to each other. Table 2 shows the difference

Table 1: The dynamic time warping distances for the motifs within themselves. The minimum distance for each motif is highlighted.

Motif	1a	1b	2a	2b	2c
1a	0	**20.96**	52.51	47.13	76.97
1b	**20.96**	0	49.04	42.25	75.07
2a	52.50	49.04	0	**27.83**	**59.43**
2b	47.13	42.26	**27.83**	0	59.63
2c	76.97	75.07	59.43	59.63	0

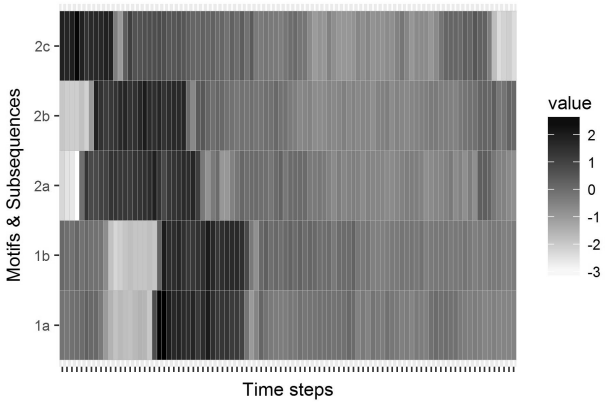


Figure 9: Heatmap of the two motifs found with their respective instances. The motifs are normalized with $\mu = 0$ and $\sigma = 1$.

calculated between each motif found with the motif discovery algorithm (1a – 2c) and the event search subsequences (a – q). As we can see in the table, the distances are for many sequences greater than those for the motifs within themselves. However, we also have some subsequences, where the distances are lower than within the motifs, e. g. subsequence q and motif 2c have a DTW distance of 23.90.

3.2 Flexibility Measures

Given the subsequences which we have found are similar enough to stem from the same underlying process, we now want to determine their flexibility. However, before going further into the analysis of the flexibility we can find by comparing the subsequences according to our

flexibility measures, we first match the subsequences with the five motif instances. Thus, we will not take the motifs into our analysis if one of the subsequences already describes them.

Table 2: The dynamic time warping distances for each subsequence (a – q) with the motifs (1a – 2c), the minimum distance is highlighted.

Motif	a	b	c	d	e	f
1a	89.43	57.26	57.52	62.80	64.21	73.88
1b	93.45	62.16	61.59	65.89	63.81	78.00
2a	63.71	**46.32**	**49.84**	**48.99**	44.44	63.37
2b	70.06	60.01	58.86	62.49	62.87	74.96
2c	**44.90**	53.19	61.79	50.12	**40.87**	**41.94**

Motif	g	h	i	j	k	l
1a	84.65	76.25	50.15	48.50	52.50	46.95
1b	94.78	78.00	53.47	52.31	53.78	46.72
2a	58.19	**37.16**	50.00	52.37	**33.36**	46.43
2b	74.75	45.24	**42.31**	**44.16**	37.70	**32.58**
2c	**37.07**	39.55	71.95	79.56	53.62	70.60

Motif	m	n	o	p	q
1a	57.39	56.30	44.55	49.01	42.53
1b	58.73	59.77	57.35	50.18	49.43
2a	49.22	39.94	42.81	45.48	**23.90**
2b	**42.19**	33.34	52.59	**44.18**	32.43
2c	62.55	**28.45**	**38.24**	62.91	54.49

Table 4 shows that all motif starting points are reasonably close to starting points of the subsequences. The motifs are included in the subsequences labelled e, m, n, o and p. Hence, we only use these subsequences from now on and use the term motifs for them interchangeably. We analyse the motifs with the help of the above-established flexibility measures. First, we examine the energy flexibility, before delving into the time flexibility.

Table 4: The subsequences and their starting indices, compared to the motifs and their starting indices. All motifs can be reasonably described by a subsequence.

Sub.	Start	Motif	Start	Sub.	Start	Motif	Start
a	0			j	913		
b	93			k	1015		
c	180			l	1103		
d	250			m	1205	2b	1223
e	381	2a	377	n	1323	2c	1317
f	491			o	1430	1a	1411
g	599			p	1517	1b	1505
h	706			q	1632		
i	805						

3.2.1 Energy Flexibility

According to our three measures described before, we inspect the differences in energy usage from our subsequences. The results are displayed in Table 5. The length of the processes varies between 49 and 131 time steps. However, the intensity for some motifs is rather similar although the lengths differ. This might indicate that no matter how long or short the process is, we need to use roughly the same amount of energy. The ramping time changes considerably between the instances. While the shortest ramping takes only two time steps, the longest takes more than 90 time steps.

Correlation between the length of the process and the intensity as measured by the area under the curve is $\rho = 0.8446$, indicating that a longer process also uses more energy. However, ramping steps and the length of the process are only slightly positively correlated with $\rho = 0.2832$, as are the ramping steps and the intensity $\rho = 0.3649$.

Table 5: Characteristics of the found patterns in the steam demand time series.

Characteristic	Min	Median	Max	StdDev
Length	49.00	102.00	131.00	19.13
Intensity	293.94	294.37	310.97	7.31
Ramping Time	2.00	12.50	90.00	21.68

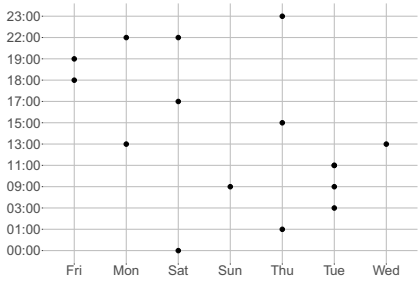


Figure 10: Starting date and time for each event.

3.2.2 Time Flexibility

Finally, we consider the time flexibility of our sequences. Figure 10 shows the hour and the day of the week where each local minimum is located. As we can see, the weekdays and times of these possible starting points are well distributed throughout the week and day. This indicates that there is no dependency on the weekday or hour of the day to start the process. This might thus be a highly automated process or one with several shifts throughout the day. The fact that there is no clear pattern in the time of the week could mean that we are free to choose the starting point time-wise and thus have a great flexibility there. It could however also imply that the process highly depends on some other variables which we do not consider.

3.3 Categorisation of Flexibility

Given the results in the previous sections, we classify the potential flexibility we get. As we only find potential flexibility at the moment, we cannot infer the technical or economical usability of those flexibilities before consulting an expert. However, we can rank the potential flexibility, and use three different flexibility categories in the following.

The highest available flexibility seems to be in the ramping process. The ramping time is almost independent from the length of the process and only slightly correlated with the intensity of the process. This could indicate that the ramping is flexible as long as we end up at the right intensity at the preferred time point.

The length of the process also varies greatly but is correlated to the intensity. This implies that we can choose whether we start a short or long batch, depending on the energy available.

Lastly, the starting times vary across a wide range, this might indicate that we do not have restrictions considering the time of day when to start a process. As we also have starting times in the middle of the night, we can assume that there are either workers available during the night or this is an automated process. Either way, the starting time can be chosen rather freely. However, as the energy intake of the process is seldom zero, the process either needs a specific amount of energy in a stand-by modus, or the starting time highly depends on what happened before. There might be restrictions as to how long the break between two batches can be, etc.

4 Conclusion and Outlook

This paper proposed a new method to find motifs in industrial batch processes and applied the method to time series of steam consumption data in chemical processes.

The flexibility we find in this paper is only *potential flexibility*. As we do not know anything about the process in terms of its technical properties or dependencies of other processes before or after it is running, we only indicate that there might be a potential as there has been some variety in the past. If this variety is only driven by technical features it would not be considered flexibility. Thus, in a next step to properly quantify

the flexibility one has to talk to the process manager and let him rate the potentials found according to their usability. We might also not want to use the flexibility found, even if technically possible, as it is not economically feasible (see Figure 2 for the differences).

Future work will address several aspects of the paper. An investigation of other event search algorithms might be useful, especially if we expect pauses in the processes which are then also a minimum but not an indicator for a process start. Concerning the similarity of the time series sequences, we want to use sequence alignment and flying bricks in a next step. The purpose then being to match the SAX representations of the subsequences and find how the motifs differ and are similar to each other instead of measuring the dynamic time warping distance. This would allow for a more detailed investigation into what can be used for flexibility purposes in the future. Furthermore, we will analyse the potential flexibility and find the real technical and economical flexibility with the help of a process expert. Lastly, given this flexibility, we want to find market mechanisms to encourage the use of flexibility by the process manager whenever this would be more efficient or cheaper. We also want to apply this method to other data and implement it in the Energy Lab 2.0 [11] at a later stage.

Acknowledgements

This work was partly funded by the German Research Foundation (DFG) Research Training Group 2153 “Energy Status Data – Informatics Methods for its Collection, Analysis and Exploitation”.

References

[1] L. Barth, N. Ludwig, E. Mengelkamp, and P. Staudt. A comprehensive modelling framework for demand side flexibility in smart grids. *Computer Science - Research and Development*, 30(4):1758, 2017.

[2] E. Berlin and K. van Laerhoven. Detecting leisure activities with dense motif discovery. In Anind K. Dey, editor, *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 250–259, New York, NY, 2012. ACM.

[3] C. Cassisi, M. Aliotta, A. Cannata, P. Montalto, D. Patanè, A. Pulvirenti, and L. Spampinato. Motif discovery on seismic amplitude time series: The case study of Mt Etna 2011 eruptive activity. *Pure and Applied Geophysics*, 170(4):529–545, 2013.

[4] G. Chicco. Overview and performance assessment of the clustering methods for electrical load pattern grouping. *Energy*, 42(1):68–80, 2012.

[5] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In Ted Senator, editor, *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 493–498, New York, NY, 2003. ACM.

[6] L. Farinaccio and R. Zmeureanu. Using a pattern recognition approach to disaggregate the total electricity consumption in a house into the major end-uses. *Energy and Buildings*, 30(3):245–259, 1999.

[7] A. Faruqui and S. Sergici. Household response to dynamic pricing of electricity: A survey of 15 experiments. *Journal of Regulatory Economics*, 38(2):193–225, 2010.

[8] J. Froehlich, E. Larson, S. Gupta, G. Cohn, M. Reynolds, and S. Patel. Disaggregated end-use energy sensing for the smart grid. *IEEE Pervasive Computing*, 10(1):28–39, 2011.

[9] E. F. Gomes, A. M. Jorge, and P. J. Azevedo. Classifying heart sounds using multiresolution time series motifs. In Bipin C. Desai, editor, *Proceedings of the International C* Conference on Computer Science and Software Engineering*, pages 23–30, New York, NY, 2013. ACM.

[10] M. Gupta, J. Gao, C. Aggarwal, and J. Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, 2014.

[11] V. Hagenmeyer, H.K. Çakmak, C. Düpmeier, T. Faulwasser, J. Isele, H. B. Keller, P. Kohlhepp, U. Kühnapfel, U. Stucky, S. Waczowicz, and R. Mikut. Information and communication technology in Energy Lab 2.0: Smart energies system simulation and control center with an Open-Street-Map-based power flow simulation example. *Energy Technology*, 4(1):145–162, 2016.

[12] C. Lin and C. L. Moodie. Hierarchical production planning for a modern steel manufacturing system. *International Journal of Production Research*, 27(4):613–628, 2007.

[13] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover. Exact discovery of time series motifs. In Haesun Park, editor, *Proceedings of the Ninth SIAM International Conference on Data Mining*, pages 473–484. SIAM, Philadelphia, 2009.

[14] M. Müller. *Information Retrieval for Music and Motion*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[15] B. Neupane, T. B. Pedersen, and B. Thiesson. Towards flexibility detection in device-level energy consumption. In Wei Lee Woon, editor, *Data analytics for renewable energy integration*, volume 8817 of *Lecture Notes in Computer Science*, pages 1–16. Springer, Cham, 2014.

[16] P. Palensky and D. Dietrich. Demand side management: Demand response, intelligent energy systems, and smart loads. *IEEE Transactions on Industrial Informatics*, 7(3):381–388, 2011.

[17] J. Serra, M. Müller, P. Grosche, and J. L. Arcos. Unsupervised detection of music boundaries by time series structure features. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[18] Y. Simmhan and M. U. Noor. Scalable prediction of energy consumption using incremental time series clustering. In Xiaohua Hu, editor, *IEEE International Conference on Big Data, 2013*, pages 29–36, Piscataway, NJ, 2013. IEEE.

[19] S. Waczowicz, M. Reischl, V. Hagenmeyer, R. Mikut, S. Klaiber, P. Bretschneider, I. Konotop, and D. Westermann. Demand response clustering - how do dynamic prices affect household electricity consumption? In *2015 IEEE Eindhoven PowerTech*, pages 1–6. IEEE, 2015.

[20] S. Waczowicz, M. Reischl, S. Klaiber, P. Bretschneider, I. Konotop, D. Westermann, V. Hagenmeyer, and R. Mikut. Virtual storages as theoretically motivated demand response models for enhanced smart grid operations. *Energy Technology*, 4(1):163–176, 2016.

Zuverlässigkeitsbasierte Fusion von Fahrstreifeninformationen für Fahrerassistenzfunktionen

Tran Tuan Nguyen[1], Jens Spehr[1], Jan-Ole Perschewski[1], Fabian Engel[1], Sebastian Zug[2], Rudolf Kruse[2]

[1]Volkswagen AG, 38436 Wolfsburg
E-Mail: {tran.tuan.nguyen, jens.spehr, jan-ole.perschewski, fabian.engel}@volkswagen.de
[2]Otto-von-Guericke Universität Magdeburg, 39106 Magdeburg
E-Mail: {sebastian.zug, rudolf.kruse}@ovgu.de

Kurzfassung

Eine zentrale Aufgabe für Fahrerassistenzsysteme und autonomes Fahren ist die robuste Fahrstreifenerkennung. Dafür müssen Informationen aus verschiedenen Sensoren kombiniert werden, etwa Kameras, Lidars, Radars etc. Die große Herausforderung bei der Fusion verschiedener Informationsquellen besteht darin, dass diese Quellen unterschiedliche Zuverlässigkeiten in unterschiedlichen Situationen aufweisen. Daher stellt diese Arbeit ein Fusionskonzept vor, welches die Zuverlässigkeiten der Quellen berücksichtigt. Dazu wird eine Metrik präsentiert, mit der sich die Qualität einzelner Fahrstreifenerkennungen quantifizieren lässt. Anschließend werden unterschiedliche Klassifikationsverfahren eingesetzt, um Zuverlässigkeiten der Sensoren in verschiedenen Szenarien zu lernen. Anhand der prädizierten Zuverlässigkeiten erfolgt eine adaptive Fusion der verlässlichen Quellen. Anhand experimenteller Erprobungen kann der Nutzen des vorgestellten Konzepts durch eine Performanzsteigerung von bis zu 7% gezeigt werden.

1 Einleitung

In den letzten Jahren rückt automatisches Fahren in den Fokus zahlreicher Forschungseinrichtungen und Unternehmen. Dabei ist die Fahrbahnerkennung eine der entscheidensten Aufgaben. Dazu präsentieren viele

Arbeiten unterschiedliche kamerabasierte Ansätze zur Erkennung von Fahrbahnmakierungen [1]. Allerdings liefern diese Ansätze keine stabilen Ergebnisse oder schlagen komplett fehl, wenn Markierungen auf einer oder beiden Seiten nicht vorhanden oder schlecht sichtbar sind. Bild 1 zeigt einige Beispiele. Zur Lösung dieser Probleme können weitere Informationsquellen herangezogen werden, wie die Trajektorie des Vorderfahrzeugs, eine digitale Karte, Freiflächeninformationen etc. Diese Quellen unterscheiden sich jedoch abhängig von Fahrbahn- bzw. Umfeldbedingungen in ihrer Performanz. Die kamerabasierte Markierungserkennung funktioniert beispielsweise gut auf Autobahnen und Landstraßen. Allerdings sinkt ihre Performanz in städtischen Szenarien, wo häufig nur Asphaltübergänge oder Bordsteine die Fahrbahn begrenzen (Bild 1b und 1c). In Szenarien mit schlechten Markierungen, engen Kurven, ungünstigen Wetter- oder Lichtbedingung steigen daher sowohl die Positions- als auch Existenzunsicherheit dieser Detektionen. In solchen Fällen besteht die Alternative darin, dem Vorderfahrzeug zu folgen. Jedoch kann dies auch zu Problemen führen, wenn der Vordermann plötzlich den Fahrstreifen wechselt. Bei Diskrepanzen zwischen erkannten Markierungen und Vorderfahrzeug, kann zur Lösung eine digitale Karte herangezogen werden. Dabei führen die Positionsungenauigkeit oder die Aktualität der Karte allerdings oftmals zu neuen Problemen, durch die sich die Performanz zusätzlich verschlechtert.

Daher besteht das Ziel der vorliegenden Arbeit darin, diese Probleme mit einem hier vorgestellten, zuverlässigkeitsbasierten Fusionskonzept zu bewältigen. Dazu wird die Zuverlässigkeit der einzelnen Quellen für jede gegebene Situation bewertet, um ausschließlich die jeweils verlässlichen Quellen zu verwenden [2].

(a) Beide Seiten (b) Nur eine Seite (c) Markierung fehlt

Bild 1: Unterschiedliche Ausprägungen der vorhandenen Markierungen

2 Stand der Technik

Im Allgemeinen bezeichnet die Informationsfusion einen Prozess, der sich mit der Assoziation, Korrelation und Kombinationen von Daten und Informationen mehrerer Quellen, unter der Betrachtung von a priori Wissen und Unsicherheitsmodellen, beschäftigt. Durch die Fusion werden verbesserte Positions- und Identitätsschätzungen so wie eine vollständige und zeitliche Bewertungen von Situationen und Bedrohungen erreicht [3, 4]. Der Erfolg von Informationsfusion kann anhand der Übereinstimmung von Fusionsergebnissen und Realität gemessen werden. Dieser ist abhängig von der Qualität und der Genauigkeit der Daten, der Unsicherheitsmodelle sowie des a priori Wissens [3].

In der Literatur bekommt bisher die Modellierung von Unsicherheitsmodellen mehr Beachtungen als der Zuverlässigkeitsaspekt [3]. Dies führt zu der optimistischen Annahme, dass alle Quellen in gleichem Maße zuverlässig sind und eine symmetrische Rolle spielen. Auf diese Annahme setzen die Autoren in [5, 6, 7], um bei einer Fusion den Positionsfehler der geschätzten Fahrbahn zu minimieren. Allerdings gilt diese Annahme nicht, wenn die Quellen unterschiedlich zuverlässig sind. Dies kann durch interne Faktoren (z.B.: Genauigkeit, Fehlerrate) oder externe Faktoren (z.B.: Standort, Wetter, Verkehr, Sichtbarkeit von Markierungen, Tageszeit) verursacht werden. Zur Vermeidung einer Redukion der Fusionsperformanz durch diese optimistische Annahme soll die Zuverlässigkeit der Informationsquellen betrachtet werden.

Brade et al. definieren eine Fehleralgebra für ein verteiltes Sensorsystem aus Komponenten mit unterschiedlichen Messprinzipien [8]. Dabei wird jedem Sensor ein Fehlerdetektor zugeordnet, der die möglichen Fehler dieses Sensors anhand dessen Messungen erkennt. Je nach Auftrittswahrscheinlichkeit der Fehler wird dem Sensor ein Validitätswert zugewiesen, welcher das Gewicht der aktuellen Sensormessung bei der Fusion darstellt. Allerdings hat dieser Ansatz den Nachteil, dass eine explizite Fehlererkennung nötig ist, welche detailliertes Wissen über die möglichen Sensorfehler sowie deren Auswirkungen für die Anwendungen voraussetzt.

Guo et al. nutzen a priori Wissen und komplexe Modelle für eine Zwei-Stufen-Plausibilisierung von detektierten Objekten und Markierungen, welche dann zur Fahrbahnschätzung verwendet werden [9]. In der ersten Stufe erfolgt eine Kreuzvalidierung zwischen den erkannten Verkehrsteilnehmern und den detektierten Markierungen, um die Anzahl der *False Positives* zu reduzieren. Die nächste Stufe realisiert eine Szeneninter-

pretation der Interaktionen zwischen den erfassten Objekten und der Fahrbahn. Jedoch kann es passieren, dass das a priori Wissen und die erstellten Modelle nicht für neue unbekannte Szenarien geeignet und nur schwer erweiterbar sind.

Als ein viel versprechender Ansatz trainieren Hartmann et al. ein neuronales Netz, um die Fehlerwahrscheinlichkeit einer Karte vorherzusagen [10]. Dabei benutzen die Autoren als Eingangsdaten die Daten aus der Karte sowie die sensorischen Detektionen aus einem Radar und einer Infrarotkamera. Das Netz ist darauf trainiert, den Fahrer bei einer Abweichung zwischen Karte und Sensordaten zu alarmieren. Allerdings ist bei einer solchen Abweichung unklar, ob die Sensormessung oder die Karte inkorrekt ist. Daher kann eine Erweiterung zur Identifikation der fehlerhaften Quellen dabei helfen, diese auszuschließen.

Anders als die Autoren aus [10] setzen Frigui et al. Clustering-Algorithmen ein, um aus Eingangsdaten unterschiedliche Cluster zu bestimmen [11]. Dabei repräsentiert jeder Cluster einen möglichen Kontext bzw. ein Szenario, in dem die Quellen ein bestimmtes Verhaltensmuster aufweisen. Anhand der Zuverlässigkeit jeder Quelle in einem bestimmten Kontext wird ihr ein entsprechendes Gewicht für die Fusion zugewiesen. Dabei besteht die Herausforderung, die richtige Anzahl der Szenarien zu bestimmen. Außerdem wird diese Problemstellung bei der Integration weiterer Sensoren komplexer, da die Anzahl der Clusters exponentiell steigt.

Ähnlich dazu beschreiben Romero et al. [12] eine umgebungsbasierte Fusion zweier Detektoren. Die Kernidee besteht darin, anhand der Aufnahmen der Detektoren eine Karte zu erstellen. Zur Laufzeit nutzt der Roboter diese Karte und verwendet den Detektor, der in der Vergangenheit an dieser Stelle die beste Performanz zeigte. Allerdings funktioniert dieser Ansatz nur für bereits abgefahrene Gebiete und ist nicht generalisierbar, da reale GPS-Positionen auf die Sensorperformanz abgebildet werden. Bei der Verwendung anderer Features anstatt der GPS-Koordinaten kann diese Methode auch für unbekannte Gebiete angewendet werden.

Mit der Zielstellung Fusionsergebnisse zu verbessern, präsentiert Realpe et al. ein fehlertolerantes Framework zur Schätzung der Zuverlässigkeit mehrerer Informationsquellen aus Trainingsdaten [13]. In der Trainingsphase werden die Aufzeichnungen verschiedener Detektoren, z.B.: Kameras, Lidars, Radars, mit den Referenzdaten eines hochpräzisen Velodyne Lidar verglichen. Anschließend wird jedem Detektor abhängig von seiner Fehlerhäufigkeit ein Gewicht für die Fusion zugewiesen. Der Vorteil dieses Ansatzes besteht darin, dass die Zuverlässigkeit der Detektoren anhand ih-

rer Performanz automatisch bestimmt wird. Im Vergleich zu [11] ist keine explizite Szenarienerkennung nötig. Ferner können Kontextinformationen integriert werden, um die Zuverlässigkeitsschätzung zu verbessern.

3 Systemarchitektur

Wie Abschnitt 2 einführt, existieren in der Literatur verschiedene Möglichkeiten, um aus Sensordaten Zuverlässigkeiten zu schätzen (Bild 2) [14].

Der klassische Ansatz (1) kombiniert einzelne Verfahren, beispielsweise Objekterkennung (*Object Detection*) aus Sensordaten mit einer anschließenden Merkmalsextraktion (*Feature Extraction*) und Mustererkennung (*Pattern Recognition*). Darauf basierend werden mit Expertenwissen manuell Regeln definiert, die die Zuverlässigkeit der Sensoren ableiten. Der Vorteil dieses Ansatzes besteht darin, dass die einzelnen Schritte gut nachvollziehbar sind und separat getestet werden können. Änderungen an der Anzahl und Art der verwendeten Sensoren bringen in diesem Fall allerdings einen hohen Anpassungsaufwand mit sich, da jedes der Module modifiziert werden muss. Außerdem besteht die Schwierigkeit alle relevanten Szenarien zu modellieren und zu erkennen. Im Rahmen neuer Verfahren wie Deep Learning werden auch Ansätze genutzt, die Sensordaten direkt auf Zuverlässigkeitswerte in einem Schritt abbilden, wie in (3) und (4) zu sehen. Dabei können sämtliche, in den Eingangsdaten enthaltenen Informationen genutzt werden. Nachteilig ist jedoch, dass diese Beziehung schwierig zu lernen und das so entstehende System für den Menschen wenig verständlich ist. Zudem kann seine Funktionsweise kaum analysiert werden und für gute Ergebnisse sind große Datenmengen erforderlich.

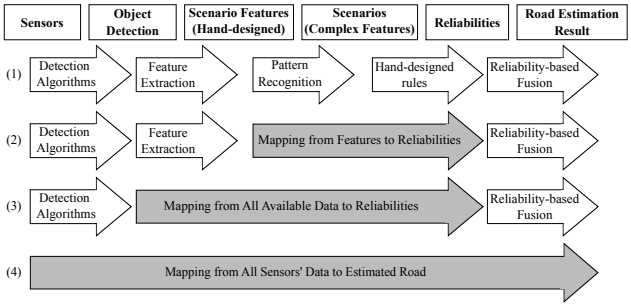


Bild 2: Verschiedene Möglichkeiten zur Zuverlässigkeitsschätzung

Aus diesen theoretischen Überlegungen folgt, dass sich die zweite Methode, eine Kombination beider Ansätze, am besten eignet. Auf dieser Basis illustriert Bild 3 die vorgestellte Systemarchitektur zur Integration von Zuverlässigkeiten in die Fusion unterschiedlicher Ego-Lane-Hypothesen. Diese Hypothesen entstehen aus einer modellbasierten Hypothesengenerierung anhand einer *Compositional Hierarchy* aus [15], eines hierarchischen, ungerichteten Graphen zur Interpretation von unterschiedlichen Straßenszenarien. Für die Zuverlässigkeitsschätzung werden aus den Sensordaten zunächst Objekte generiert und aus ihnen Merkmale extrahiert. Anschließend werden die Zuverlässigkeitswerte durch maschinelles Lernen und Mustererkennung geschätzt. Die darauffolgende Fusion berücksichtigt die Zuverlässigkeiten, um die verfügbaren Hypothesen bestmöglich zu kombinieren. Der aus der Fusion entstandene Fahrstreifenverlauf wird schließlich mithilfe eines Kalman-Filters stabilisiert. Zusammenfassend dienen die entwickelten Methoden als Ergänzung zur modellbasierten Hypothesengenerierung der Fahrbahnerkennung.

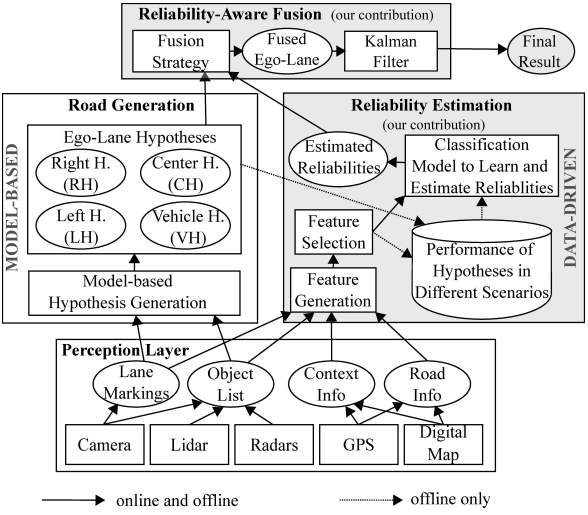


Bild 3: Systemarchitektur zur Integration von Zuverlässigkeiten in die Fusion unterschiedlicher Ego-Lane-Hypothesen

Anhand der kamerabasierten Markierungserkennung und der Objektdetektion aus Radar- und Lidar-Sensoren können in dieser Arbeit vier Hypothesen über den Verlauf des eigenen Fahrstreifens (*ego-lane*) gebildet werden. Die *Right Hypothesis (RH)* und *Left Hypothesis (LH)* basieren auf den Markierungen der jeweiligen Seite. Die *Center Hypothesis (CH)* bezieht beide Markierungen ein und liefert einen gemittelten Verlauf. Auf Basis von Position und Bewegung des Vorderfahrzeugs wird darüber hinaus die *Vehicle Hypothesis (VH)* generiert. Aus den Informationen über Fahrbahnmarkierungen, andere Fahrzeuge und den aktuellen Standort werden Merkmale extrahiert, welche dann zum Trainieren von Klassifikatoren verwendet werden. Dabei umfassen die Eingangsdaten den möglichen Fahrbahnverlauf aller Hypothesen als eine approximative Form einer Klothoide ($y(x) = y_0 + \phi x + \frac{1}{2}c_0x^2 + \frac{1}{6}c_1x^3$ mit l_x (Länge), x_0 (Beginn auf der x-Achse), y_0 (anfängliche laterale Verschiebung), ϕ (Winkel zur x-Achse), c_0 (Krümmung), c_1 (Krümmungsänderung)) sowie deren Abweichungen zueinander. Die offline gelernten Klassifikationsmodelle können anschließend genutzt werden, um die Zuverlässigkeit der Hypothesen online zu bewerten.

Die Herausforderung bei der anschließenden Fusion liegt darin, dass keine der Hypothesen in allen Szenarien eine gute Performanz bietet. Eine reine Mittelwertbildung führt dabei nur zu mäßigen Ergebnissen, weil fehlerhafte Hypothesen einzelner Sensoren das Ergebnis verfälschen. Daher soll sich die Fusion auf die zuverlässigen Quellen beschränken mit dem Ziel, die Querregelung zu verbessern und die Verfügbarkeit der automatischen Fahrfunktion in allen Szenarien zu steigern.

Dazu werden im weiteren Verlauf dieser Arbeit folgende Fragestellungen diskutiert:

- Was ist unter der Zuverlässigkeit einer Quelle zu verstehen und mit welchem Maß lässt sich diese quantifizieren?
- Wie kann die Zuverlässigkeit einer Quelle aus bestimmten Merkmalen gelernt werden?
- Welches Verfahren ist geeignet, eine Fusion der Informationen aus verschiedenen Quellen unter Berücksichtigung ihrer jeweiligen Zuverlässigkeit durchzuführen?

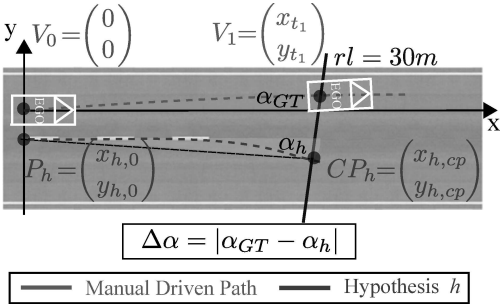


Bild 4: Beurteilung der Zuverlässigkeit einer Hypothese anhand ihres Winkels zur manuell gefahrenen Trajektorie

4 Zuverlässigkeit

Zuverlässigkeit ist nicht einfach ein Maß für die Messunsicherheit eines Sensors, also eine statistische Abweichung, sondern bildet auch den systematischen Fehler eines Sensors in bestimmten Szenarien ab. Daher spricht man auch von einer *Higher-Level Uncertainty* oder *stability of the first-order uncertainty* [3, 16]. Viele Sensoren stellen neben ihren Messwerten auch einen Zuverlässigkeitswert zur Verfügung, der allerdings nicht immer zutreffend ist. Zudem wenden verschiedene Hersteller unterschiedliche Berechnungsverfahren zur Ermittlung sowie verschiedene Skalen zur Darstellung der Zuverlässigkeit von Detektionen an. Daher sind die von unterschiedlichen Sensoren erhaltenen Werte nicht direkt miteinander vergleichbar.

Bisherige Arbeiten zur Fusion nutzen daher unterschiedliche Methoden, um die Zuverlässigkeit der zu fusionierenden Quellen zu schätzen [2]. Ein Vergleich der Klothoiden und ihrer Parametern erlaubt es dabei, vom Konsens abweichende Hypothesen zu erkennen und deren Zuverlässigkeit als geringer zu bewerten [17]. Eine weitere Möglichkeit ist der Vergleich der Parameter mit Kartenmaterial [10], wobei insbesondere der laterale Offset und der Winkel zwischen Fahrzeug und Fahrstreifen verwendet werden. Die notwendigen hochgenauen Karten sowie die präzise Lokalisierung sind jedoch mit hohen Kosten verbunden.

Nguyen et al. präsentierten einen Ansatz, der die von den Sensoren gelieferten Hypothesen mit der manuell gefahrenen Trajektorie vergleicht [18]. Wie Bild 4 zeigt, wird zunächst der in einer bestimmten Entfernung *rl* (run length) auf der Hypothese liegende Punkt ermittelt. Nachdem

die Entfernung rl zurückgelegt wurde, wird die Winkeldifferenz $\Delta\alpha$ zwischen der berechneten α_h und der tatsächlich gefahrenen Trajektorie α_{GT} bestimmt mit

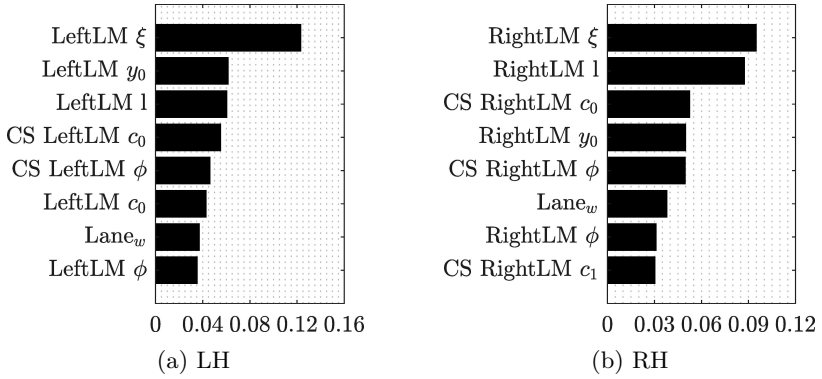
$$\begin{aligned}\Delta\alpha &= |\alpha_h - \alpha_{GT}| \\ &= \left| arctan\left(\frac{y_{h,cp} - y_{h,0}}{x_{h,cp} - x_{h,0}}\right) - arctan\left(\frac{y_{t_1}}{x_{t_1}}\right)\right|\end{aligned}$$

Ein kleines $\Delta\alpha$ impliziert eine hohe Übereinstimmung bezüglich der Parallelität zwischen der Hypothese und der Referenz.

5 Zuverlässigkeitsschätzung mit überwachtem Lernen

Dieser Abschnitt wendet mehrere Klassifikationsverfahren aus dem Bereich des überwachten Lernens (*supervised learning*) an, um die Zuverlässigkeit der Hypothesen aus den extrahierten Features zu ermitteln. Für jede Hypothese $h \in H$ bestehen die Features aus drei Mengen: sensorrelevante Merkmale s_h , Konsens- τ_h und Kontextfeatures γ. Als s_h werden dabei unter anderem die Markierungen repräsentierenden Klothoidenparameter und der vom Sensor bereitgestellte Existenzwert ξ verwendet. Im Gegensatz zu der Positionsunsicherheit, repräsentiert ξ die sensoreigene Einschätzung der Zuverlässigkeit, etwa ob das Objekt wirklich existiert. Anhand der Erkennung des Vorderfahrzeugs werden darüber hinaus beispielsweise auch dessen Geschwindigkeit und Gierrate berücksichtigt. Außerdem beschreibt τ_h die Übereinstimmung zwischen den Hypothesen. Des Weiteren wird γ durch Informationen aus der Karte gebildet, wie Straßentyp (Autobahn, Landstraße, Ausfahrt, Urban etc.), Fahrbahntyp (Normal, Split, Merge etc.), Geschwindigkeitsbegrenzung etc. Insbesondere ist der Straßentyp von Bedeutung, da die Performanz vieler Sensoren auf Autobahnen und im städtischen Bereich sehr unterschiedlich ist.

Aufgabe der überwachten Lernverfahren ist es, die Hypothesen anhand dieser Features einer der beiden Klassen *Reliable* oder *Unreliable* zuzuweisen. Folglich soll eine binäre Entscheidung über die Zuverlässigkeit jeder Hypothese getroffen werden. Je nach Art des Lernverfahrens wird aus dem Klassifikationsergebnis anschließend ein Zuverlässigkeitswert $R_h \in [0, 1]$ berechnet. Ein wesentlicher Vorteil dieses Verfahrens liegt darin, dass auf alle Hypothesen, unabhängig von Art und Hersteller der verwendeten Sensoren, das gleiche Zuverlässigkeitsmaß angewendet wird. Wie Bild 5 darstellt, ist der Einfluss der einzelnen Merkmale auf



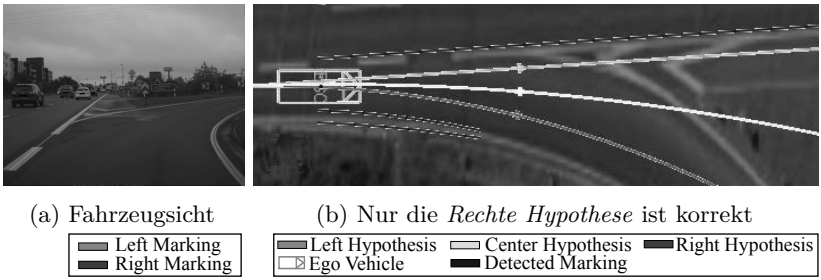
(a) LH (b) RH

Bild 5: Feature Importance zur Zuverlässigkeitsschätzung der Ego-Lane-Hypothesen *LH* und *RH*

das Klassifikationsergebnis sehr unterschiedlich. Die Zuverlässigkeit der Hypothesen *LH* und *RH*, die entweder auf den links oder rechts gelegenen Markierungen basieren, kann insbesondere anhand des Existenzwerts ξ sowie den Parametern Länge l und Offset y_0 der Klothoide beurteilt werden [19]. Für die Zuverlässigkeit der CH ist hingegen die geschätzte Breite des Fahrstreifens das aussagekräftigste Merkmal. Dies weist darauf hin, dass ungewöhnliche Schätzungen der Fahrstreifenbreite oftmals auf eine fehlerhafte Erkennung einer der Markierungen oder die Kombination nicht zusammengehöriger Markierungen wie in Bild 6 zurückzuführen sind. In diesen Fällen ist die zentrale Hypothese *CH* fehlerhaft, kann aber offenbar anhand der geschätzten Fahrstreifenbreite auch als solche identifiziert werden. Im Anschluss daran wird eine weitere Dimensionsreduktion mittels einer *Principal Component Analysis* angewendet, um redundante Features zu eliminieren. Dadurch soll der reduzierte Merkmalsraum eine verbesserte Klassifikationsperformanz ermöglichen. Anschließend werden drei ausgewählte Klassifikationsverfahren eingesetzt, um die Zuverlässigkeiten aus Trainingsdaten zu lernen.

5.1 Erprobte Verfahren

Bayessches Netz (BN) : *BN* findet in zahlreichen Bereichen Anwendung [20]. Dabei bietet *BN* die Möglichkeit, verschiedenste Arten von Informationen zu modellieren, z.B. Expertenwissen, Verteilung von Testdaten etc. In dieser Arbeit wird ein Bayessches Netz erstellt, bei dem jeder



(a) Fahrzeugsicht (b) Nur die *Rechte Hypothese* ist korrekt

Bild 6: Autobahn-Auffahrt mit erkannten Markierungen und den geschätzten Hypothesen (unter Verwendung des *Automotive Data and Time-Triggered Frameworks und Google Maps*^{TM})

Knoten ein ausgewähltes relevantes Merkmal repräsentiert. Dabei wird die Annahme getroffen, dass die Zuverlässigkeiten mehrerer möglicherweise abhängiger Quellen unabhängig voneinander sind, um die Inferenz zu vereinfachen. Die Zuverlässigkeit jeder Hypothese h ergibt sich aus der bedingten Wahrscheinlichkeit des Knotens L_h für die gegebene Evidenz: $R_h = P(L_h = Reliable|s_h, \tau_h, \gamma)$.

Random Forest (RF) : Jeder *Random Forest* besteht aus einer Anzahl N von Entscheidungsbäumen, die jeweils mit einer unterschiedlichen Teilmenge von Merkmalen und Trainingsdaten gebildet werden [21]. Jeder Baum beurteilt eine Hypothese als zuverlässig oder unzuverlässig. Anschließend wird das Ergebnis des *RF*s per Mehrheitsvotum der Bäume bestimmt. Das Maß der Zuverlässigkeit wird schließlich anhand der Anzahl N_R der Bäume, die die Hypothese als zuverlässig bewerten, mit $R_h = \frac{N_R}{N}$ berechnet.

k-Nearest Neighbor (kNN) : Bei *kNN* erfolgt die Bewertung anhand benachbarter Punkte in dem durch den Merkmalsvektor aufgespannten Raum. Nach dem Prinzip der Mehrheitsentscheidung berechnet sich die Zuverlässigkeit nach $R_h = \frac{k_R}{k}$ als Anteil der Nachbarpunkte k_R, die die Hypothese als zuverlässig bewerten, an allen Nachbarpunkten k.

6 Zuverlässigkeitsbasierte Fusion

Unter Berücksichtigung der ermittelten Zuverlässigkeiten stellt dieser Abschnitt verschiedene Fusionsstrategien vor.

Mittelwert-Fusion (AVG) : Die *Mittelwert-Fusion* ist der einfachste Ansatz mit der Annahme, dass jede Hypothese mit gleichem Gewicht in der Fusion berücksichtigt wird. Dieser Ansatz wird schlechte Resultate liefern, da die unzuverlässige Hypothesen das Fusionsergebnis negativ beeinflussen können.

Gewichtete Fusion (WBF) : Im Gegensatz zu *AVG*, werden bei der *gewichteten Fusion* die Hypothesen anhand ihres Zuverlässigkeitswerts anteilig berücksichtigt. Dadurch können bessere Ergebnisse erzielt werden, indem *WBF* die zuverlässigeren Quellen den Schlechteren vorzieht.

Winner-Takes-All (WTA) : Mit dem Ansatz *Winner-Takes-All* wird kein durchschnittlicher Wert berechnet, sondern nur die zuverlässigste Hypothese ausgewählt. Bei gleichem R_h mehrerer Hypothesen wird eine der Hypothesen nach der Rangfolge $CH > LH > RH > VH$ ausgewählt. Dies ergibt sich aus der Tatsache, dass die zentrale Hypothese *CH*, die aus zwei Markierungen konstruiert wird, das stabilste Fahrverhalten ermöglicht. Die *VH* wird als letztes berücksichtigt, da der Vordermann den Fahrstreifen wechseln kann. Dies kann bei einer Folgefahrt zu einem unerwünschten Fahrstreifenwechsel des Ego-Fahrzeugs führen.

7 Ergebnisse

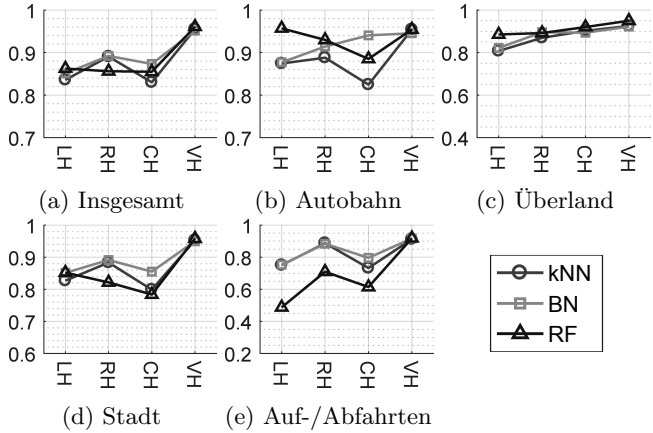
7.1 Auswertung der Zuverlässigkeitsschätzung

Die Klassifikatoren werden mit 70% des Datensatzes trainiert und mit dem Rest validiert. Der Datensatz besteht aus ungefähr 25% Autobahnen, 50% Stadt, 15% Überland und 10% Auf- und Abfahrten.

Wie Tabelle 1 zeigt, beträgt die a priori Wahrscheinlichkeit für eine verlässliche Hypothese h ($\Delta\alpha_h < 2°$) im Fall von *LH* in einem Autobahnszenario 99%. Nur durch die Vorhersage, dass alle Daten *Reliable* sind, kann dadurch schon ein *F-Score* von 99% erreicht werden. Um dem Klassenungleichgewicht zwischen *Reliable* und *Unreliable* vorzubeugen und ein einheitliches Maß zu erstellen, wird ein Down-Sampling der Daten durchgeführt, bevor die Klassifikatoren trainiert und getestet werden.

Tabelle 1: A priori Wahrscheinlichkeit für eine zuverlässige Hypothese h ($\Delta\alpha_h < 2°$) für jeden Straßentyp

	Gesamt	Autobahn	Überland	Stadt	Auf-/Abfahrten
$P(C_{LH} = R)$	87.0%	99.3%	97.6%	80.9%	63.3%
$P(C_{RH} = R)$	92.0%	99.3%	97.8%	88.1%	82.2%
$P(C_{CH} = R)$	83.7%	99.0%	97.9%	76.6%	50.6%
$P(C_{VH} = R)$	54.5%	70.7%	52.2%	49.5%	40.0%



(a) Insgesamt (b) Autobahn (c) Überland

(d) Stadt (e) Auf-/Abfahrten

Bild 7: Klassifikationsergebnisse gemessen mit $F_{0.8}$-Score

Zur Bewertung der Zuverlässigkeit R_h der Hypothesen muss ein Schwellwert bestimmt werden, mit dem eine optimale Klassenzuweisung ($R_h \geq \epsilon_R \rightarrow h$ ist *Reliable*) erzielt wird. Dafür wird der gewichtete $F_{0.8}$-Score aus *Precision (PR)* und *Recall (RC)*: $F_\beta = \frac{(1+\beta^2)\cdot PR\cdot RC}{(\beta^2\cdot PR)+RC}$ berechnet. Hierbei ist die *Precision* von höherer Bedeutung als der *Recall*, da für automotive Anwendungen die Sicherheit im Fokus liegt. Mit dem Ziel den $F_{0.8}$-Score für die Schätzung der jeweiligen R_h zu maximieren, wird für jede Hypothese h ein Schwellwert ϵ_R bestimmt.

Gemessen an den $F_{0.8}$-Scores, vergleicht Bild 7 die Klassifikationsergebnisse für alle Hypothesen in unterschiedlichen Szenarien. In der Bild 7a ist zu erkennen, dass *BN* und *RF* über alle Daten für jeweils zwei Hypothesen die besten Ergebnisse erzielen. Obwohl *kNN* ähnliche Ergebnisse erreicht, ist der hohe Zeitaufwand bei der Suche kritisch. Auf

der Autobahn (Bild 7b) und auf Überlandstraßen (Bild 7c) erreichen *RF* und *BN* abwechselnd die besten Ergebnisse. Für Stadt (Bild 7d) und Auf-/Abfahrten (Bild 7e), schneiden *BN* und *kNN* besser ab. Dieses Ergebnis ist besonders interessant, da die letzten beiden Szenarien die höchste Komplexität aufweisen.

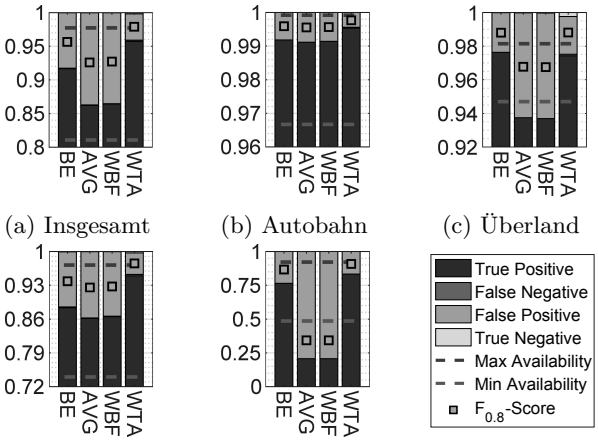
Da *BN* stabile Ergebnisse in allen Szenarien aufweist, wird *BN* als der darunterliegende Zuverlässigkeitsschätzer für die Fusion verwendet.

7.2 Auswertung der Fusion

Dieser Abschnitt beschäftigt sich mit der Auswertung der vorgestellten Fusionsstrategien. Sollte das Fusionsergebnis einen geschätzten Fahrstreifenverlauf O ausgeben, wird dieser als positives Sample gewertet. Falls $\Delta\alpha$ von O im Vergleich zu der Referenz kleiner als 2° ist, wird O als *True Positive (TP)* gewertet. Andernfalls wird es als *False Positive (FP)* eingeordnet. Des Weiteren ist es möglich, dass die Fusionsstrategie keine der Hypothesen als *Reliable* einstuft und somit alle Hypothesen aus der Fusion exkludiert. Dieser Fall wird als *False Negative (FN)* oder *True Negative (TN)* gewertet abhängig davon, ob mindestens eine zuverlässige Hypothese existiert. Basierend darauf ergibt sich die Verfügbarkeit AV mit $AV = \frac{TP}{TP+FP+FN+TN}$, die den Anteil der Daten beschreibt, in dem eine automatische Fahrfunktion möglich ist.

Mit *BN* als Zuverlässigkeitsschätzer, vergleicht Bild 8 die Performanz unterschiedlicher Fusionsstrategien. Es wird kein Down-Sampling der Daten durchgeführt, um die tatsächliche Performanz unter den gegebenen Straßenbedingungen wiederzugeben. Außerdem wird zum Vergleich das Ergebnis der Arbeit von [15] als Baseline (*BE*) herangezogen.

Bild 8a illustriert, dass *WTA* im Vergleich zu *BE* insgesamt eine Steigerung der Verfügbarkeit von 3% erzielt. Auf der Autobahn erbringen alle Fusionsverfahren annähernd gleiche Resultate (Bild 8a und 8b). In den übrigen Szenarien mit höherer Komplexität schneiden erwartungsgemäß *AVG* und *WBF* besonders schlecht ab, da die unzuverlässigen Hypothesen das Fusionsergebnis verschlechtern. Die restlichen Verfahren bewegen sich auf ähnlichem Niveau mit Ausnahme des Szenarios Stadt, in welchem *WTA* eine deutliche Steigerung zu *BE* darstellt (Bild 8d). Für das Szenario Auf- und Abfahrten erreicht die Strategie *WTA* ebenfalls eine Verbesserung gegenüber *BE*, welche lediglich die Geometrie und die Positionsunsicherheit der Hypothesen betrachtet (Bild 8e).



(a) Insgesamt (b) Autobahn (c) Überland

(d) Stadt (e) Auf-/Abfahrten

Bild 8: Performanz unterschiedlicher Fusionsstrategien

8 Zusammenfassung und Ausblick

Diese Arbeit präsentiert eine zuverlässigkeitsbasierte Fusion von Fahrstreifenschätzungen. Dabei werden viele Aspekte der Zuverlässigkeit diskutiert, wie die Feature-Auswahl oder Lernen mittels Klassifikatoren. Die geschätzten Zuverlässigkeiten werden bei der Fusion verwendet, um unzuverlässige Hypothesen für die endgültige Fusion zu verwerfen. Verglichen mit dem Ansatz von [22] ermöglicht der vorgestellte Ansatz eine Verbesserung der Verfügbarkeit von 7% für den Stadtverkehr und 6% für Ab- bzw. Auffahrten. In weiterführenden Arbeiten werden die Krümmung einer Standard-Navigationskarte und der geschätzte freie Raum mit in den Merkmalsvektor eingehen. Weiterhin können die Bildinformationen oder die Beziehungen zwischen Hypothesen in Betracht gezogen werden.

Literatur

[1] A. Bar Hillel, R. Lerner, and D. Levi, "Recent progress in road and lane detection: a survey," *Machine Vision and Applications*, 2014.

[2] T. T. Nguyen, J. Spehr, M. Uhlemann, S. Zug, and R. Kruse, "Learning of lane information reliability for intelligent vehicles," in *IEEE Multisensor Fusion and Integration for Intelligent Systems*, 2016.

[3] G. L. Rogova and V. Nimier, "Reliability in information fusion: literature survey," in *Information Fusion*, 2004.

[4] F. E. White, Naval Ocean Systems Center . Architecture, and A. R. Branch, *Data Fusion Lexicon*. Naval Ocean Systems Center, 1986.

[5] Albrecht Klotz, Jan Sparbert, and Dieter Hoetzer, "Lane Data Fusion for Driver Assistance Systems," *Information Fusion*, 2004.

[6] C. Gackstatter, P. Heinemann, S. Thomas, B. Rosenhahn, and G. Klinker, "Fusion of clothoid segments for a more accurate and updated prediction of the road geometry," in *IEEE Intelligent Transportation Systems*, 2010.

[7] Á. F. García-Fernández, M. Fatemi, and L. Svensson, "Bayesian Road Estimation Using Onboard Sensors," *IEEE Transactions on Intelligent Transportation Systems*, 2014.

[8] T. Brade, S. Zug, and J. Kaiser, "Validity-Based Failure Algebra for Distributed Sensor Systems," in *IEEE Symposium on Reliable Distributed Systems*, 2013.

[9] Chunzhao Guo, J. Meguro, Y. Kojima, and T. Naito, "CADAS: A multimodal advanced driver assistance system for normal urban streets based on road context understanding," in *IEEE Intelligent Vehicles,*, 2013.

[10] O. Hartmann, M. Gabb, and K. Dietmayer, "Towards autonomous self-assessment of digital maps," in *IEEE Intelligent Vehicles*, 2014.

[11] H. Frigui, L. Zhang, and P. Gader, "Context-Dependent Multi-Sensor Fusion for Landmine Detection," in *Geoscience and Remote Sensing Symposium, IEEE*, 2008.

[12] A. R. Romero, P. V. K. Borges, A. Elfes, and A. Pfrunder, "Environment-aware sensor fusion for obstacle detection," in *IEEE Multisensor Fusion and Integration for Intelligent Systems*, 2016.

[13] M. Realpe, B. Vintimilla, and L. Vlacic, "Towards fault tolerant perception for autonomous vehicles: Local fusion," in *IEEE Conference on Cybernetics and Intelligent Systems and Conference on Robotics, Automation and Mechatronics*, 2015.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ser. Adaptive Computation and Machine Learning. MIT Press, 2016.

[15] D. Toepfer, J. Spehr, J. Effertz, and C. Stiller, "Efficient Road Scene Understanding for Intelligent Vehicles Using Compositional Hierarchical Models," *IEEE Transactions on Intelligent Transportation Systems*, 2015.

[16] P. Wang, "Confidence as Higher-Order Uncertainty," in *ISIPTA*, 2001.

[17] F. Delmotte, L. Dubois, and P. Borne, "Context-dependent trust in data fusion within the possibility theory," in *Systems, Man, and Cybernetics, IEEE*, 1996.

[18] T. T. Nguyen, J. Spehr, J. Xiong, M. Baum, S. Zug, and R. Kruse, "A Survey of Performance Measures to Evaluate Ego-Lane Estimation and A Novel Sensor-Independent Measure Along with Its Applications," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2017.

[19] ——, "Online Reliability Assessment and Reliability-Aware Fusion for Ego-Lane Detection Using Influence Diagram and Bayes Filter," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2017.

[20] R. Kruse, C. Braune, C. Borgelt, and S. Mostaghim, *Computational Intelligence: A Methodological Introduction.* Springer London, 2016.

[21] L. Breiman, "Random Forests," *Machine Learning*, no. 1, 2001.

[22] D. Töpfer, J. Spehr, J. Effertz, and C. Stiller, "Efficient scene understanding for intelligent vehicles using a part-based road representation," in *IEEE Intelligent Transportation Systems*, 2013.

Actor-Critic Reinforcement-Learning in der Anwendung am Beispiel einer schweren Kette

Christian Dengler, Boris Lohmann

Technische Universität München
E-Mail: c.dengler@tum.de

1 Einführung

Der klassische Reglerentwurf benötigt in der Regel zum einen ein möglichst genaues Modell der zu regelnden Strecke, und zum anderen Kenntnisse in der Regelungstechnik. Als Alternative gibt es Algorithmen zum Erlernen eines Reglers, basierend auf einem Gütemaß sowie einem Simulationsmodell oder alternativ auch Messdaten der realen Strecke. Diese Algorithmen werden unter dem Begriff Reinforcement-Learning zusammengefasst. Die Algorithmen sind in den letzten Jahren von großen Erfolgen in Brett- und Computerspielen geprägt [1][2], ähnlich herausragende Erfolge in der Regelungstechnik blieben bisher allerdings aus. In diesem Beitrag wird ein solcher Algorithmus in der Praxis am Beispiel einer schweren Kette angewendet und mit einem analytischen nichtlinearen Regler verglichen. Dabei werden auch auf die Schwierigkeiten, welche bei der Anwendung dieses Algorithmus auftreten können, beschrieben.

Es werden erst kurz die grundlegenden Bausteine vieler Reinforcement-Learning-Algorithmen in Abschnitt 2 erklärt, woraufhin in Abschnitt 3 der tatsächlich verwendete Algorithmus beschrieben wird. Anschließend wird in Abschnitt 3 die Problemstellung beschrieben und Ergebnisse in der Simulation werden in Abschnitt 5 gezeigt. In Abschnitt 6 wird die Anwendung am realen System beschrieben.

2 Grundlagen des Reinforcement-Learning

Die Grundlagen des Reinforcement-Learning gehen auf Markov-Entscheidungsprozesse zurück und werden im Folgenden anhand dieser erläutert. Die hier vorgestellten Grundlagen finden sich detaillierter in etwas anderer Notation in [3].

Ein Markov-Entscheidungsprozesse wird beschrieben durch den Tupel $(\mathcal{X}, \mathcal{U}, \mathcal{P}, \mathcal{R}, d_0)$, wobei $\mathcal{X}$ die diskreten Zustände, $\mathcal{U}$ die diskrete Menge an verfügbaren Aktionen, $\mathcal{P}$ die Übergangswahrscheinlichkeiten zwischen den Zuständen, $\mathcal{R}$ die Belohnung der Übergänge, sowie d_0 die Anfangsverteilung darstellt. Ziel ist es, die Belohnung durch eine gute Wahl der Aktionen zu maximieren. Die Strategie nach welcher die Aktionen gewählt werden, kann entweder stochastischer Natur sein und durch eine Wahrscheinlichkeitsverteilung $\pi(u|x)$ beschrieben werden, oder deterministisch als $u = \pi(x)$ vorliegen.

Bevor darauf eingegangen wird, wie die optimale Strategie gefunden werden kann, wird zunächst die Werte-Funktion V, sowie Aktionswerte-Funktion Q eingeführt, welche die Grundbausteine vieler Reinforcement-Learning-Algorithmen darstellen. Die Wertefunktion V beschreibt, wie viele Belohnung in Zukunft unter der aktuellen Strategie erwartungsgemäß erhalten wird, ausgehend von einem Zustand $x \in \mathcal{X}$. Mit $r_t \in \mathcal{R}$ der Belohnung zum diskreten Zeitpunkt t, $\mathbf{E}^\pi$ der Erwartungswert unter der Strategie π sowie $p_{xx'}$ der Übergangswahrscheinlichkeit vom Zustand x zu x' unter π, kann V beschrieben werden als

$$V^\pi(x) = \mathbf{E}^\pi \left\{ \sum_{t=0}^{T_{end}} r_t | x_0 = x \right\} \tag{1}$$

$$= \mathbf{E}^\pi \left\{ r_0 + \sum_{t=1}^{T_{end}} r_t | x_0 = x \right\} \tag{2}$$

$$= \mathbf{E}^\pi \left\{ r_0 + \sum_{x' \in X} p_{xx'} V^\pi(x') | x_0 = x \right\}. \tag{3}$$

Sollte es keinen Endzustand geben, und somit kein End-Zeitpunkt T_{end}, so ist die Funktion möglicherweise unbeschränkt, weshalb in diesem Fall ein Skalierungsfaktor $\gamma \in [0, 1)$ für zukünftige Belohnungen eingeführt werden muss, und die Wertefunktion somit lautet

$$V^\pi(x) = \mathbf{E}^\pi \left\{ \sum_{t=0}^{\infty} \gamma^t r_t | x_0 = x \right\} \tag{4}$$

$$= \mathbf{E}^\pi \left\{ r_0 + \gamma \sum_{x' \in X} p_{xx'} V^\pi(x') | x_0 = x \right\}. \tag{5}$$

Im Folgenden werden nur noch Fälle ohne Endzeitpunkt betrachtet.

Die Aktionswerte-Funktion Q beschreibt die zu erwartende Belohnung ausgehend vom Zustand x, falls im aktuellen Zeitschritt die Aktion u gewählt wird, und danach alle weiteren Aktionen mit der aktuellen Strategie gewählt werden. Der Unterschied zur Wertefunktion liegt somit nur in der Erweiterung der zum aktuellen Zeitpunkt zu wählenden Aktion. Es gilt die Beziehung

$$V^{\pi}(x) = \mathbf{E}\left\{Q^{\pi}(x, \pi(u|x))|x = x\right\} \tag{6}$$

$$= \sum_{u \in U} \pi(u|x) Q(x, u), \tag{7}$$

bzw. im Falle einer deterministischen Strategie entfällt der Erwartungswert und es gilt

$$V^{\pi}(x) = Q^{\pi}(x, \pi(x)). \tag{8}$$

Mit $p^{u}_{xx'}$ der Übergangswahrscheinlichkeit vom Zustand x zu x' wenn u gewählt wird, kann Q beschrieben werden als

$$Q^{\pi}(x, u) = \mathbf{E}^{\pi}\left\{\sum_{t=0}^{\infty} \gamma^{t} r_t | x_0 = x, u_0 = u\right\} \tag{9}$$

$$= \mathbf{E}^{\pi}\left\{r_0 + \gamma \sum_{x' \in X} p^{u}_{xx'} Q^{\pi}(x', u') | x_0 = x, u_0 = u\right\} \tag{10}$$

Für Markov-Entscheidungsprozesse ohne versteckte Zustände gibt es immer eine deterministische optimale Strategie π^*, für welche die Bellmann-Gleichung

$$Q^{\pi^*}(x, u) = \mathbf{E}\left\{r_t + \gamma \max_{u'} Q^{\pi^*}(x', u')\right\} \tag{11}$$

gilt. Das Optimum lässt sich als Fixpunkt durch iterative Anwendung von (11) auf alle Zustände ermitteln [3]. Die optimale Policy ist danach implizit in $Q^{\pi^*(x)}$ enthalten als

$$\pi^*(x) = \arg\max_{u} Q^{\pi^*(x)}(x, u). \tag{12}$$

Für die Berechnung der Erwartungswerte müssen alle Wahrscheinlichkeiten im Markov-Prozess bekannt sein. Dies ist für die meisten realen Probleme nicht möglich oder zu aufwendig. Verwendet man statt einem bekannten Markov-Entscheidungsprozess nun Datenwerte, so redet man vom Reinforcement-Learning. Die Basis für die Auslegung der Strategie

ist also nicht mehr ein Markov-Entscheidungsprozess, sondern Datenpunkte der Form (x_t, u_t, r_t, x_{t+1}), mit u_t der Aktion, welche von x_t nach x_{t+1} führte. Diese Tupel können als Stichproben verwendet werden, um die Bellmann-Gleichungen approximativ zu lösen. Hier gibt es nun verschiedene Algorithmen und Verfahren, je nachdem ob man diskrete oder kontinuierliche Aktions- und/oder Zustandsräume betrachtet, ob die Strategie explizit gespeichert wird, ob zusätzlich ein Modell auf den Daten trainiert wird usw. Der Übergang von diskretem zu kontinuierlichem Zustandsraum wird über Funktions-Approximatoren, zum Beispiel neuronale Netze, gelöst. Leider kann durch die Verwendung solcher Approximatoren die Konvergenz oft nicht mehr sichergestellt werden, und es werden weitere Freiheitsgrade hinzugefügt, wie zum Beispiel die Anzahl an Parametern oder die Lernrate. Diese Approximation erlaubt jedoch die praktische Anwendung auf eine sehr breite Klasse an dynamischen Systemen.

Da es sich in der Regelungstechnik meistens um Probleme mit kontinuierlichen Zustands- und Aktionsräumen handelt, wird im Folgenden ein Actor-Critic Algorithmus implementiert, welcher eine deterministische Strategie (einen deterministischen Regler) trainiert.

3 Actor-Critic mit deterministischem Policy-Gradienten

Als Actor-Critic Algorithmus beschreibt man Algorithmen, welche sowohl die Aktionswerte-Funktion Q, also auch die Strategie π explizit als abgespeicherte Funktion verwenden. Die Strategie soll also nicht implizit aus Q abgeleitet werden. Dies ist von Vorteil, da für (12) der arg max-Operator verwendet werden muss. Hierbei kann es sich um ein nicht-konvexes hochdimensionales Optimierungsproblem handeln, weshalb es für eine schnelle Auswertung der Strategie sinnvoll ist, sie einzeln im Speicher als alleinstehende Funktion auswerten zu können [9]. Im Folgenden werden nun die Zustände kontinuierlich $\mathbf{x} \in \mathcal{X}$ mit $\mathcal{X} \subset \mathbb{R}^n$ sowie die Aktionen ebenfalls kontinuierlich als $\mathbf{u} \in \mathcal{U}$ mit $\mathcal{U} \subset \mathbb{R}^m$ angenommen. Des weiteren werden die Übergänge zwischen den Zuständen weiterhin als stochastisch, die Strategie jedoch als deterministisch angenommen.

Die Strategie oder Regler wird im vorgestellten Algorithmus über den Gradienten der Gütefunktion bezüglich der Parameter $\boldsymbol{\theta}$ der Strategie $\pi_{\boldsymbol{\theta}}$ [4] trainiert. Der Notation in [4] folgend sei $d(\mathbf{x} \to \mathbf{x}', t, \pi)$ die Wahrscheinlichkeitsdichte im Zustand $\mathbf{x}'$ nach t Zeitschritten, ausgehend vom

Zustand $\mathbf{x}$. Die Gütefunktion $J(\pi_{\boldsymbol{\theta}}) = \mathbf{E}^{\pi_{\boldsymbol{\theta}}} \{\sum_{t=0}^{\infty} \gamma^t r_t | x_0 \sim d_0\}$ kann unter Verwendung einer deterministischen Strategie sowie von $\rho^{\pi}(\mathbf{x})$ als kumulierte, über die Zukunft gewichtete Zustandsverteilungen

$$\rho^{\pi}(\mathbf{x}) = \int_X \sum_{t=0}^{\infty} \gamma^t d_0(\mathbf{x}') d(\mathbf{x}' \to \mathbf{x}, t, \pi) d\mathbf{x}' \tag{13}$$

auch dargestellt werden als

$$J(\pi_{\boldsymbol{\theta}}) = \int_X \rho^{\pi}(\mathbf{x}) r(\mathbf{x}, \pi_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x}. \tag{14}$$

und es wird in [4] gezeigt, dass der Gradient bezüglich des Parametervektors der Strategie angegeben werden kann als

$$\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) = \int_X \rho^{\pi}(x) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(x) \nabla_{\mathbf{u}} Q^{\pi}(x, u)|_{u=\pi_{\boldsymbol{\theta}}(x)} dx \tag{15}$$

$$= \mathbf{E}_{x \sim \rho^{\pi}} \left\{ \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(x) \nabla_{\mathbf{u}} Q^{\pi}(x, u)|_{u=\pi_{\boldsymbol{\theta}}(x)} \right\}. \tag{16}$$

Der Erwartungswert in (16) kann über Samples $\mathbf{x} \sim \rho^{\pi}(\mathbf{x})$ approximiert werden, falls die Aktionswerte-Funktion Q bekannt ist. Die Funktion Q wird ebenfalls über Parameter approximiert, welche über Standard-Verfahren wie z.B. TD-Learning [3] angepasst werden können. Es gilt noch zu erwähnen, dass 1) für eine gute Approximation von $Q(\mathbf{x}, \mathbf{u})$ möglichst viele Zustände besucht werden müssen, vor allem auch für $\mathbf{u} \neq \pi_{\boldsymbol{\theta}}(\mathbf{x})$. Deshalb ist es sinnvoll für das Trainieren eine zusätzliche Abweichung auf die Stellgröße u aufzubringen. Insbesondere für träge Systeme ist es sinnvoll kein weißes Rauschen, sondern korreliertes Rauschen aufzubringen, um die Zuständsänderungen trotz der Trägheit im System zu erlernen.

Die Aktionswertefunktion wird zerlegt in die Summe $Q(\mathbf{x}, \mathbf{u}) = V(\mathbf{x}) + A(\mathbf{x}, \mathbf{u})$ aus Wertefunktion und Vorteilsfunktion [6], um die in [4] vorgeschlagenen kompatiblen Funktionsapproximatoren realisieren zu können, sodass trotz Approximation von Q kein Bias in (16) entsteht. Die drei Funktionen V, A und π werden als linear in ihren Parametern approximiert als:

$$V_{\boldsymbol{\nu}}(\mathbf{x}) = \sum_{i=1}^{N} \phi_i(\boldsymbol{x})\nu_i = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\nu} \tag{17}$$

$$\begin{aligned} A_{\boldsymbol{\Omega}}(\mathbf{x}, \mathbf{u}) &= (\mathbf{u}_t - \pi_{\boldsymbol{\Theta}}(\mathbf{x}_t))^T \sum_{i=1}^{N} \phi_i(\boldsymbol{x})\boldsymbol{\omega}_i \\ &= (\mathbf{u}_t - \pi_{\boldsymbol{\Theta}}(\mathbf{x}_t))^T \boldsymbol{\Omega}\boldsymbol{\phi}(\mathbf{x}) \end{aligned} \tag{18}$$

$$\pi_{\boldsymbol{\Theta}}(\mathbf{x}) = \sum_{i=1}^{N} \phi_i(\boldsymbol{x})\boldsymbol{\theta}_i = \boldsymbol{\Theta}\boldsymbol{\phi}(\mathbf{x}), \tag{19}$$

mit $\boldsymbol{\Omega} = \begin{bmatrix} \boldsymbol{\omega}_1 & \boldsymbol{\omega}_2 & \dots & \boldsymbol{\omega}_N \end{bmatrix}$, $\boldsymbol{\omega}_i \in \mathbb{R}^m$ sowie $\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\theta}_1 & \boldsymbol{\theta}_2 & \dots & \boldsymbol{\theta}_N \end{bmatrix}$, $\boldsymbol{\theta}_i \in \mathbb{R}^m$. Die Parameter können mittels (16) sowie des TD-Learnings [3] in jedem Zeitschritt verändert werden über

$$\delta_t = r_t + \gamma V_{\boldsymbol{\nu}}(\mathbf{x}_{t+1}) - Q(\mathbf{x}_t, \mathbf{u}_t) \tag{20}$$

$$\boldsymbol{\theta}_{i,t+1} = \boldsymbol{\theta}_{i,t} + \alpha_\theta \phi_i(\mathbf{x}) \boldsymbol{\Omega}_t \boldsymbol{\phi}(\mathbf{x}) \tag{21}$$

$$\boldsymbol{\omega}_{i,t+1} = \boldsymbol{\omega}_{i,t} + \alpha_\omega \delta_t \phi_i(\mathbf{x}_t) \left(\mathbf{u}_t - \pi_{\boldsymbol{\Theta}}(\mathbf{x}_t)\right) \tag{22}$$

$$\nu_{i,t+1} = \nu_{i,t} + \alpha_\nu \delta_t \phi_i(\mathbf{x}_t). \tag{23}$$

Die freien Parameter $\alpha_\theta, \alpha_\omega$ und α_ν bezeichnen hierbei die Lernraten, bzw. wirken auf die Schrittweite für den Gradientenabstieg. Eine schlechte Wahl kann einerseits zur Divergenz, andererseits zu langsamen Lernen führen, weshalb ein Tuning der Parameter meistens erforderlich ist. Eine Methode diese Parameter a priori zu berechnen oder abzuschätzen ist dem Autor nicht bekannt.

In dieser Arbeit wurden als Funktionsapproximatoren Takagi-Sugeno-Systeme [5] der Form

$$\mathbf{f}(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^{N_{sys}} e^{\left((\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)\right)} \left(\mathbf{A}\mathbf{x} + \mathbf{d}\right) \tag{24}$$

$$k = \sum_{i=1}^{N_{sys}} e^{\left((\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)\right)} \tag{25}$$

verwendet. Die Spalten der freien Parameter $\mathbf{A}$ und $\mathbf{d}$ entsprechen dabei den $\boldsymbol{\omega}_i$, $\boldsymbol{\theta}_i$ oder ν_i aus (17), (18) und (19). Die $\phi_i(\mathbf{x})$ entsprechen entweder $\frac{x_j}{k} e^{\left((\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)\right)}$, $j \in \{1, \dots, n\}$ bzw. $\frac{1}{k} e^{\left((\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)\right)}$, je nachdem ob die Funktion auf Spalten von $\mathbf{A}$ oder $\mathbf{d}$ wirkt.

Bild 1: Aufbau einer schweren Kette mit Wagen, angetrieben von einem Elektromotor

4 Problembeschreibung und Modellierung

Das Ziel der folgenden Abschnitte ist es, das Verständniss der Probleme und eine Einschätzunng des benötigten Aufwands des Lernverfahrens an einem realen System ohne volle Zustandsinformation zu erlangen, sowie einen Vergleich zu einem analytisch ausgelegten Regler bezüglich der Regelgüte durchzuführen. Als Testsystem wird eine schwere Kette betrachtet, welche an einem Wagen angebracht ist und ohne zusätzliche Masse am unteren Ende von einem Punkt zum nächsten gefahren werden soll, ohne dabei zu stark nachzuschwingen. Ein Bild des Versuchsaufbaus ist in Bild 1 zu sehen. Die Kette hat 62 Glieder, eine Länge von $L = 1.178\text{m}$ und eine Masse von $M_W = 2.71\text{kg}$. Der Wagen wird von einem Elektromotor mit einer Spannung bis zu 45V angetrieben und hat eine Masse von $M_K = 1.39\text{kg}$.

Bei diesem Versuch kann nicht der gesamte Zustand der Kette, welcher aus Position und Geschwindigkeit aller Kettenglieder besteht, zurückgeführt werden, sondern lediglich die Position x_0 und Geschwindigkeit v_0 des Wagens sowie Winkel φ_0 und Winkelgeschwindigkeit $\dot{\varphi}_0$ des obersten Kettengliedes. Die Strecke erfüllt damit nicht die Markov-Eigenschaft, trotzdem sind die Algorithmen in der Praxis anwendbar. Aus den zurückgführten Zuständen werden anschließend die für den analytischen Regler und Lernalgorithmus benötigten Zustände

$$
\begin{aligned}
x_{0,e} &= x_0 - x_{0,soll} \\
v_{0,e} &= v_0 - v_{0,soll} \\
\phi_{0,e} &= \phi_0 - \phi_{0,soll} \\
\dot{\phi}_{0,e} &= \dot{\phi}_0 - \dot{\phi}_{0,soll}
\end{aligned}
$$

berechnet.

Als Vergleich wird ein analytischer Regler aus [7] herangezogen. Dieser basiert auf einer Modellierung als Kontinuum in Form einer partiellen Differentialgleichung. Nach dem Aufstellen der kinetischen und potentiellen Fehlerenergie kann über Backstepping der folgende Regler hergeleitet werden:

$$
\begin{aligned}
r &= -a(\rho g L \phi_{e,0} + c x_{e,0}) \\
\dot{r} &= -a(\rho g L \dot{\phi}_{e,0} + c v_{e,0}) \\
a_{e,\text{soll}} &= \dot{r} + (\frac{1}{a} + b) r - b v_{e,0} \\
F &= M a_{e,\text{soll}} - \rho g L \phi_{e,0}, \qquad (26)
\end{aligned}
$$

mit $\rho = 2.3\frac{kg}{m} = \frac{M_K}{L}$. Die drei freien Parameter a, b und c müssen dabei per Hand eingestellt werden. Der Regler stellt eine Abnahme der Fehlerenergie am Modell sicher, sodass die Kette aktiv an der gewünschten Position zur Ruhe geregelt wird.

Um die Regler zu vergleichen werden drei Gütemaße eingeführt. Die Gütemaße lauten

$$
r_{t,1} = \begin{cases} -dt(10x_{e,0}^2 + v_{e,0}^2 + 20\phi_{e,0}^2 + \dot{\phi}_{e,0}^2) & \text{, für } \operatorname{sgn}(x_{e,0}) = \operatorname{sgn}(v_{e,0}) \\ -0.1dt(10x_{e,0}^2 + 0.1v_{e,0}^2 + 0.1\phi_{e,0}^2 + 0.1\dot{\phi}_{e,0}^2) & \text{, sonst} \end{cases} \qquad (27)
$$

$$
r_{t,2} = \begin{cases} dt & \text{für } |x_{e,0}| < 0.01 \wedge |v_{e,0}| < 0.01 \wedge |\phi_{e,0}| < \frac{\pi}{60} \wedge |\dot{\phi}_{e,0}| < \frac{\pi}{60} \\ 0 & \text{sonst} \end{cases} \qquad (28)
$$

$$
r_{t,3} = \begin{cases} dt & \text{für } |x_{e,L}| < 0.01 \\ 0 & \text{sonst.} \end{cases} \qquad (29)
$$

Das erste Gütemaß wurde für das Training mittels des Actor-Critic-Algorithmus aus Abschnitt 3 verwendet und belohnt das Fahren in Richtung der Abnahme der Fehlerposition durch eine kleinere Gewichtung der quadrierten Fehlerterme. Dadurch wird im Gegensatz zu einer einfa-

chen quadratischen Gütefunktion ein Fahren weit über die Sollposition verhindert.

Das zweite Gütemaß ergibt in der Summe die Zeit in der sich die Kette zumindest nach messbaren Zuständen nah an der Ruhelage befand.

Das dritte Gütemaß betrachtet die Zeit, in welcher sich das untere Kettenende im Bereich um 1cm um die Ruhelage befand. Da die Kettenauslenkung am unteren Ende x_L nicht messbar ist, kann dieses Gütemaß nur in der Simulation berechnet werden.

Als Trajektorie für das Training in der Simulation des Actor-Critic-Reglers wurde die Trajektorie

$$x_{0,soll} = x_{L,soll} = \begin{cases} 0.0 & \text{für } 0 < t \leq 3 \vee 12 < t \leq 20 \\ 0.5 & \text{für } 3 < t \leq 12 \end{cases} \tag{30}$$

wiederholt abgefahren. Für die Berechnung der Gütemaße des gelernten Reglers sowie für den analytischen Regler wurde folgende Wunschtrajektorie aufgestellt

$$x_{0,soll} = x_{L,soll} = \begin{cases} 0 & \text{für } t \leq 2 \vee 25 < t \\ 0.5 & \text{für } t \leq 10 \\ 0.2 & \text{für } t \leq 17 \\ 0.4 & \text{für } t \leq 25 \end{cases} \tag{31}$$

und für $t = 0$ bis $t = 30$ simuliert.

In dem realen Aufbau wurde das gleiche Vorgehen gewählt, allerdings wurde $x_{0,soll}$ aufgrund des begrenzten Fahrwegs des Wagens skaliert auf $x_{0,soll,\text{Aufbau}} = 0.2 + 0.8 x_{0,soll}$.

Zusätzlich zum realen System wird der Algorithmus an einem Simulationsmodell getestet. Dieses dient in diesem Beitrag aber nicht als Entwurfsmodell und der Regler wird nicht vom Modell auf das reale System übertragen, sondern kann als zweites unabhängiges System betrachtet werden. Für die Simulation der Kette wurde ein Mehrkörpermodell einer Kette der gleichen Länge, jedoch mit 20 Kettengliedern erstellt, welches im Vergleich zu einem Mehrkörpermodell mit 62 Kettengliedern kaum Unterschiede in der Dynamik aufweist, jedoch eine um Faktor 10 geringere Simulationszeit hatte. Um alle Trägheiten im System, z.B. des Motorankers und der Antriebsriemen ebenfalls zu berücksichtigen, wurde der Wagen mit einer erhöhten Masse von 6.15kg modelliert. Weitere

Unterschiede zum realen Modell sind zum einen der Eingang, welcher im realen System die Motorspannung und im Simulationsmodell die Kraft auf den Wagen ist. Auch die Haftreibung des Wagens und des Motors sind im Simulationsmodell vernachlässigt. Die Massenmatrix und der Kraftvektor in den generalisierten Koordinaten wurden in der freien Software SymPy-Mechanics [8] hergeleitet.

5 Ergebnisse in der Simulation

Für die Simulation des Mehrkörpermodell wurde eine Zeitschrittweite von $dt_{\text{sim}} = 0.001\text{s}$ gewählt, da die Stabilität, ähnlich wie beim numerischen lösen partieller Differentialgleichungen, nur für ausreichen kleine Schrittweiten sichergestellt ist. Die Regler und der Lernalgorithmus jedoch wurden nur alle $dt_{\text{rl}} = 0.04\text{s}$ ausgeführt.

Die TS-Systeme wurden als äquidistantes Gitter mit $7 \times 5 \times 11 \times 5$ Gitterpunkten ausgelegt, mit $x_{e,0} \in [-1, 1]$, $v_{e,0} \in [-1, 1]$, $\phi_{e_0} \in [\frac{-\pi}{24}, \frac{-\pi}{24}]$ und $\dot{\phi}_{e_0} \in [-1.2, 1.2]$. Für γ wurde 0.99 gewählt. Nach manuellem Einstellen der Lernraten ergab sich $\alpha_\theta = 0.015$, $\alpha_w = 0.01$ und $\alpha_v = 0.08$. Leider konnte kein methodisches Vorgehen oder sinnvolle Reihenfolge zum Einstellen der Parameter gefunden werden. Als Belohnung r_t für den Lernalgorithmus wurde (27) benutzt, wodurch das Fahren des Wagens über die Sollposition im Gegensatz zu einfachen Quadratischen Belohnungsfunktionen deutlich verringert werden konnte. Insgesamt spielt die Belohnungsfunktion eine entscheidende Rolle sowohl für die Lerngeschwindigkeit als auch für den am Ende erhaltenen Regler [10].

Für den analytischen Regler wurden die Parameter per Hand auf $a = 0.03$, $b = 10$, $c = 10$ eingestellt.

Der Verlauf der Position des Wagens sowie des Kettenendes x_L sind in Bild 2 für den erlernten Regler und in Bild 3 für den analytischen Regler dargestellt. Die erreichten Gütemaße sind in Tabelle 1 aufgelistet.

Es zeigt sich, dass der gelernte Regler die Trajektorie aus dem Training mit Sprüngen der Amplitude 0.5m fast ohne Überschwinger sowohl des Wagens als auch des Kettenendes und im Vergleich zum analytischen Regler kleinerem Übertreten der Sollposition auskommt. Für Sprünge der Sollposition anderer Amplitude als im Training verbleiben jedoch Schwingungen des Kettenendes. Der analytische Regler dagegen zeigt hier deutlich kleinere Restschwingungen und ist bei passender Wahl der

drei freien Parametern in allen Gütemaßen besser als der gelernte Regler. Für eine gute Regelgüte sollte die Trainingstrajktorie demnach möglichst nah an der späteren Wunschtrajektorie liegen.

Tabelle 1: Gütemaße für den analytischen und gelernten Regler in der Simulation

	gelernter Regler	analyt. Regler
$r_{t,1}$	-0.757	-0.439
$r_{t,2}$	10.68	19.68
$r_{t,3}$	9.08	23.52

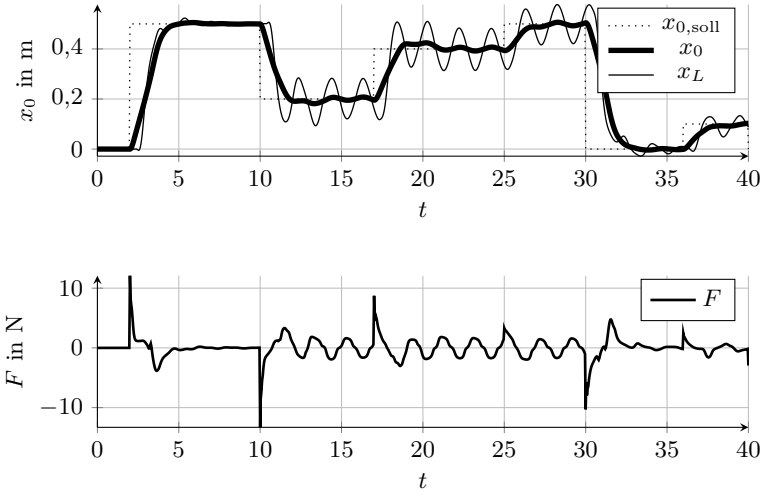


Bild 2: Testtrajektorie mit der aufgebrachten Kraft F für den gelernten Regler

6 Anwendung am realen System

Die Anwendung eines Reinforcement-Learning-Algorithmus an echten Systemen bringt einige Herausforderungen mit sich [10]. Da das Lernen am echten System in Echtzeit stattfindet, sind Parameterjustierungen aufwändig und zeitintensiv. Die Lernraten und auch Gütefunktion sollten möglichst günstig gewählt werden, was nur in Kombination mit Systemwissen möglich ist.

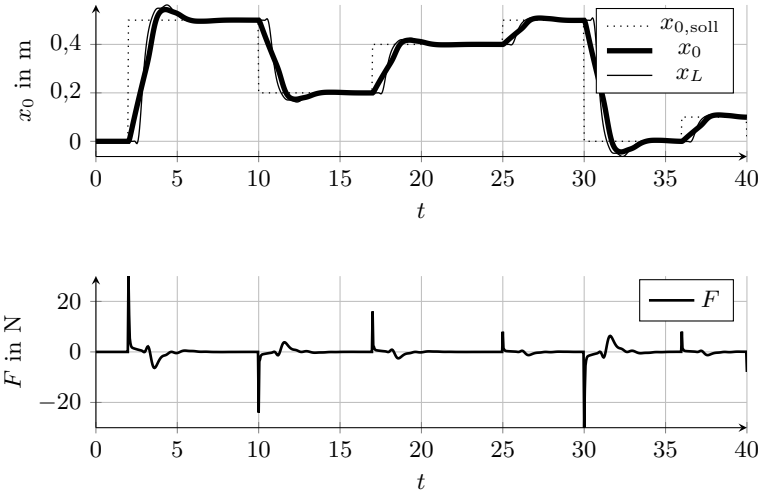


Bild 3: Testtrajektorie mit der aufgebrachten Kraft F für den analytischen Regler

Die Sensoren wurden mit einer Abtastzeit von $dt_{sensor} = 0.001$s ausgelesen während der Lernalgorithmus sowie auch der analytisch Regler auf eine Abtastzeit von $dt_{rl} = 0.1$s agierten. Nach drei Parametereinstellungen bei denen der Aktor ein zu hektisches Verhalten erreichte und der Versuch abgebrochen werden musste um die Hardware zu schonen, wurden die Lernraten aus der Simulation ($\alpha_\theta = 0.015$, $\alpha_w = 0.01$, $\alpha_v = 0.08$) übernommen. Insgesamt lernte der Regler sechs Stunden an der Strecke, nach weiteren drei Stunden Training zeigte sich eine Verschlechterung des Reglers da er um die Sollposition herum eine Dauerschwingung erzeugte.

Die Gütemaße des gelernten Reglers sowie des analytischen Reglers sind in Tabelle 2 aufgelistet. Die Fehlerposition und der Winkel der abgefahrenen Trajektorien sind in Bild 4 zu sehen, aufgrund der Speichergröße des Echtzeitsystems jedoch auf die letzten 12.5s beschränkt.

Der gelernte Regler schneidet in beiden Gütemaßen schlechter ab als der analytischen Regler. Vor allem werden vorhandenen Restschwingungen vom analytischen Regler schneller ausgeregelt.

Tabelle 2: Gütemaße für den analytischen und gelernten Regler am realen Aufbau

	gelernter Regler	analyt. Regler
$r_{t,1}$	-2.202	-1.701
$r_{t,2}$	0.3	5.1

Die größte Schwierigkeit für die Anwendung liegt in der hohen Anzahl an einzustellenden Freiheitsgraden, z.B. die Lernraten, Anzahl an Parametern, Startwerte und Zeitschrittweite. So kann es sein, dass sich ein deutlich besserer Regler mit anderen Parametereinstellungen lernen ließe, was jedoch aufgrund des hohen Zeitaufwandes füer das Lernen nur mit großem Aufwand realisierbar ist. Dies ist in der Simulation deutlich schneller machbar, da in dem Fall mehrere Parametereinstellungen parallel getestet werden können. Als Zusammenfassung lässt sich sagen, dass der Lernalgorithmus zwar sehr allgemein einsetzbar ist, das Tuning der Parameter jedoch eine in der Praxis relevante Einschränkung darstellt, sowie die Regelgüte in diesem Fall schlechter ausfiel als diejenige des analytischen Regler.

7 Zusammenfassung und Ausblick

In diesem Beitrag wurden die Grundlagen des Reinforcement-Learning erläutert und anschließend ein Actor-Critic-Algorithmus basierend auf [4] beschrieben. Der Algorithmus wurde sowohl in der Simulation als auch am realen System einer hängenden Kette angewendet.

Es zeigte sich, dass vor allem das Einstellen der vielen Freiheitsgrade des Lernalgorithmus problematisch sind. Auch nach mehreren Iterationen blieb die Regelgüte des gelernten Reglers deutlich hinter der eines zum Vergleich herangezogenen, modellbasierten Reglers zurück. Bessere Parametereinstellungen und längeres Lernen könnten die Regelgüte zwar verbessern, der dazu benötigte Zeitaufwand ist jedoch sehr hoch. Als Vorteil ist jedoch die vielseitige Anwendbarkeit dieser Art von Lern-Algorithmus anzumerken, sowie der Verzicht auf eine mathematische Modellierung der Strecke.

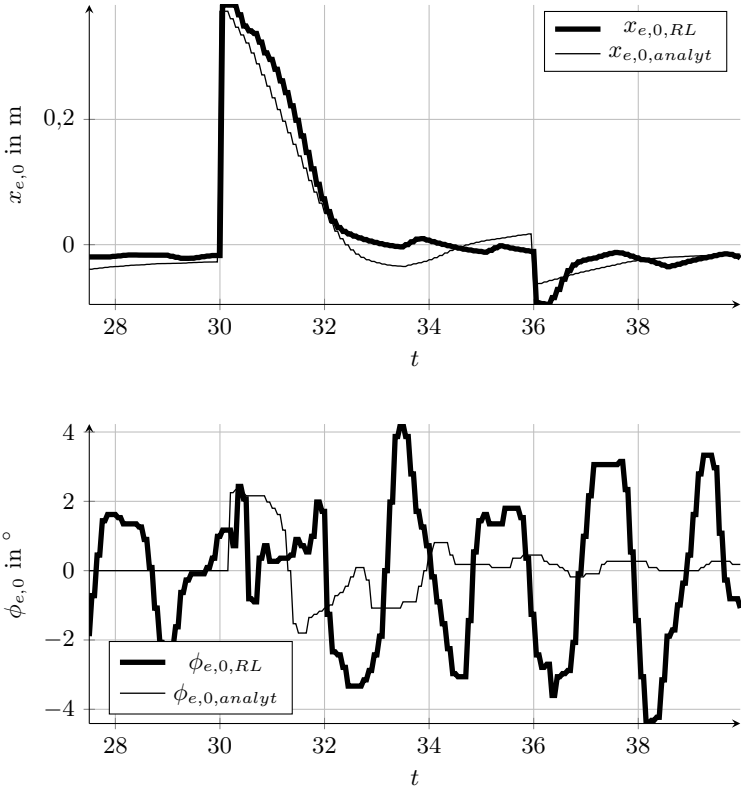


Bild 4: Fehlerposition und Winkel im realen Versuch

Als Weiterführung der Arbeit werden weitere Algorithmen und Ideen, z.B. adaptive Lernraten[11], an diesem und anderen Systemen implementiert mit dem Ziel einem robusten, für die industrielle Praxis sowie für Nicht-Experten auf dem Gebiet einsetzbaren Algorithmus näher zu kommen.

Danksagung: Der Autor dankt der Deutschen Forschungsgemeinschaft (DFG) für die Förderung dieser Forschung im Rahmen des Sonderforschungsbereichs 768 (Zyklenmanagement von Innovationsprozessen – verzahnte Entwicklung von Leistungsbündeln auf Basis technischer Produkte).

Literatur

[1] Mnih, V. et al „Human-level Control through deep reinforcement learning“ Nature 518, 2015.

[2] Silver, D. et al. „Mastering the game of Go with deep neural networks and tree search“ Nature 529, 2016.

[3] Sutton, R. S. und Barto, A. G. „Reinforcement Learning: An Introduction“. 1998.

[4] Silver, D. et al. „Deterministic Policy Gradient Algorithms“. ICML, 2014.

[5] Takagi, T. und Sugeno, M. „Fuzzy identification of systems and its applications to modeling and control“. IEEE Trans Syst Man Cybern Syst, 1985.

[6] Baird, Leemon C. „Advantage Updating“. Wright Lab, 1993.

[7] Thull, D. und Wild, D. und Kugi, A. „Infinit-dimensionale Regelung eines Brückenkranes mit schweren Ketten“. at - Automatisierungstechnik 53, 2005.

[8] Meurer, A. et al. „SymPy: symbolic computing in Python“. PeerJ Computer Science 3:e103, 2017.

[9] Grondman, I. et al. „A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients“ IEEE Trans Syst Man Cybern Syst, 2012.

[10] Kober, J. und Bagnell J. A. und Peters J. „Reinforcement Learning in Robotics: A Survey“ International Journal of Robotics Research, 2013.

[11] Dabney, W. C. „Adaptive step-sizes for reinforcement learning“. Phd.-Thesis, University of Massachusetts, 2014.

Discrete Wavelet Transforms and Multivariate Gaussian Distributions for Anomaly Detection in Time Series

Markus Thill[1], Wolfgang Konen[1], Thomas Bäck[2]

[1]TH Köln – University of Applied Sciences, Dept. of Computer Science
Steinmüllerallee 1, 51643 Gummersbach, Germany
E-Mail: {markus.thill, wolfgang.konen}@th-koeln.de

[2]Leiden University, Leiden Institute of Advanced Computer Science
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
E-Mail: t.h.w.baeck@liacs.leidenuniv.nl

1 Introduction

Wavelet transforms [7] are commonly used to decompose a time series signal into accurate time-localized frequency information. In practice, typically the discrete wavelet transform (DWT) is performed, which can be efficiently implemented with time-discretized filters.
In the following, we describe an unsupervised learning algorithm (abbreviated DWT-MLEAD) for anomaly detection in time series, which is based on discrete wavelet transforms and maximum likelihood estimation. For each time series the DWT-MLEAD algorithm (i) computes a decimating DWT using Haar wavelets, (ii) estimates the parameters of a multivariate Gaussian distribution for the individual frequency scales of the DWT, (iii) flags unusual data points in each frequency scale based on a (Mahalanobis-distance-based) quantile estimate, and finally (iv) aggregates the unusual data points of all frequency scales and detects anomalies in the original time series. The motivation for this work is to advise an algorithm which works well on time series with anomalies at different frequency levels.

1.1 Related Work

Although wavelet transforms are widely used in many fields of signal and image processing and in many data mining tasks such as time series classification, clustering, prediction & forecasting and similarity search [1],

their potential for time series anomaly detection is only partially exploited: Kwon et al. [4] use wavelet packet transforms in order to detect anomalies in network traffic which might indicate an malicous attack. Kanarchos et al. [3] use wavelets in conjunction with neural networks and Hilbert transforms.

2 DWT-MLEAD: An Algorithmic Description

In this section we briefly describe the main components of the DWT-MLEAD algorithm. A detailed discussion will be presented in a forthcoming paper [9]. The underlying idea for the design of the algorithm is to analyze a time series signal at different frequency scales. We expect that anomalies are present on different frequency scales and that some anomalies might even be more apparent at certain scales than in the original time series. For achieving this, DWT-MLEAD performs a decimating discrete wavelet transform (DWT) using simple Haar wavelets for a given time series. The DWT returns the so called detail and approximation coefficients, which are arranged in L levels (or scales) and basically contain information about the original time series at different frequency resolution levels. The number of levels L depends on the length of the original time series. The highest level only includes detail coefficients which represent the original time series and the lowest level consists of only one coefficient. Due to pyramidal shape of the DWT-decomposition, one can arrange both sets of coefficients in two separate binary DWT trees. This is illustrated in Fig. 1. Both, detail and approximation coefficients will be used for further processing, however, lower levels are typically omitted, since they mostly do not contain useful patterns.
Then, a window of a specified length (typically between 2–10, depending on the level) is slid over each detail and approximation level separately. Each window content observed while sliding the window over a certain level represents a data point and is added as a row to a matrix. Subsequently, for each generated matrix, DWT-MLEAD estimates the parameters of a Gaussian distribution using maximum likelihood estimation (MLE). In order to locate "unusual" instances in each scale, the algorithm requires a method to separate the unusual data points from the others. For this purpose, DWT-MLEAD computes the log-likelihood of each observed data point for the estimated distribution and flags those instances, for which the probability lies beneath a given value. This value can be determined with either of two approaches: In the first approach,

an empirical quantile (e.g. the 2nd percentile of all log-probabilities) is computed. The second approach is based on the observation that a Gaussian random variable has a squared Mahalanobis distance from its mean, which is Chi-squared distributed. As shown in Fig. 1, instances previously flagged as unusual cause an event which is passed down the DWT tree to all connected leaf nodes representing the original time series. In the leaf nodes, all incoming events are counted in a leaf counter. Finally, if a counter exceeds a specified threshold, the corresponding point in the time series will be flagged as anomalous.
A detailed algorithmic description can be found in [9].

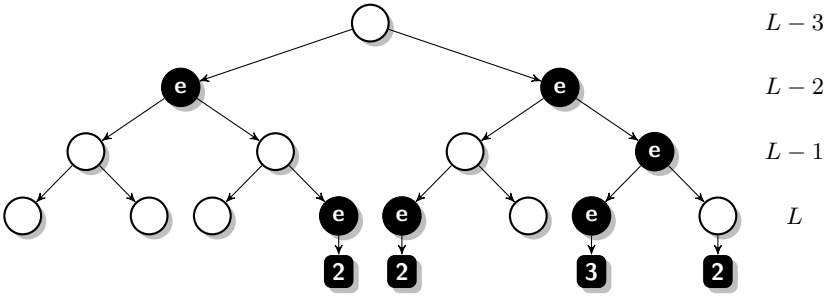


Figure 1: Illustration of the DWT tree for anomaly detection. Both, detail and approximation coefficients are arranged in a separate binary tree. Along the vertical axis are the DWT levels, along the horizontal axis are the time indices. Each event increases the leaf counters (rectangles) connected with the e node. Only counters with count ≥ 2 are shown.

3 Results

In order to assess the performance of the proposed DWT-MLEAD algorithm, we construct a benchmark consisting of the well known Numenta Anomaly Benchmark (NAB, 58 mostly real-world time series) [6] and of a subset of the Yahoo's S5 Webscope benchmark [5] (A3, 100 synthetic time series) and we compare our algorithm to several state-of-the-art algorithms: Numenta's NuPic [2], our previous algorithm SORAD [8], and Twitter's ADVec algorithm [10]. The parameters of each algorithm for both datasets were determined empirically. The results are listed in Tab. 5. Correctly identified anomalies within a time series are considered as true-positives (TP) and misclassifications as false-positives (FP) and

false-negatives (FN). According to the derived F_1-score, our algorithm is placed first on both benchmarks and clearly outperforms NuPic & ADVec on the A3 data and SORAD & ADVec on the NAB data.

Table 1: Results for various algorithms on the A3 and NAB dataset. Shown are the sums of TP, FP, FN over all time series and the metrics precision, recall and F_1, derived from these sums. All algorithms have their parameters chosen such that F_1 is maximized.

Dataset	Algorithm	TP	FP	FN	Precision	Recall	F_1 Score
A3	DWT-MLEAD	806	8	44	**0.99**	**0.95**	**0.97**
	NuPic	172	267	678	0.39	0.2	0.27
	SORAD	810	22	40	0.97	**0.95**	0.96
	ADVec	190	216	660	0.47	0.22	0.3
NAB	DWT-MLEAD	69	65	46	**0.51**	0.6	**0.55**
	NuPic	76	113	39	0.4	**0.66**	0.5
	SORAD	57	313	58	0.15	0.5	0.24
	ADVec	66	164	49	0.29	0.57	0.38

4 Conclusion & Future Work

We found that the DWT is an useful instrument to extract meaningful patterns on different frequency scales from a time series and to detect temporal anomalies based on these patterns. Our proposed DWT-MLEAD algorithm was tested on a diverse benchmark consisting of 158 time series and compared to several state-of-the-art algorithms, achieving promising results. In the next steps we are planning to improve and extend this initial version of our algorithm, which for example could include: (a) investigating wavelets other than Haar wavelets, (b) models other than Gaussian distributions and (c) developing an (semi-) online version of the algorithm.

References

[1] Chaovalit, P., Gangopadhyay, A., Karabatis, G., Chen, Z.: Discrete wavelet transform-based time series analysis and mining. ACM Comput. Surv. 43(2), 6:1–6:37 (Feb 2011)

[2] George, D., Hawkins, J.: Towards a mathematical theory of cortical micro-circuits. PLoS Computational Biology 5(10) (2009)

[3] Kanarachos, S., Mathew, J., Chroneos, A., Fitzpatrick, M.: Anomaly detection in time series data using a combination of wavelets, neural networks and Hilbert transform. In: International Conf. on Information, Intelligence, Systems and Applications (IISA). pp. 1–6 (2015)

[4] Kwon, D., Ko, K., Vannucci, M., Reddy, A.N., Kim, S.: Wavelet methods for the detection of anomalies and their application to network traffic analysis. Quality and Reliability Engineering International 22(8), 953–969 (2006)

[5] Laptev, N., Amizadeh, S.: Yahoo anomaly detection dataset S5 (2015), `http://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70`, (accessed: Sep. 2017)

[6] Lavin, A., S. Ahmad, S.: Evaluating real-time anomaly detection algorithms – the Numenta anomaly benchmark. In: IEEE Conference on Machine Learning and Applications (ICMLA2015) (2015)

[7] Meyer, Y., Salinger, D.: Wavelets and Operators, Cambridge Studies in Advanced Mathematics, vol. 1. Cambridge University Press (1995)

[8] Thill, M., Konen, W., Bäck, T.: Online anomaly detection on the Webscope S5 dataset: A comparative study. In: IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS 2017). p. 1. Springer (2017)

[9] Thill, M., Konen, W., Bäck, T.: Time series anomaly detection with discrete wavelet transforms and maximum likelihood estimation. In: International Work-Conference on Time Series (ITISE2017) (accepted Jul 2017)

[10] Vallis, O., Hochenbaum, J., Kejariwal, A.: A novel technique for long-term anomaly detection in the cloud. In: 6th USENIX Workshop on Hot Topics in Cloud Computing, Philadelphia, PA (2014)

Algorithms for the Identification of Merged Local Model Networks

Tobias Münker, Oliver Nelles

Mess- und Regelungstechnik – Mechatronik, Universität Siegen
Paul-Bonatz-Straße 9-11, D-57068 Siegen
E-Mail: tobias.muenker@uni-siegen.de

Abstract

This contribution deals with the identification of nonlinear input/output relationships from data. Local model networks which use validity functions to blend linear relationship between different regions of the input space, see [12] for an overview, have proven to be a powerful tool for nonlinear regression. Though this structure offers the advantage that once the validity functions are determined the identification simplifies to a linear weighted least squares problem which can be solved efficiently, the identification of the validity functions is much more involved.

This paper introduces a novel methodology for the identification of these validity functions and compares different algorithms to heuristically solve the resulting optimization problem. The key idea is to assign a Gaussian membership function to each data point. These functions are normalized to obtain the validity functions. For the identification of the local models an optimization problem which allows only a specific number of different local model parameter values is formulated. The process of restricting many local model parameters to be equal can be seen as a merging of several local models. The resulting optimization problem is non-convex since the cardinality of the different possible parameter values is constrained. Two heuristics to solve this optimization problem are discussed. The first one uses a norm penalty of the differences of the parameters similar to group lasso. The other one uses the k-means clustering algorithm to restrict the estimated parameters to be equal to the found cluster centers.

Compared to state-of-the art methods for identification of validity functions the methods offer two advantages. The first one is the flexibility since the obtained validity functions are not constructed through consecutive

splits as it is usually the case for tree-based construction algorithms, e.g. [14] or [19]. The second one is that compared to common approaches based on clustering, e.g. [1], which only consider the distribution of points in the combined input and output space, our approach favors to merge regions that can be described best with a linear relationship. The method is applied for two static regression examples.

1 Introduction

Identification of local model networks from measured data is a technique that has been successfully used in many applications, see [12] for an overview. In recent times for the identification of input/output relationships of nonlinear processes approaches have been introduced that offer a greater flexibility for the identification. These approaches include Gaussian process models [16] and deep neural networks, see [8]. The approach recently introduced in [11] and discussed in this contribution follows the same route and provides a possibility to increase the flexibility of the identified validity functions for local model networks.

1.1 Local Model Networks

The basic idea of local model networks with linear local models, which are also referred to as Takagi-Sugeno Fuzzy networks [17], is that in specific regions of the input space the input/output relationship can be described by a linear function. These linear relationships are then blended with validity functions. Beside the fact that having a linear relationship in a specific region of the input space is a useful assumption on its own this enables the usage of weighted linear regression techniques for the inference of these linear relationships. The shortcoming of this approach however is that it is necessary to determine the validity functions. Therefore several possibilities have been proposed. The approaches include clustering in the input space [19] or in the product space with k-means [2] or Gath-Geva clustering [1], axis orthogonal sequential splitting algorithms [14] and algorithms that subsequently split the input space with oblique splits [19]. The approaches based on clustering have the disadvantage that the input/output relationship is not considered for the determination of the validity regions, whereby the sequential splitting of the input space

circumvents this issue but assumes that for each subsequent split the resulting local models are separated by a straight line.

1.2 Piecewise Affine Models

Another interesting and related class of models are piecewise linear or piecewise affine models. Instead of blending the linear relationship the models are switched depending on the input. They are especially relevant since many hybrid dynamic systems can be represented by this model class, see [3] for an overview of equivalent descriptions. A popular and successful algorithm for the identification of this model class is the tree based CART [6] and the hinging hyperplane algorithm [5]. In [7] an approach based on clustering has been provided. This approach starts by a local regression based on a number of points lying nearby and then clusters the parameters in that space. This procedure though has some drawbacks. First the number of points selected as nearby points is somewhat arbitrary and remains to be tuned. Furthermore, it is difficult for new points to assign them to a cluster accordingly. This problem is circumvented by the validity functions of local model networks which are able to provide the validity for any point in the input space.

1.3 Structure of the Contribution

The contribution is structured as follows. First the problem of nonlinear static system identification will be introduced and the idea of merged local model networks will be presented. This leads to the non-convex optimization problem that has to be solved for the identification of merged local model networks. Afterwards an algorithm that uses a 1-norm convexification of the 0-pseudo-norm as a heuristic solution of the optimization problem is discussed. It will be shown that this algorithm fails to provide satisfying solutions for the optimization problem. Therefore an alternative heuristic for the solution of the problem, which has been introduced in [11] is presented. This approach relies on a two-step procedure. In the first step the parameters of models described by the local validity functions are estimated and then within the obtained parameters clusters are created using the k-means algorithm to ensure that the constraints are met. In the end illustrative examples are provided and a conclusion is given.

2 Problem Statement

We consider that data $\mathcal{D} = \{\mathbf{u}(i), y(i)\}_{i=1}^{N}$ with $N \in \mathbb{N}$ denoting the number of measurements, the input and output $\mathbf{u}(i) \in \mathbb{R}^{n_u}$, $y(i) \in \mathbb{R}$ is given. Our goal is to identify a nonlinear relationship

$$\hat{y}(k) = f(\mathbf{u}) \tag{1}$$

such that the loss function

$$J = \sum_{i=1}^{N} \left(y(i) - f\left(\mathbf{u}(i)\right)\right)^2 \tag{2}$$

is minimized.

Local Model Networks In the presented approach we use a local model network for the function approximator $f(\mathbf{u})$

$$L(\mathbf{u}) = \sum_{i=1}^{m} L_i(\mathbf{u})\phi_i(\mathbf{u}) \tag{3}$$

with the m local models

$$L_i(\mathbf{u}) = \mathbf{w}_i^T \begin{bmatrix} 1 & \mathbf{u}^T \end{bmatrix}^T \tag{4}$$

and the validity functions ϕ_i, which posses the partition of unity property

$$\sum_{i}^{N} \phi_i(\mathbf{u}) = 1 \tag{5}$$

for all $\mathbf{u}$. Usually these functions ϕ_i are obtained via clustering or a successive splitting of the input space.

Merged Local Model Networks In this contribution we propose another approach which is similar to a kernel based approach. Therefore we consider membership functions

$$\eta_i(\mathbf{u}) = \exp\left(\frac{\|\mathbf{u} - \mathbf{u}(i)\|_2^2}{\sigma^2}\right) \tag{6}$$

with the bandwidth σ^2, which lie on every point of the training data. It is possible to derive a local validity function

$$\nu_i(\mathbf{u}) = \frac{\eta_i(\mathbf{u})}{\sum_{j=1}^{N} \eta_j(\mathbf{u})}. \tag{7}$$

These functions fulfill the partition of unity property due to the normalizing denominator. When the ν_i are added the respective local models are merged and the global validity functions are obtained. It is easy to see that combining the ν_i to obtain global validity functions ϕ_i does not change the partition of unity property. The goal of our approach is to find the final validity functions not by clustering or partitioning of some input or combined input/output space but by merging the validity functions to a local model. This idea is illustrated in Fig. 1 where 5 input data locations are considered equally spaced between 0.2 and 0.8. The upper subfigure shows the local membership and the second subfigure the local validity functions. The third figure shows the global validity functions when ϕ_1 is found merging ν_1 and ν_2 and ϕ_2 is found when ν_3, ν_4 and ν_5 are summed up.

Local and Global Identification According to other local model approaches it is possible to derive both a local and a global identification problem. The global identification problem considers the validity function inside the loss function and can be written as

$$\begin{aligned} \underset{\mathbf{l}_i, \mathbf{w}_i}{\text{minimize}} \quad & \sum_{i=1}^{N} \left(y(i) - \sum_{j=1}^{N} \nu_j(\mathbf{u}(i)) \left[1 \ \mathbf{u}^T(i)\right] \mathbf{l}_j \right)^2 . \\ \text{subject to} \quad & \mathbf{l}_k \in \{\mathbf{w}_1, \ldots, \mathbf{w}_m\} \quad k = 1, \ldots, N \end{aligned} \tag{8}$$

The constraint enforces that there are only m different values for the local parameters, which is the same as combining the local validity functions ν_i to the global validity functions ϕ_i. For the local optimization problem the loss function is minimized for each local model individually. Thus the identification problem becomes

$$\begin{aligned} \underset{\mathbf{l}_i, \mathbf{w}_i}{\text{minimize}} \quad & \sum_{i=1}^{N} \sum_{j=1}^{N} \nu_j(\mathbf{u}(i)) \left(y(i) - \sum_{j=1}^{N} \left[1 \ \mathbf{u}^T(i)\right] \mathbf{l}_j \right)^2 . \\ \text{subject to} \quad & \mathbf{l}_k \in \{\mathbf{w}_1, \ldots, \mathbf{w}_m\} \quad k = 1, \ldots, N \end{aligned} \tag{9}$$

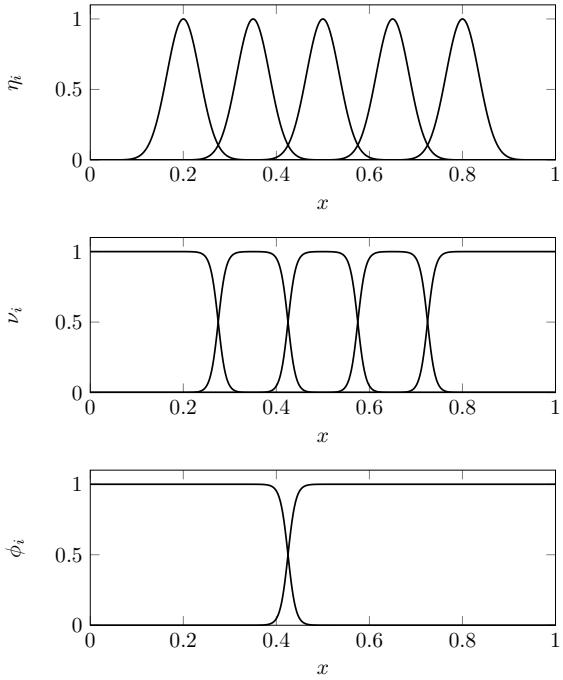


Figure 1: Illustration of the main idea for the merged local model network approach.

To put the weighting factor $\nu_j(\mathbf{u}(i))$ in front of the squared loss allows to solve N independent least-squares problems. This will decrease the model accuracy on training data in areas where more than one local model is valid. Nevertheless often the performance on test data is increased since the restricted model flexibility in the regions where the model is blended affects the variance error of the parameters positively. Thus in this contribution the local estimation problem (9) is considered. Both (8) and (9) are non-convex. How to find a suitable heuristic to deal with this non-convexity is subject of the following sections.

3 Regularization Approach Based on 1-Norm

One approach to solve the non-convex optimization problem described in the previous section is to use convex relaxations. In recent times

LASSO [18] which uses the 1-norm for convex relaxation of the regressor selection problem has proven to yield good solutions there. Thus we also try, based on the work of [15] for PWA systems, to derive a convex relaxation of the problem. The difficulty here is that it is not required that the N parameter vectors $\mathbf{l}_j$ are sparse but the differences between them should be sparse. Furthermore, the vectors $\mathbf{l}_i$ have $n+1$ entries and the differences between these entries should be sparse in groups. The last problem can be solved similar to the group lasso approach [20]. Therefore, instead of the 1-norm of the individual parameters the 2 norm of the parameters are used. To solve the first problem it is required that the differences of variables are penalized for regularization for N local models there are N^2 possible differences between the variables. These differences can be weighted with the validity functions to consider the location of the point in the formulation of the optimization problem. Considering this two ideas yields the optimization problem

$$\underset{\mathbf{l}_i,\mathbf{w}_i}{\text{minimize}} \quad \sum_{i=1}^{N}\sum_{j=1}^{N} \nu_j(\mathbf{u}(i)) \left(y(i) - \left[1 \ \mathbf{u}^T(i)\right] \mathbf{l}_j\right)^2 + \lambda \|\mathbf{D}\|_0 \tag{10}$$

where the pseudo-norm $\|\cdot\|_0$ stands for the number of nonzero entries of a matrix and $\mathbf{D}$ contains

$$d_{ij} = \phi_i(\mathbf{u}(j))\|\mathbf{l}_i - \mathbf{l}_j\|_2. \tag{11}$$

The matrix $\mathbf{D}$ contains the weighted differences of the local models. The parameter λ is used to determine the complexity of the model. If λ is chosen very high only one global linear model is obtained, if λ is chosen as 0 for each local validity function an independent model is obtained. For the identification of linear parameter varying systems a similar problem has to be solved. In [15] a similar approach is introduced which is referred to as sum-of-norms regularization. There the pseudo 0-norm is replaced with the convex 1-norm. Instead of the combinatorial problem a convex optimization problem is obtained. The relaxed 1-norm problem is written as

$$\underset{l_i,w_i}{\text{minimize}} \quad \sum_{i=1}^{N}\sum_{j=1}^{N} \nu_j(\mathbf{u}(i)) \left(y(i) - \left[1 \ \mathbf{u}^T(i)\right]^T \mathbf{l}_j\right)^2 + \lambda \|\mathbf{D}\|_1 \quad . \tag{12}$$

with $\|\mathbf{D}\|_1 = \sum_{i=1}^{N}\sum_{j=1}^{N} |D_{ij}|$ denoting the entry wise 1 matrix norm. Since each entry of $\mathbf{D}$ is a norm and thus positive $\|\mathbf{D}\|_1$ can be simply

evaluated by forming the sum of the entries. The obtained optimization problem is now convex since the objective is a sum of a quadratic loss function and a sum of norms. This holds due to the fact that the sum of convex functions yields a convex function, see [4]. These type of problems can be solved for moderate number of variables using interior-point methods. Since the objective function has many terms this approach is only feasible here for small N, say 100. The examples calculated in the next sections use CVX [9] to solve the resulting optimization problems.

3.1 One Dimensional Test Problem

To illustrate the difficulties involved using the group lasso penalty for the identification of a one dimensional relationship a simple example is presented. Therefore a simple parabola $f(u) = 4u^2 - 4u + 1$ is used as the data generating function and the input is linearly spaced between -1 and 1 with 10 samples. These are illustrated in Fig. 2. To discuss the behavior of the algorithm no noise is used. The result that is obtained depends on the regularization parameter λ. Thus λ has been varied between 0 which corresponds to the point individual local regression problem and 10 which corresponds to the least-squares solution of a global local model. The lower parts of the figure illustrate the behavior of the local parameters of $\mathbf{l}_i$. The first entry is the offset and is shown in the middle plot and the last entry is the slope which is shown in the lower plot.

3.2 Problems of the Heuristic

One problem, that is also present in the approach presented by [15], is that the penalization introduces a bias towards zero. Often sparsity is achieved in an area where many models lie nearby. The matrix $\mathbf{D}$ becomes sparse but this does not result in a satisfying solution of the regression problem. In Fig. 2 this can be clearly observed for $\lambda = 1$. Though the regularization causes the parameters for $u_1 \in [0, 0.5]$ to be equal, for $u_1 \in [0.5, 1]$ the parameters are biased towards 0 but are still allowed to differ from one another. Thus it is not possible to find appropriate global validities from the solution of the sum of norms problem.

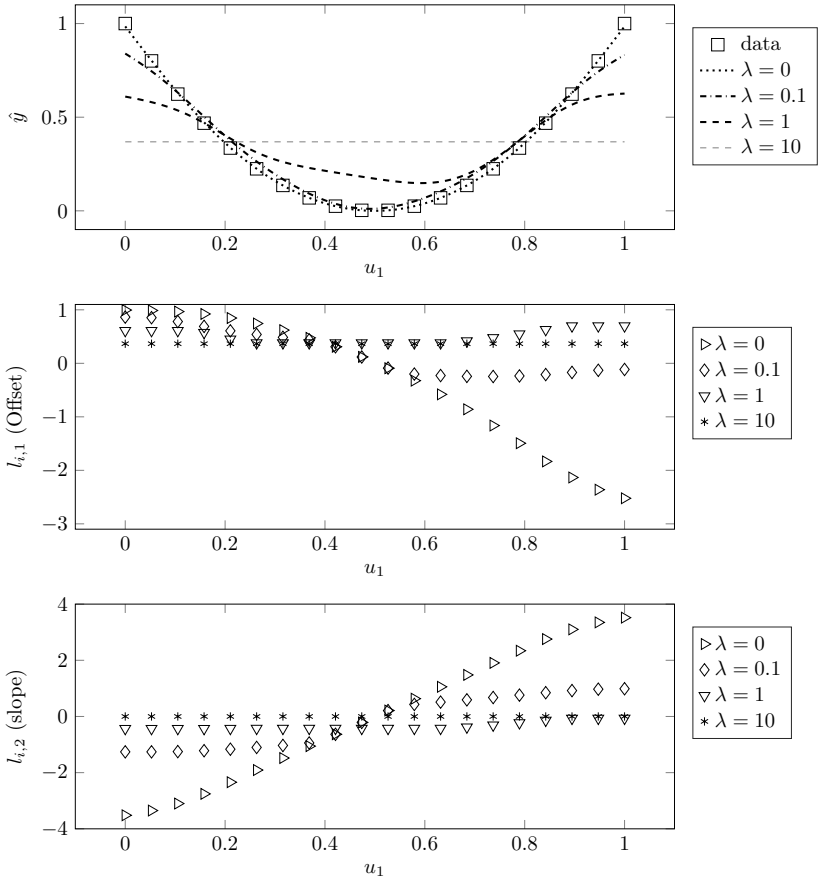


Figure 2: Solution of the 1 heuristic for a parabola. Regression result for different choices of λ (upper axis), offset and slope of the local models (middle and lower axis).

4 Approach based on clustering

The convexification of the previous section guarantees that the global solution of the relaxed problem (12) is found. However it does not guarantee that an global optimum or a near global optimum of problem (9)is obtained. It can be seen that the solution found is often strongly suboptimal and will yield a poor description of the partition. Another approach to tackle the optimization problem is to use a stepwise procedure. In the first step a local regression problem will be solved and in the second step the obtained solution is used to enforce the constraints of the optimization problem. Afterwards the obtained validity functions can be used to reestimate the parameters of the local models.

Local Smoothing In the first step the local regression problem for each point in the input space is solved. Therefore the problem

$$\mathbf{l}_i = \left(\mathbf{X}^T\mathbf{Q}_i\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Q}_i\mathbf{y} \tag{13}$$

with the local $N \times N$ weighting matrix

$$\mathbf{Q}_i = \begin{bmatrix} \nu_i(\mathbf{u}(1)) & 0 & \dots & 0 \\ 0 & \nu_i(\mathbf{u}(2)) & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \nu_i(\mathbf{u}(N)) \end{bmatrix} \tag{14}$$

is solved. Hereby the regressor $\mathbf{X}$ and the output vector $\mathbf{y}$ are defined as

$$\mathbf{X} = \begin{pmatrix} 1 & \mathbf{u}(1)^T \\ 1 & \mathbf{u}(2)^T \\ \vdots & \vdots \\ 1 & \mathbf{u}(N)^T \end{pmatrix} \qquad \begin{pmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{pmatrix}. \tag{15}$$

This approach is equivalent to local smoothing and thus strongly depends on the width σ of the local validity function. If σ is chosen to small the flexibility is increased and it is possible that the matrix $\left(\mathbf{X}^T\mathbf{Q}\mathbf{X}\right)$ can be ill-conditioned. Furthermore, the higher σ is chosen the nearer the solution of (13) will be to the global regression problem $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}\mathbf{y}$. The parameter σ is a hyperparmeter of the algorithm. It can be determined with crossvaliditation or alternatively based on geometric considerations. A good choice can be to use a factor proportional to the median of the distances of the points in the input space.

Clustering in the Parameter Space The local parameters now minimize the objective of the optimization problem described by (9) but do not fulfill the constraints. To ensure the satisfaction of the constraints the obtained solutions for the parameters is now restricted to m different values. If the nearest m nearest parameters to the $\mathbf{l}_i$ are of interest these can be found by optimizing

$$\underset{\mathbf{w}_i}{\text{minimize}} \sum_{i=1}^{N} \min_{\mathbf{c} \in \mathcal{W}} \|\mathbf{c} - \mathbf{l}_i\|_2^2 \tag{16}$$

with the set $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$ containing the m possible parameter vectors for the respective local model. This problem is the well studied and beside its non-convexity can be solved with the k-means algorithm [10].

Reestimation of the Local Models After the parameters are clustered it is possible to reestimate the parameters of the local models. Therefore, the parameter σ which describes the lengthscale of the membership functions can be altered. A good choice has been to use $\sigma^* = \frac{\sigma}{3}$ for the reestimation.

5 Exemplification of the Approach Based on Clustering

To show the applicability of the approach first the one-dimensional example used for the illustration of the behavior of the 1 norm based heuristic will be investigated. Afterwards a two dimensional example is used to illustrated the results of the clustering.

5.1 One Dimensional Example

To illustrate the behavior of the approach based on clustering we use the same example as for the 1-norm regularized example. The result is illustrated in Fig. 3. Here the results for the noise free identification with m varied between 1 and 6 models is presented. The first graph shows the data used for modeling and the response of the identified model for different M. In the last two figures the parameter values for the respective local model is shown. It can be seen that compared to the heuristic based on the 1 norm the constraints of the optimization

problem are met. The parameters of the different local models which are equal lie in groups together. Thus a appropriate regression result is obtained already with 4 different local models.

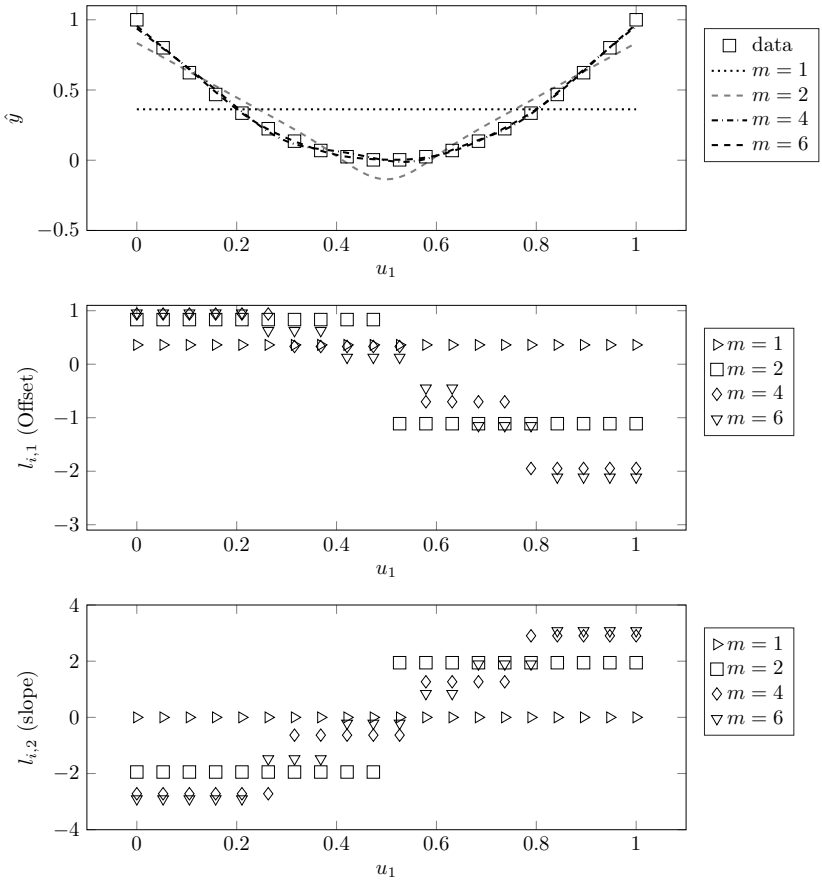


Figure 3: Result of the proposed parameter clustering heuristic for the parabolic example. Regression result for different number of local models m (upper axis), offset and slope of the local models (middle and lower axis).

5.2 Two Dimensional Example

As a two dimensional example the function

$$f(\mathbf{u}) = \frac{1}{(u_1 + u_2)^2} + n \tag{17}$$

where n is i.i.d Gaussian noise with the standard deviation of 0.01. From this function $576 = 24^2$ points on a grid are generated that cover the input spaced equally. The results are shown in Fig. 4. The upper part of the figure shows the training data points and the different marks of the points indicate to which cluster they are affiliated. It can be observed that the merged local validity functions form global validity functions which are not straight and appear especially in the area of the input space with the highest nonlinearity. This is illustrated in the second subfigure which shows the five global validity functions. The process of clustering the local parameters is illustrated in the third figure. Therefore, the parameters which have been obtained for the local validity functions are shown and the affiliation to the different clusters is displayed.

6 Conclusion

In this contribution a novel method for the identification of local model networks has been proposed. The approach constructs the local model network by merging point based validity functions to obtain the validity functions of the local model network. Two heuristics to solve the resulting optimization problem have been presented. The heuristic, which is based on the 1 norm of a difference of the parameter vector norms, which can be seen as a generalization of group lasso, does not show satisfying behavior. An alternative heuristic for the problem has been proposed using a two-step procedure. In the first step a local identification of the local models with point based validity functions is performed and in the second step the global validity functions are obtained by clustering in the parameters or the combined parameter input space. The results obtained with this method on two example problems are promising. A deeper investigation of the properties of the algorithm and its behavior with high dimensional input spaces is subject of further research.

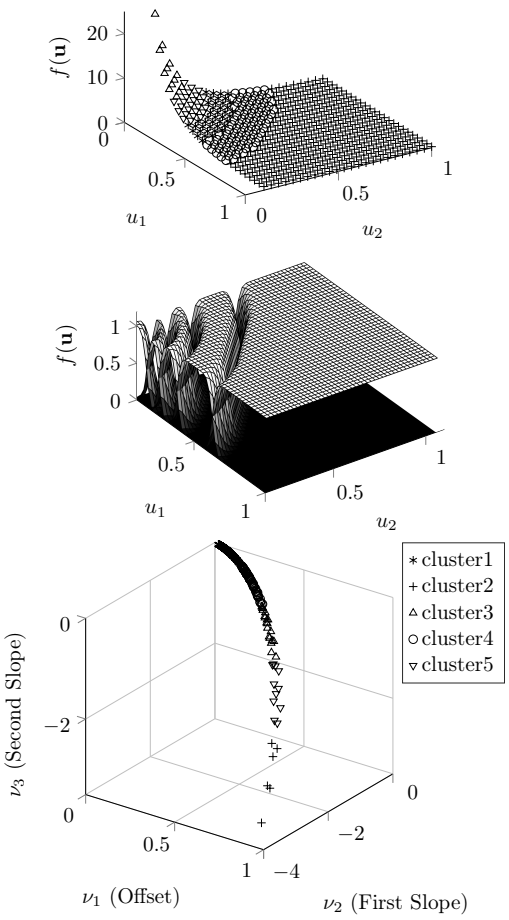


Figure 4: Two dimensional example function different marks of the points indicate the affiliation to a specific cluster

References

[1] Janos Abonyi, Robert Babuska, and Ferenc Szeifert. Modified gath-geva fuzzy clustering for identification of takagi-sugeno fuzzy models. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 32(5):612–621, 2002.

[2] R Babuška, JA Roubos, and HB Verbruggen. Identification of MIMO systems by input-output TS fuzzy models. In *IEEE World Congress on Computational Intelligence*, pages 657–662, 1998.

[3] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[4] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[5] Leo Breiman. Hinging hyperplanes for regression, classification, and function approximation. *Information Theory, IEEE Transactions on*, 39(3):999–1013, 1993.

[6] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and Regression Trees*. CRC Press, New York, 1999.

[7] Giancarlo Ferrari-Trecate, Marco Muselli, Diego Liberati, and Manfred Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217, 2003.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[9] Michael Grant, Stephen Boyd, and Yinyu Ye. Cvx: Matlab software for disciplined convex programming, 2008.

[10] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[11] Tobias Münker and Oliver Nelles. Generalizing piecewise affine system identification to local model networks. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI) (accepted for publication)*, 2017.

[12] Oliver Nelles. *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. Springer Science & Business Media, 2001.

[13] Oliver Nelles. Axes-oblique partitioning strategies for local model networks. In *International Symposium on Intelligent Control*, pages 2378–2383. IEEE, 2006.

[14] Oliver Nelles and Rolf Isermann. Basis function networks for interpolation of local linear models. In *Decision and Control, 1996., Proceedings of the 35th IEEE Conference on*, volume 1, pages 470–475. IEEE, 1996.

[15] Henrik Ohlsson and Lennart Ljung. Identification of switched linear regression models using sum-of-norms regularization. *Automatica*, 49(4):1045–1050, 2013.

[16] Carl Rasmussen and Christopher Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning series)*. The MIT Press, 2005.

[17] Tomohiro Takagi and Michio Sugeno. Fuzzy identification of systems and its applications to modeling and control. *Systems, Man and Cybernetics, IEEE Transactions on*, (1):116–132, 1985.

[18] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[19] Y Yoshinari, Witold Pedrycz, and K Hirota. Construction of fuzzy models through clustering techniques. *Fuzzy sets and systems*, 54(2):157–165, 1993.

[20] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Einsatz eines Self-Enforcing Networks zur kontrollierten Modellbildung am Beispiel der Bewertung von Lösungen für Mathematikaufgaben

Christina Klüver[1], Björn Zurmaar[2]

[1]CoBASC, Universität Duisburg-Essen
Universitätstr.12, 45117 Essen
E-Mail: christina.kluever@uni-due.de

[2]Paluno, Universität Duisburg-Essen
Gerlingstr. 16, 45127 Essen
E-Mail: bjoern.zurmaar@uni-due.de

1 Einleitung

Das Self-Enforcing Network (SEN) ist ein selbst-organisiert lernendes Netzwerk, das von der Forschungsgruppe CoBASC (Computer Based Analysis of Social Complexity) für Klassifikationsprobleme entwickelt wurde (Klüver und Klüver 2013; 2014). Die zu trainierenden Objekte und die korrespondierenden Attribute werden in einer sog. „Semantischen Matrix“ abgebildet. Die Datenbasis kann importiert werden, wie es im Falle großer Datenmengen der Fall ist, oder manuell erstellt werden, indem für jedes Objekt der Zugehörigkeitsgrad eines Attributes angegeben wird. Darüber hinaus wurde ein sog. „cue validity factor“ (cvf) eingeführt, um Attribute, die für die Analyse besonders wichtig sind, gesondert zu gewichten (Klüver, 2016). Für die Modellierung realer Probleme besteht der Vorteil von SEN darin, dass die Gewichtswerte durch eine spezifische Lernregel aus der semantischen Matrix abgeleitet werden, wodurch die Gewichtsmatrix eine genaue Abbildung der semantischen Matrix darstellt. Anhand unterschiedlicher Aktivierungsfunktionen und der Angabe der Lernrate kann untersucht werden, wie sich diese Parameter auf den Lernprozess sowie auf die Ergebnisse auswirken.

Ist der Lernprozess abgeschlossen, werden die Ähnlichkeiten zwischen neuen Inputvektoren und dem trainierten Netz sowohl durch ein Ähnlichkeits- als auch durch ein Distanzmaß berechnet. Die Notwendigkeit, beide Berechnungen zu berücksichtigen, entstand durch die verschiedenen Modelle,

die in SEN zwischenzeitlich implementiert wurden. Es gibt zahlreiche Probleme, bei denen kategoriale Variablen eine wesentliche Rolle spielen und die Ergebnisse anhand des Ähnlichkeitsmaßes am besten reale (Entscheidungs-)Prozesse abbilden (Klüver et al., 2015). Bei Modellen wiederum, die primär metrische Variablen enthalten und bei denen es um eine genaue Differenzierung zwischen den Objekten und deren Attributen geht, ist die Orientierung bei den berechneten Ergebnissen durch das Distanzmaß zu empfehlen.

Interessant wird die Modellierung, wenn die Probleme es erfordern, dass beide Maße zu denselben Ergebnissen führen, wie zum Beispiel im Falle medizinischer Diagnosesysteme (Klüver, 2016), Empfehlungen von Start- und Landebahnen (Klüver et al., 2017) oder bei der Überprüfung von Mathematikaufgaben, die Gegenstand dieses Beitrages sind. Anhand der Berechnungen für das Ähnlichkeits- wie für das Distanzmaß ist es möglich, sofort zu überprüfen, ob das Modell valide ist und die Ergebnisse eindeutig sind oder nicht.

Die Zielsetzung der Entwicklung dieses Modells besteht langfristig darin, dass ein Benutzer seine eigenen Lösungen anhand der Vorgaben von SEN – gewissermaßen den Musterlösungen – sowohl in Bezug auf die einzelnen Schritte als auch auf das Endergebnis überprüfen kann.

In diesem Beitrag wird der Schwerpunkt auf die unterschiedlichen Berechnungen und Visualisierungen gelegt, um eine kontrollierte Modellbildung sowie Analyse der Ergebnisse zu erlauben.

2 Die konkrete Entwicklung eines Modells

Die Modellentwicklung birgt einige Herausforderungen, insbesondere die unmittelbare Validierung des Modells durch SEN, die dazu führen kann, dass die Werte in der semantischen Matrix oder die Anzahl der Attribute überprüft werden sollten. Sofern die zu lernenden Muster in deren Attributen eindeutig differenzierbar sind, stellt sich das Problem nicht, insbesondere falls eine binäre Kodierung vorliegt. Sobald Überlappungen in den Attributen auftreten oder einzelne Muster Attribute aufweisen, die zum Beispiel eine Untermenge eines anderen Musters sind, können bereits Schwierigkeiten bei der Interpretation der Klassifizierung auftreten. Das Problem verschärft sich, falls die Zuordnungen der Attribute zu den Objekten reell kodiert werden.

Der kontrollierte Aufbau eines Modells, in dem die Lösungen von Mathematikaufgaben gelernt und anschließend die Eingaben von Lernenden überprüft werden, soll dies illustrieren. Überwiegend besteht das Problem bei der automatischen Bewertung von Mathematikaufgaben darin, dass die Lösungswege nicht berücksichtigt werden, sondern nur das Endergebnis (Daniel et al., 2014). Die Herausforderung besteht demnach darin, ein Modell für SEN zu entwickeln, das den Lösungsweg bewertet und zugleich, sofern dies möglich ist, unterschiedliche Vorgehensweisen zulässt. Da es bekanntlich durchaus vorgekommen ist, dass eine Aufgabenlösung durch unterschiedliche Lehrer in der Bewertung um mehrere Noten variiert (Holmeier, 2013), wurde, um dieses Problem zu relativieren, ein Lösungsalgorithmus entwickelt, der für alle Aufgaben desselben Typus relevant ist.[1] Dieser Algorithmus kann damit als idealtypische Orientierung für Lehrende wie Lernende dienen, um die individuellen Lösungen daran zu überprüfen und auch zu bewerten.

Bei den hier vorgestellten Aufgaben handelt es sich um die Lösung von Wurzelgleichungen.[2] Insgesamt wurden 10 Aufgaben gestellt in der Form:

$$\sqrt{x-8} - \sqrt{3-x} = 7 \tag{1}$$

$$\sqrt{3x-5} = \sqrt{5x-9} \tag{2}$$

$$\sqrt{2x+1} = -x - 1 \tag{3}$$

Für die Lösung aller Aufgaben musste ein allgemeiner, d.h. vollständiger Lösungsweg berücksichtigt werden, der für SEN vorgegeben wird (Bild 1).

Zusätzlich sind insgesamt 91 Lösungsschritte für die einzelnen Aufgaben als Attribute definiert. Die einzelnen Aufgaben werden als Objekte in der semantischen Matrix repräsentiert; für die Lösungsschritte (Attribute) wird eine 1.0 (erforderlich) bzw. 0.0 (nicht erforderlich) zugewiesen. Mit der linearen Aktivierungsfunktion und einer Lernrate von 0.1 lernt das SEN bereits nach einer Iteration die Lösungsschritte aller Aufgaben. Somit hat jedes Objekt (Aufgabe) insgesamt 105 Attribute, die als 105-dimensionaler binär codierter Vektor dargestellt sind. Zunächst werden alle Aufgaben als Eingabevektoren präsentiert, um das Modell zu validieren.

[1] Der Lösungsalgorithmus wurde von der Forschungsgruppe CoBASC entwickelt.
[2] Die Aufgaben wurden von Alexander Lewintan zur Verfügung gestellt.

Name
1. Definitionsbereich bestimmen
2. Definitionsbereich angeben
3. Definitionsbereich leer
4./7./13./17. Lösungsmenge leer
5. Definitionsbereich nicht leer
6. Lösung bestimmen
8. Äquivalenzumformungen bis x=a
9./18. a liegt in Definitionsbereich
10./19. a ist Lösungsmenge
11./15. a liegt nicht in Definitionsbereich
12./16. a ist nicht Lösungsmenge
14. keine Äquivalenzumformungen x=a
20. Probe

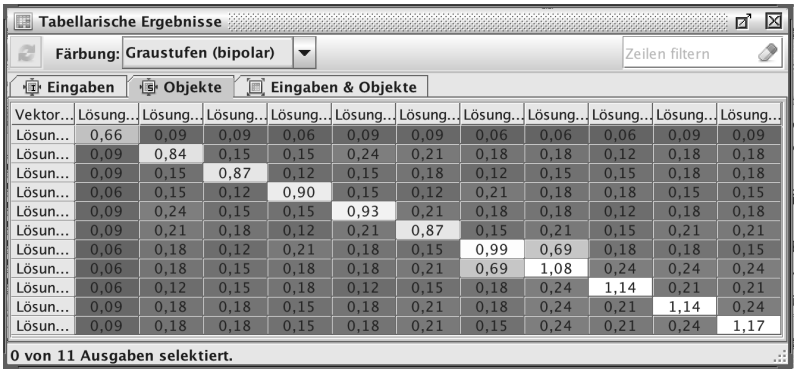
Bild 1: Angabe einzelner Schritte des Lösungsweges als Attribute

3 Ergebnisdarstellungen in SEN

Um eine schnelle Interpretation der Ergebnisse zu ermöglichen, bietet die Software SEN.SE einige Visualisierungsformen an, die anhand des konkreten Modellierungsproblems vorgestellt werden.

Die einfachste jedoch unanschauliche Form findet sich im Fenster Tabellarische Ergebnisse wieder. Dort wird zu jedem Objekt- und Eingabevektor der entsprechende Ausgabevektor angezeigt. Die Zeile identifiziert die jeweilige Eingabe, in den Spalten sind die Ausgabeneuronen aufgetragen. Der Wert in einer bestimmten Zelle gibt also die Endaktivierung des in der jeweiligen Spalte angegebenen Neurons bei der Eingabe des in der jeweiligen Zeile angegebenen Vektors an. Verschiedene Optionen, die Tabellenzelle je nach Wert einzufärben, unterstützen dabei, schnell Muster oder Extremwerte in den Ausgabedaten zu identifizieren. In Bild 2 werden die Aktivierungswerte für alle Aufgaben nach dem Lernprozess dargestellt.

Im Fenster Rangliste werden die Daten einer Zeile der Tabellarischen Ergebnisse aufbereitet und wiedergegeben. Wählt man einen Eingabevektor aus, so wird die Endaktivierung aller Ausgabeneuronen in Form eines Balkendiagramms dargestellt. Die Neuronen werden dabei absteigend gemäß ihrer Endaktivierung sortiert, so dass sich das am stärksten assoziierte Objekt am Beginn der Liste befindet. Somit ermöglicht diese Darstellungsform, aufgrund der Sortierung das Ergebnis der Klassifikation unmittelbar abzulesen. Die Balken mit den aufgetragenen Endaktivierungen vermitteln zudem einen Eindruck von der Eindeutigkeit des Ergebnisses (Bild 3).



Vektor...	Lösung...	Lösung...	Lösung...	Lösung...	Lösung...	Lösung...	Lösung...	Lösung...	Lösung...	Lösung...	Lösung...
Lösun...	0,66	0,09	0,09	0,06	0,09	0,09	0,06	0,06	0,06	0,09	0,09
Lösun...	0,09	0,84	0,15	0,15	0,24	0,21	0,18	0,18	0,12	0,18	0,18
Lösun...	0,09	0,15	0,87	0,12	0,15	0,18	0,12	0,15	0,15	0,18	0,18
Lösun...	0,06	0,15	0,12	0,90	0,15	0,12	0,21	0,18	0,18	0,15	0,15
Lösun...	0,09	0,24	0,15	0,15	0,93	0,21	0,18	0,18	0,12	0,18	0,18
Lösun...	0,09	0,21	0,18	0,12	0,21	0,87	0,15	0,21	0,15	0,21	0,21
Lösun...	0,06	0,18	0,12	0,21	0,18	0,15	0,99	0,69	0,18	0,18	0,15
Lösun...	0,06	0,18	0,15	0,18	0,18	0,21	0,69	1,08	0,24	0,24	0,24
Lösun...	0,06	0,12	0,15	0,18	0,12	0,15	0,18	0,24	1,14	0,21	0,21
Lösun...	0,09	0,18	0,18	0,15	0,18	0,21	0,18	0,24	0,21	1,14	0,24
Lösun...	0,09	0,18	0,18	0,15	0,18	0,21	0,15	0,24	0,21	0,24	1,17

Bild 2: Tabellarische Ergebnisse in SEN. In hellgrau markierte Komponenten zeigen die höchsten Aktivierungswerte.

Das Ergebnis zeigt, dass Aufgabe 7a mit 0.99 am stärksten aktiviert wird, gefolgt von der Lösung für die Aufgabe 7b, die jedoch bereits einen kleineren Aktivierungswert vorweist (0.69). Die anderen Lösungen haben einen deutlich geringeren Aktivierungswert. Diese Darstellung erlaubt die Interpretation, dass die Lösung zur Aufgabe 7a die korrekte Lösung ist.

Die SEN Visualisierung operiert auf den gleichen Daten wie die Rangliste, bietet aber eine andere Darstellungsform. Im Zentrum der Visualisierung steht die jeweilige Eingabe. Proportional zur Endaktivierung der Neuronen werden die jeweiligen Objekte zum Zentrum hin gezogen, so dass die Ähnlichkeit zweier Objekte sich in deren geometrischer Distanz widerspiegelt (Bild 4).

Die Ergebnisse des Rankings werden in dieser Visualisierung anschaulich dargestellt und erlauben eine schnelle Übersicht der „Entfernungen“. Somit ist die Lösung der Aufgabe 1 im Vergleich zu 7a in dieser Berechnungsform als „sehr unähnlich“ zu verstehen.

Während Rangliste und SEN Visualisierung auf einzelnen Ausgabevektoren operieren, stellt das Fenster Distanzen das Verhältnis eines Ausgabevektors einer Eingabe zu den Ausgabevektoren aller Objekte dar, indem es die euklidische Distanz zwischen ihnen berechnet und diese dann in einem Balkendiagramm – ähnlich zur Darstellung im Fenster Rangliste – in aufsteigender Reihenfolge auflistet. Inhaltlich wird also hier nicht nur das Ergebnis einer konkreten Klassifikation betrachtet,

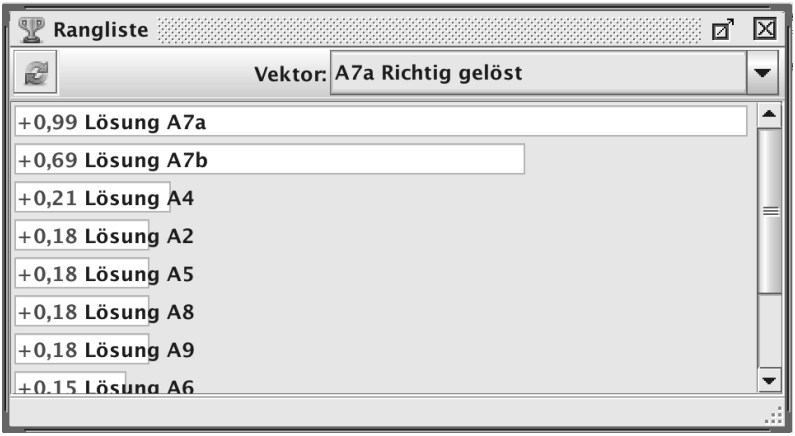


Bild 3: Rangliste in Bezug auf die Lösung einer Aufgabe durch einen Benutzer.

sondern stattdessen wird das aktuelle Klassifikationsergebnis mit anderen bereits bekannten Klassifikationsergebnissen verglichen (Bild 5).

Die Ansicht Distanzen eröffnet somit eine neue, globalere Perspektive auf die Endergebnisse und ermöglicht somit deren Kontrolle im Hinblick auf ihre Plausibilität. Ist das Klassifikationsergebnis der aktuellen Eingabe einem Klassifikationsergebnis eines anderen Objekts als dem am stärksten aktivierten am ähnlichsten, so ist dies ein Indikator für eine unvollständige oder inkonsistente Modellierung. Typische Ursache für diese Situation ist beispielsweise eine stark asymmetrische Verteilung der Gewichtswerte, bei der bestimmte Objekte durch eine extrem hohe oder geringe Summe an eingehenden Verbindungsgewichten bevorteilt, bzw. benachteiligt werden. Ebenso möglich ist, dass sich der zu modellierende Sachverhalt mit den gewählten Attributen noch nicht hinreichend differenzieren lässt.

Die Ergebnisse der beiden Berechnungsformen nebeneinander zeigen, dass sowohl im Ranking als auch in den Distanzen die Ergebnisse dieselbe Klassifizierung in Bezug auf die gelöste Aufgabe vorweisen (Bild 6). Für die Modellentwicklung kann dieses Ergebnis zunächst als zufriedenstellend betrachtet werden.

Wird jedoch eine fehlerhafte Eingabe eines Benutzers als Inputvektor vorgegeben, entsteht die Situation, dass die Klassifizierung zwischen Ranking und Distanzen nicht mehr übereinstimmend ist. Dies ist der Fall,

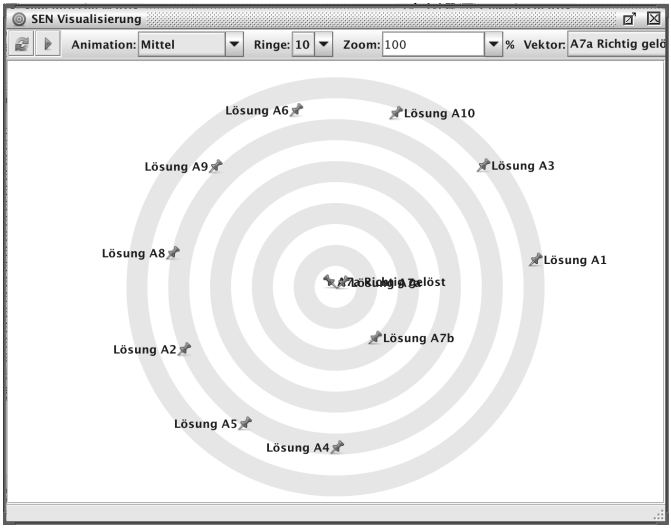


Bild 4: SEN-Visualisierung

da sich etliche Attribute in dem Eingabevektor mit verschiedenen Attributen unterschiedlicher Objekte in der semantischen Matrix überlappen. Folgende Aufgabe (A1) sollte gelöst werden:

$$\sqrt{x-8} - \sqrt{3-x} = 7 \tag{4}$$

Die Lösung beinhaltet etliche Fehler und die Berechnungsergebnisse werden in Bild 7 dargestellt.

In diesem Fall sind die Ergebnisse völlig unterschiedlich, wodurch eine eindeutige Zuordnung nicht mehr möglich ist. Für die Modellierung solcher Probleme bedeutet das Ergebnis, dass das Modell nicht hinreichend ist, um eine eindeutige Klassifizierung für beliebige Lösungen vornehmen zu können.

Eine mögliche Lösung für dieses Problem wurde von Studierenden[3] im Rahmen einer Übung entwickelt. Es wurden 10 zusätzliche Attribute eingefügt, die für die jeweilige Aufgabenstellung stehen. Diese Attribute enthalten einen hohen cue validity factor von 4.0, wodurch repräsentiert

[3]Zakaria Alkhafik, Yannik Beeck, Stephan Napierala und Yavuz Uslubas.

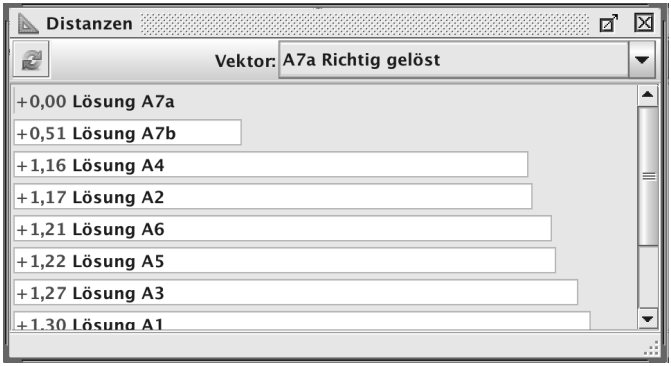


Bild 5: Darstellung der Distanzen

wird, dass die Aufgabenstellung insbesondere zu berücksichtigen ist. Das Ergebnis ist an erster Stelle jeweils erneut eindeutig (Bild 8):

Durch diese Lösung wird stets die Aufgabenstellung berücksichtigt und das Ergebnis zeigt in beiden Modi, dass wesentliche Fehler vorliegen. An welcher Stelle die Fehler erfolgt sind, kann durch diese Visualisierungen und Berechnungen jedoch nicht aufgezeigt werden.

Zu diesem Zweck wurde zuletzt die Eingabeanalyse implementiert, um das Aufspüren von Defiziten in der Modellierung bzw. in den Eingabevektoren zu unterstützen. Diese Analyse operiert nicht auf den Ausgabevektoren, sondern ermöglicht es, einen ausgewählten Eingabevektor komponentenweise mit den Objektvektoren zu vergleichen. Dargestellt wird in den Tabellenzeilen die Differenz zwischen dem Wert des selektierten Eingabevektors und dem jeweiligen Objektvektor. Optional können die Tabellenzeilen eingefärbt werden, wobei die Farbe das Vorzeichen und die Farbintensität den Betrag des Wertes wiedergibt. Dies ermöglicht, schnell Gemeinsamkeiten und Unterschiede zwischen Vektoren zu identifizieren. In den letzten beiden Spalten sind zwei aggregierte Werte zu finden, nämlich die durchschnittliche Differenz zwischen dem gewählten Eingabevektor und dem jeweiligen Objektvektor sowie der durchschnittliche Betrag aller Differenzen. Ersterer erlaubt die Identifikation von Objekten und Vektoren, die über außergewöhnlich starke oder schwache eingehende Verbindungen verfügen, der zweite Wert dient als Indikator für die Unterschiedlichkeit der Vektoren. In den Bildern 9 und 10 wird zuerst die richtige Lösung angezeigt und anschließnd eine mit fehlerhaften Angaben:

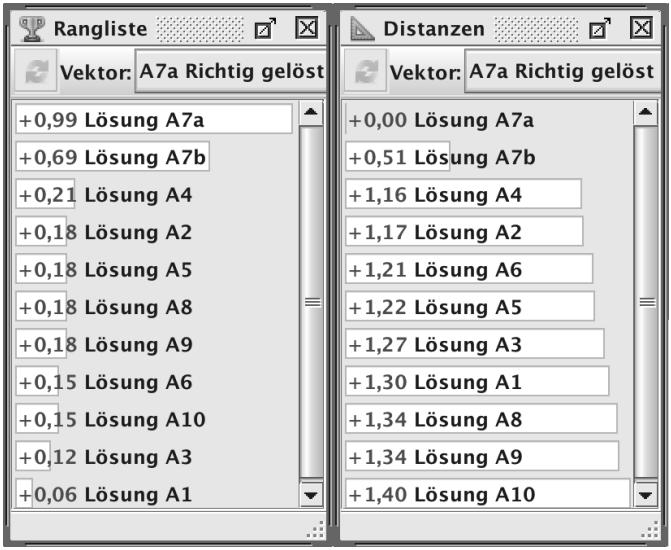


Bild 6: Ranking und Distanzen haben an erster Stelle dieselbe Referenzierung zur Aufgabe 7a

Anhand des Ergebnisses auf der rechten Seite kann man davon ausgehen, dass ein Benutzer, der die Aufgabe 1 lösen sollte, praktisch nur geraten hat.

Durch die Eingabeanalyse ist es zusätzlich möglich, auch die Ergebnisse genau festzuhalten. In diesem Fall wurde ein zusätzliches Attribut „Ergebnis“ in das SEN eingefügt. Dazu wird Aufgabe 4 herangezogen:

$$\sqrt{3x-5} = \sqrt{5x-9} \tag{5}$$

Das richtige Ergebnis ist in diesem Fall $\mathbb{L} = \{2\}$. Durch die Eingabeanalyse wird der Fehler genau angezeigt (Bilder 11 und 12).

Durch die Eingabeanalyse wird demnach jeder Fehler genau angezeigt. Sowohl die Rankings als auch die Distanzen zeigen, dass die Aufgabe 4 jeweils bis auf das Ergebnis fehlerfrei ist. Im Falle, dass eine 3 statt 2 als Lösung eingegeben wird, zeigt die Distanz einen Wert von 0.04 an; im Falle der -2 statt 2 ist die Abweichung von 0.16. Entsprechend differieren auch die Werte im Ranking geringfügig.

Bild 7: Ergebnis des Rankings und der Distanzen bei fehlerhaften Eingaben.

4 Fazit

Die unterschiedlichen Berechnungen und Visualisierungselemente in SEN ermöglichen einen kontrollierten Modellaufbau, indem stets überprüft wird, ob die gelernten Muster bei erneuter Eingabe eindeutig zugeordnet werden können. Im Falle fehlerhafter oder abweichender Eingaben zeigt sich sehr schnell, dass das Modell nicht hinreichend ist, um die Zuordnung der Aufgaben zu erhalten. Der Modellentwickler muss zusätzliche Differenzierungsattribute einfügen, um die Eindeutigkeit zu gewährleisten. Die komponentenbasierte Analyse erlaubt darüber hinaus die Überprüfung der Abweichungen zwischen den Eingaben und der semantischen Matrix. Dies ist sowohl für die Modellentwicklungen als auch für die Analyse und Interpretation der Ergebnisse von besonderer Bedeutung.

Literatur

[1] M. Daniel, N. Köcher, R. Küstenmann. „eKlausuren in der angewandten Mathematik – Herausforderungen und Lösungen“. S. Tratsch, R. Plötzen, G. Schneider, C. Gayer, D. Sassiat, N. Wöhrle (Hrsg.). Lecture Notes in Informatics, DeLFI 2014 – Die 12. e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. (GI) 265-270. 2014

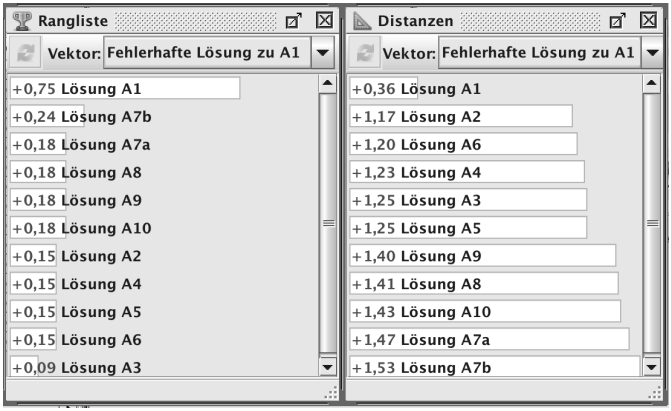


Bild 8: Ergebnis des Rankings und der Distanzen.

[2] Hohlmeier, M. „Leistungsbeurteilung im Zentralabitur.“ Wiesbaden: Springer VS 2014. C. Klüver: „Self-Enforcing Networks (SEN) for the development of (medical) diagnosis systems.“ Proceedings of the IEEE World Congress on Computational Intelligence (IEEE WCCI), Vancouver, pp. 503–510. 2016.

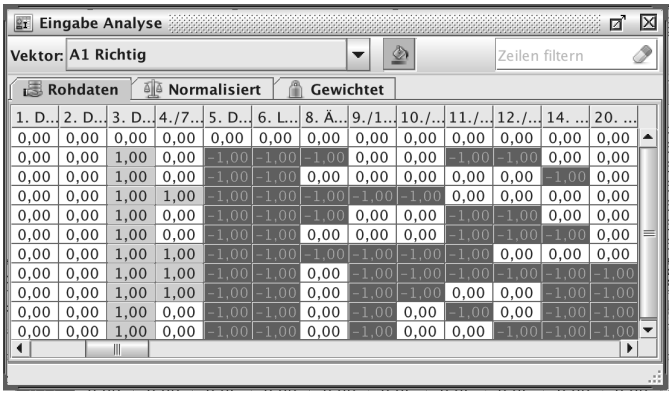
[3] C. Klüver und J. Klüver. „Self-organized Learning by Self-Enforcing Networks.“ In: I. Rojas, G. Joya, and J. Cabestany (Eds.): Advances in Computational Intelligence. 12th International Work-Conference on Artificial Neural Networks, Proceedings, Part I LNCS 7902, Berlin Heidelberg: Springer, pp. 518–529. 2013.

[4] C. Klüver und J. Klüver. „New Learning Rules for Three-layered Feed-forward Neural Networks based on a General Learning Schema.“ In: Madani, K. (Ed.). ANNIIP 2014: International Workshop on Artificial Neural Networks and Intelligent Information Processing. Portugal: Scitepress, pp. 27–36. 2014.

[5] C. Klüver, J. Klüver und B. Zurmaar „OSWI: a consulting system for pupils and prospective students on the basis of neural networks.“ Journal of AI & Society. Volume 30, Issue 1, pp 23–30. Springer, 2015.

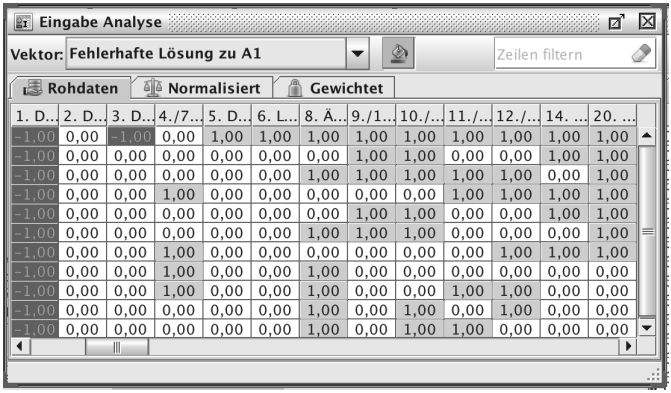
[6] C. Klüver, J. Klüver und D. Zinkhan „ A Self-Enforcing Neural Network as Decision Support System for Air Traffic Control based on Probabilistic Weather Forecasts.“ Proceedings of the IEEE Inter-

national Joint Conference on Neural Networks (IJCNN). Anchorage, pp. 729-736. 2017.



1. D...	2. D...	3. D...	4./7...	5. D...	6. L...	8. Ä...	9./1...	10./...	11./...	12./...	14. ...	20. ...
0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
0,00	0,00	1,00	0,00	-1,00	-1,00	-1,00	0,00	0,00	-1,00	-1,00	0,00	0,00
0,00	0,00	1,00	0,00	-1,00	-1,00	0,00	0,00	0,00	0,00	0,00	-1,00	0,00
0,00	0,00	1,00	1,00	-1,00	-1,00	-1,00	-1,00	-1,00	0,00	0,00	0,00	0,00
0,00	0,00	1,00	0,00	-1,00	-1,00	-1,00	0,00	0,00	-1,00	-1,00	0,00	0,00
0,00	0,00	1,00	0,00	-1,00	-1,00	0,00	0,00	0,00	-1,00	-1,00	-1,00	0,00
0,00	0,00	1,00	1,00	-1,00	-1,00	-1,00	-1,00	-1,00	-1,00	0,00	0,00	0,00
0,00	0,00	1,00	1,00	-1,00	-1,00	0,00	-1,00	-1,00	-1,00	-1,00	-1,00	-1,00
0,00	0,00	1,00	1,00	-1,00	-1,00	0,00	-1,00	-1,00	0,00	0,00	-1,00	-1,00
0,00	0,00	1,00	0,00	-1,00	-1,00	0,00	-1,00	0,00	-1,00	0,00	-1,00	-1,00
0,00	0,00	1,00	0,00	-1,00	-1,00	0,00	-1,00	0,00	0,00	-1,00	-1,00	-1,00

Bild 9: Ergebnis der Eingabeanalyse. Sind alle Werte in einer Zeile 0.0 bedeutet es, dass keine Fehler vorliegen.



1. D...	2. D...	3. D...	4./7...	5. D...	6. L...	8. Ä...	9./1...	10./...	11./...	12./...	14. ...	20. ...
-1,00	0,00	-1,00	0,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
-1,00	0,00	0,00	0,00	0,00	0,00	0,00	1,00	1,00	0,00	0,00	1,00	1,00
-1,00	0,00	0,00	0,00	0,00	0,00	1,00	1,00	1,00	1,00	1,00	0,00	1,00
-1,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	1,00	1,00	1,00	1,00
-1,00	0,00	0,00	0,00	0,00	0,00	0,00	1,00	1,00	0,00	0,00	1,00	1,00
-1,00	0,00	0,00	0,00	0,00	0,00	1,00	1,00	1,00	0,00	0,00	0,00	1,00
-1,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00	1,00	1,00	1,00
-1,00	0,00	0,00	1,00	0,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00
-1,00	0,00	0,00	1,00	0,00	0,00	1,00	0,00	0,00	1,00	1,00	0,00	0,00
-1,00	0,00	0,00	0,00	0,00	0,00	1,00	0,00	1,00	0,00	1,00	0,00	0,00
-1,00	0,00	0,00	0,00	0,00	0,00	1,00	0,00	1,00	1,00	0,00	0,00	0,00

Bild 10: Die hellgraue Markierung weist darauf hin, dass von dem Benutzer eine 1 eingegeben wurde, die in der semantischen Matrix an der jeweiligen Stelle nicht vorliegt. Die dunkelgraue Markierung zeigt entsprechend an, dass in der semantischen Matrix eine 1 vorliegt, diese in der Eingabe jedoch fehlt.

Bild 11: Der Benutzer hat eine 3 statt 2 eingegeben; in der hellgrauen Markierung wird entsprechend eine 1 angegeben.

Bild 12: Der Benutzer hat -2 anstatt 2 angegeben. In der dunkelgrauen Markierung wird entsprechend eine -4 als Abweichung angegeben.

Sequential Possibilistic One–Means Clustering With Variable Eta

Thomas A. Runkler, James M. Keller

Siemens AG
Corporate Technology
CT RDA BAM
81730 Munich, Germany
E-Mail: Thomas.Runkler@siemens.com

University of Missouri–Columbia
Electrical Engineering and
Computer Science Department
Columbia, MO 65211, USA
E-Mail: kellerj@missouri.edu

1 Introduction

Two popular models for finding clusters in data are *fuzzy c–means (FCM)* [2] and *possibilistic c–means (PCM)* [7]. Both models have specific features, which may imply specific advantages and disadvantages when used for finding clusters in data. FCM is more sensitive to noise and outliers [9], and PCM may find coincident clusters [1, 8]. This work is motivated by projects dealing with large numbers of clusters. It was shown recently that none of both models FCM or PCM is well suited for finding large numbers of clusters, and a new iterative clustering method was proposed called *sequential possibilistic one–means (SP1M)* [10]. SP1M was shown to perform very well in finding 100 clusters in the BIRCH data set [13], but the diameters η of the possibilistic clusters were specified manually. In this paper we present an extension of the SP1M algorithm that includes an adaptation mechanism of the parameters η of each cluster. To illustrate the performance of the new algorithm compared with FCM and PCM we present experiments with the original BIRCH data set and a modification of this data set with clusters of different sizes.

2 Alternating cluster optimization

The most common algorithm to optimize the FCM and PCM models is *alternating cluster optimization (ACO)* [9] as shown in Algorithm 1.

Algorithm 1.: Alternating Cluster Optimization

```
1: function ACO(X = {x_1, ..., x_n} ⊂ R^p, c ∈ {2, n − 1}) ▷ data, # clusters
2:     V ← c random vectors in R^p                          ▷ initialize cluster centers
3:     for t = 1 : t_max do
4:         for i = 1 : c do
5:             for k = 1 : n do
6:                 u_ik ← u_ik(V, X)                        ▷ update memberships
7:             v_i ← v_i(U, X)                              ▷ update cluster center
```

The function to update the cluster centers is

$$v_i = \frac{\sum_{k=1}^{n} u_{ik}^m x_k}{\sum_{k=1}^{n} u_{ik}^m} \tag{1}$$

and the functions for updating the memberships are

$$\text{FCM:} \quad u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{\|v_i - x_k\|}{\|v_j - x_k\|} \right)^{\frac{2}{m-1}}} \tag{2}$$

$$\text{PCM:} \quad u_{ik} = \frac{1}{1 + \left(\frac{\|v_i - x_k\|^2}{\eta_i} \right)^{\frac{1}{m-1}}} \tag{3}$$

For FCM this yields Algorithm 2, and for PCM this yields Algorithm 3.

3 Sequential cluster optimization

In the ACO algorithm equation (2) considers all cluster centers $v_1, \ldots, v_c$ in the computation of u_{ik}, whereas equation (3) considers only cluster center v_i. This means that unlike FCM, PCM computes clusters independently of each other, which has motivated *possibilistic one–means (P1M)* [6] as shown in Algorithm 4. So instead of running one instance of PCM we can sequentially run c instances of P1M, and will (given the same initializations) obtain the same results. A similar sequential approach was proposed for noise clustering in [3].

Algorithm 2.: Fuzzy c–Means

function FCM($X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p, c \in \{2, n-1\}$) ▷ data, # clusters
  $V \leftarrow c$ random vectors in $\mathbb{R}^p$ ▷ initialize cluster centers
  for $t = 1 : t_{\max}$ **do**
    for $i = 1 : c$ **do**
      for $k = 1 : n$ **do**
        $u_{ik} \leftarrow \frac{1}{\sum_{j=1}^{c} \left(\frac{\|v_i - x_k\|}{\|v_j - x_k\|} \right)^{\frac{2}{m-1}}}$ ▷ update memberships
      $v_i \leftarrow \frac{\sum_{k=1}^{n} u_{ik}^m x_k}{\sum_{k=1}^{n} u_{ik}^m}$ ▷ update cluster center

Algorithm 3.: Possibilistic c–Means

function PCM($X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p, c \in \{2, n-1\}, \{\eta_1, \ldots, \eta_c\}$)
▷ data, # clusters, cluster sizes
  $V \leftarrow c$ random vectors in $\mathbb{R}^p$ ▷ initialize cluster centers
  for $t = 1 : t_{\max}$ **do**
    for $i = 1 : c$ **do**
      for $k = 1 : n$ **do**
        $u_{ik} \leftarrow \frac{1}{1 + \left(\frac{\|v_i - x_k\|^2}{\eta_i} \right)^{\frac{1}{m-1}}}$ ▷ update memberships
      $v_i \leftarrow \frac{\sum_{k=1}^{n} u_{ik}^m x_k}{\sum_{k=1}^{n} u_{ik}^m}$ ▷ update cluster center

Algorithm 4.: Possibilistic One-Means

function P1M($X = \{x_1, \ldots, n\} \subset \mathbb{R}^p, \eta$) ▷ data, cluster size
  $v \leftarrow$ random vector in $\mathbb{R}^p$ ▷ initialize cluster center
  for $t = 1 : t_{\max}$ **do**
    for $k = 1 : n$ **do**
      $u_k \leftarrow \frac{1}{1 + \left(\frac{\|v - x_k\|^2}{\eta} \right)^{\frac{1}{m-1}}}$ ▷ update memberships
    $v \leftarrow \frac{\sum_{k=1}^{n} u_k^m x_k}{\sum_{k=1}^{n} u_k^m}$ ▷ update cluster center

Both PCM and P1M compute the clusters independently of each other, so both algorithms may yield coincident clusters. This may be undesirable if we want to find many clusters, because then we will have to find even more clusters, from which many will have to be discarded.

Coincident clusters can be avoided by interaction between the clusters. Three different interaction schemes for possibilistic clustering have been proposed in the literature: cluster merging [12, 5], cluster repulsion [11], and sequential clustering with improved initialization [10]. We are currently working on a comparison between these three schemes. In this paper, however, we focus on sequential possibilistic clustering and therefore only consider the third interaction scheme: sequential clustering with improved initialization. This scheme can be viewed as a modification of the P1M algorithm, where cluster centers are not initialized purely randomly, but based on the memberships of the previously found clusters. One instance of this scheme is the *sequential possibilistic one–means (SP1M)* [10] as shown in Algorithm 5, where the initial cluster centers are picked from X with probabilities

$$p(x_k) = \begin{cases} 1/n & \text{if } i = 1 \\ \dfrac{1 - \max\limits_{j=1,\ldots,i} u_{jk}}{n - \sum\limits_{s=1}^{n} \max\limits_{j=1,\ldots,i} u_{js}} & \text{otherwise} \end{cases} \tag{4}$$

4 Adaptation of Cluster Sizes

The PCM, P1M, and SP1M algorithms need the cluster sizes to be specified as input parameters. One of the suggestions in the original PCM paper [7] to adapt the cluster sizes is

$$\eta_i = \frac{\sum\limits_{x_k \in (\pi_i)_\alpha} \|x_k - v_i\|^2}{|(\pi_i)_\alpha|} \tag{5}$$

where $(\pi_i)_\alpha$ is the alpha cut of the i^{th} row of the membership matrix,

$$x_k \in (\pi_i)_\alpha \quad \Leftrightarrow \quad u_{ik} \geq \alpha \tag{6}$$

Algorithm 5.: Sequential Possibilistic One-Means

function SP1M($X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p, c, \eta$) ▷ data, # clusters, cluster size
  $U \leftarrow \emptyset$
  $V \leftarrow \emptyset$
  for $i = 1 : c$ **do**
    repeat
      $v \leftarrow$ probabilistic pick from X using (4) ▷ initialize cluster center
      repeat
        for $k = 1 : n$ **do**
          $u_k \leftarrow \frac{1}{1+\left(\frac{\|v-x_k\|^2}{\eta}\right)^{\frac{1}{m-1}}}$ ▷ update memberships
        $v_{\text{previous}} \leftarrow v$
        $v \leftarrow \frac{\sum_{k=1}^{n} u_k^m x_k}{\sum_{k=1}^{n} u_k^m}$ ▷ update cluster center
      until $\max \|v - v_{\text{previous}}\| < \varepsilon$
    until $\min_{w \in V} \|v - w\| \geq \delta$ ▷ new center
    $U \leftarrow (U, u)$ ▷ append memberships
    $V \leftarrow (V, v)$ ▷ append center

For a rough first initialization of the cluster sizes we suggest to use the standard deviation across all dimensions of the data set computed from the variances as

$$\eta_i = \sqrt{\text{var}_1 X + , \ldots , + \text{var}_p X} \tag{7}$$

$i = 1, \ldots, c$. Inserting this into PCM yields *possibilistic c–means with adaptive eta (PCM–AE)*, see Algorithm 6, and inserting this into SP1M yields *sequential possibilistic one–means with adaptive eta (SP1M–AE)*, see Algorithm 7. Notice that SP1M–AE uses only one variable η but sequentially computes each cluster separately, so each cluster has a different size η_i, $i = 1, \ldots, c$. In our experiments we tried different values of α and obtained the best results with $\alpha = 1/3$.

Algorithm 6.: Possibilistic c–Means with Adaptive Eta

function PCM–AE($X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p, c \in \{2, n-1\}$)
▷ data, # clusters

$V \leftarrow c$ random vectors in $\mathbb{R}^p$ ▷ initialize cluster centers

$\eta_1, \ldots, \eta_c \leftarrow \sqrt{\text{var}_1 X + , \ldots , + \text{var}_p X}$ ▷ initialize cluster sizes

for $t = 1 : t_{\max}$ **do**

  for $i = 1 : c$ **do**

    for $k = 1 : n$ **do**

      $u_{ik} \leftarrow \frac{1}{1 + \left(\frac{\|v_i - x_k\|^2}{\eta_i} \right)^{\frac{1}{m-1}}}$ ▷ update memberships

    $v_i \leftarrow \frac{\sum_{k=1}^{n} u_{ik}^m x_k}{\sum_{k=1}^{n} u_{ik}^m}$ ▷ update cluster center

    $\eta_i \leftarrow \frac{\sum_{x_k \in (\pi_i)_\alpha} \|x_k - v_i\|^2}{|(\pi_i)_\alpha|}$ ▷ update cluster size

5 Experiments

In this section we present our experiments to compare the FCM, PCM–AE, and SP1M–AE algorithms for finding many clusters in data. Two different data sets are considered. The first data set is the BIRCH data set [13] constructed as a two–dimensional array of 10×10 clusters evenly spaced by $4\sqrt{2}$, where each cluster contains 1000 points drawn from a two–dimensional Gaussian distribution with variance $\sqrt{2}$, see Fig. 1 left.

Algorithm 7.: Sequential Possibilistic One-Means with Adaptive Eta

function SP1M–AE($X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p, c$) ▷ data, # clusters
  $U \leftarrow \emptyset$
  $V \leftarrow \emptyset$
  for $i = 1 : c$ **do**
    repeat
      $v \leftarrow$ probabilistic pick from X using (4) ▷ initialize cluster center
      $\eta \leftarrow \sqrt{\mathrm{var}_1 X + , \ldots, + \mathrm{var}_p X}$ ▷ initialize cluster size
      repeat
        for $k = 1 : n$ **do**
          $u_k \leftarrow \frac{1}{1 + \left(\frac{\|v - x_k\|^2}{\eta} \right)^{\frac{1}{m-1}}}$ ▷ update memberships
        $v_{\text{previous}} \leftarrow v$
        $v \leftarrow \frac{\sum_{k=1}^{n} u_k^m x_k}{\sum_{k=1}^{n} u_k^m}$ ▷ update cluster center
        $\eta \leftarrow \frac{\sum_{x_k \in (\pi)_\alpha} \|x_k - v_i\|^2}{|(\pi)_\alpha|}$ ▷ update cluster size
      until $\max \|v - v_{\text{previous}}\| < \varepsilon$
    until $\min_{w \in V} \|v - w\| \geq \delta$ ▷ new center
    $U \leftarrow (U, u)$ ▷ append memberships
    $V \leftarrow (V, v)$ ▷ append center

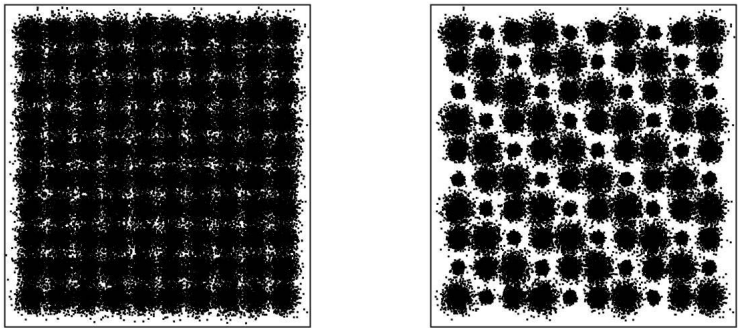


Figure 1: The considered data sets: BIRCH and modified BIRCH.

The second data set is modification of the first data set, where Gaussian distributions with different variances $\sqrt{2}$, $\sqrt{2}/2$, and $\sqrt{2}/3$ are used, so that each diagonal of clusters has the same variance, as shown in Fig. 1 right.

In our first experiment we run the FCM algorithm, $c = 100$, $m = 2$, $t_{\max} = 100$, for the original and the modified BIRCH data sets. Fig. 2 shows the results of these experiments. The top row shows the final cluster centers of one run of FCM each (blue ticks). For the original BIRCH data set (top left diagram) five (of 100) clusters have not been found, and five (of 100) other clusters are covered by two cluster centers each. Notice that for the five multiply covered clusters the cluster centers are not coincident but cover separate halves of the corresponding point cloud. For the modified BIRCH data set (top right diagram) ten (of 100) clusters have not been found, and ten (of 100) other clusters are covered by two (not coincident) cluster centers each. Only one of the missed clusters has variance $\sqrt{2}/3$, and the other nine missed clusters have higher variance, which indicates that clusters with lower point density seem to be more likely to be missed by FCM.

The bottom row shows the number of clusters found over the $t_{\max} \cdot c = 100 \cdot 100 = 10000$ update steps of cluster centers for ten runs of FCM with different random initializations. For the original BIRCH data set (bottom left diagram) FCM finds between 90 and 95 clusters, and for the modified BIRCH data set (bottom right diagram) FCM finds only about 90 clusters.

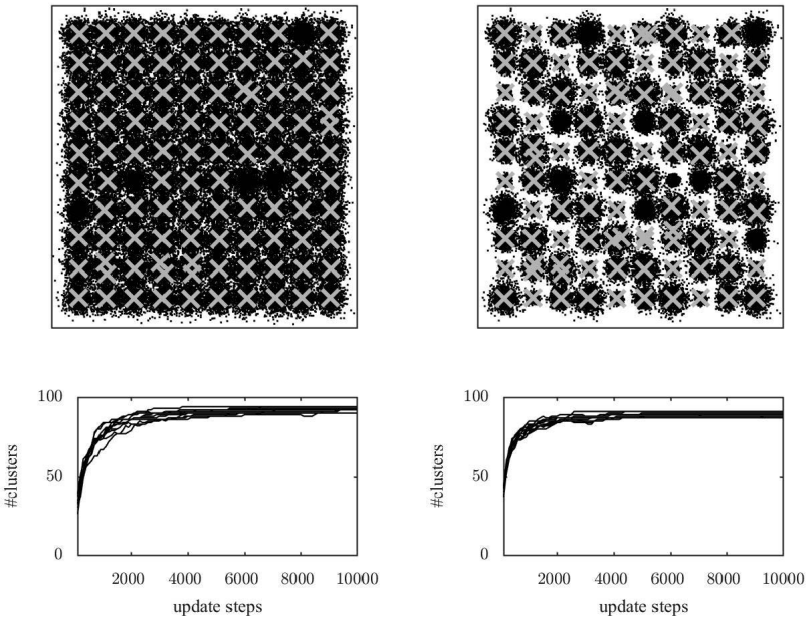


Figure 2: Results for the FCM algorithm.

In our second experiment we run the PCM–AE algorithm, $c = 100$, $m = 2$, $t_{\max} = 100$, $\alpha = 1/3$, for the two considered BIRCH variants (Fig. 3). The first row shows the final cluster centers of one run of PCM–AE each (blue ticks). For the first data set 55 of 100 clusters are found, and many cluster centers are coincident. For the second data set 54 of 100 clusters are found, and many cluster centers are coincident. For this specific data set PCM–AE (unlike FCM) does not seem to prefer dense clusters.

The second row shows the number of clusters for ten runs of PCM–AE. For both data sets PCM–AE always finds only about 55 clusters. After 6000 update steps PCM–AE even loses some of the clusters that have already been found.

The third row shows the values of $\eta_1, \ldots, \eta_c$ for the experiment associated with the first row. Each η_i, $i = 1, \ldots, c$, is updated $t_{\max}$ times, not $t_{\max} \cdot c$, so the number of update steps is only 100 not 10000. After about 30 update steps the adaptation is mostly converged, and the values found

for η_i are between almost zero and about half of the cluster variances, so at termination each possibilistic cluster covers only a part of the corresponding Gaussian distribution.

In our third experiment we run the SP1M–AE algorithm, $c = 100$, $m = 2$, $\epsilon = 0.01$, $\delta = 1$, $\alpha = 1/3$ (Fig. 4). In the results shown in the first row SP1M–AE finds all 100 clusters. The second row shows that this is achieved in all of our runs with ten different initializations. The third row shows that η mostly converges to reasonable values after about 10 to 20 update steps, but in some cases η becomes very large, sometimes higher than 4, and then takes many steps to move back to reasonable values.

6 Conclusions

Our experiments have shown that the approach to adapt the parameter η of possibilistic clusters proposed in [7] is well suited for the sequential possibilistic one–means algorithm proposed in [10]. We obtained the sequential possibilistic one–means algorithm with adaptive eta (SP1M–AE) which greatly outperformed both FCM and PCM–AE in finding clusters in the BIRCH data set and a modification of the BIRCH data set with clusters of different sizes. SP1M–AE can therefore be considered a good choice for finding many clusters in data.

In the future we are planning to extend this work in the following directions: We want to compare our sequential approach with methods based on cluster merging [12, 5] and cluster repulsion [11]. We want to apply our approach to other clustering criteria, such as the modified version of PCM proposed in [4], and also compare it with sequential noise clustering proposed in [3]. Finally, we want to test these methods with higher–dimensional and also real world data sets.

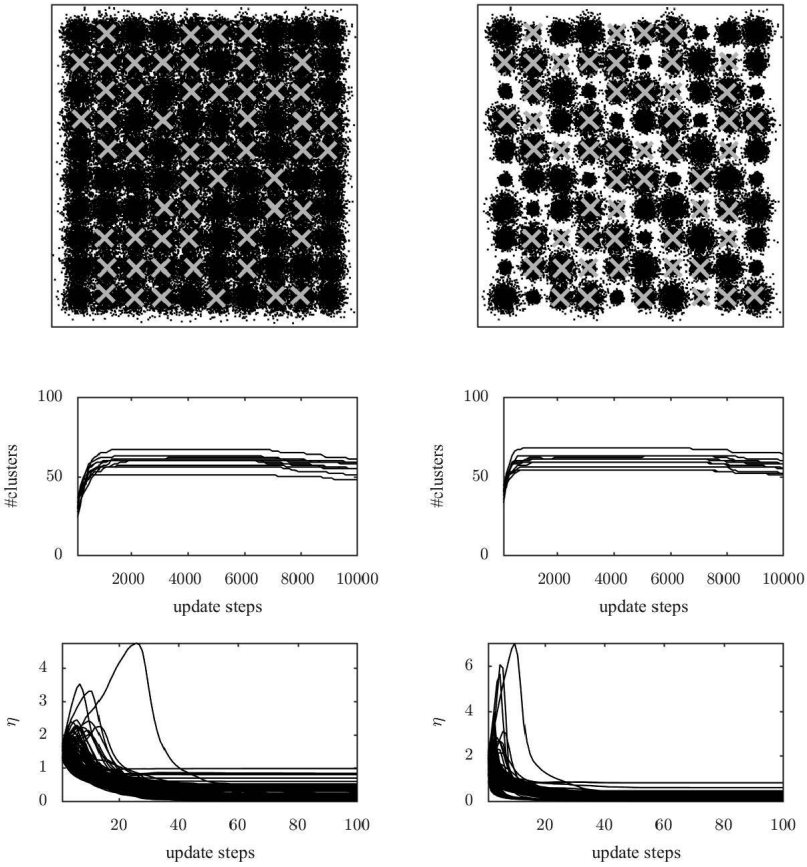


Figure 3: Results for the PCM–AE algorithm.

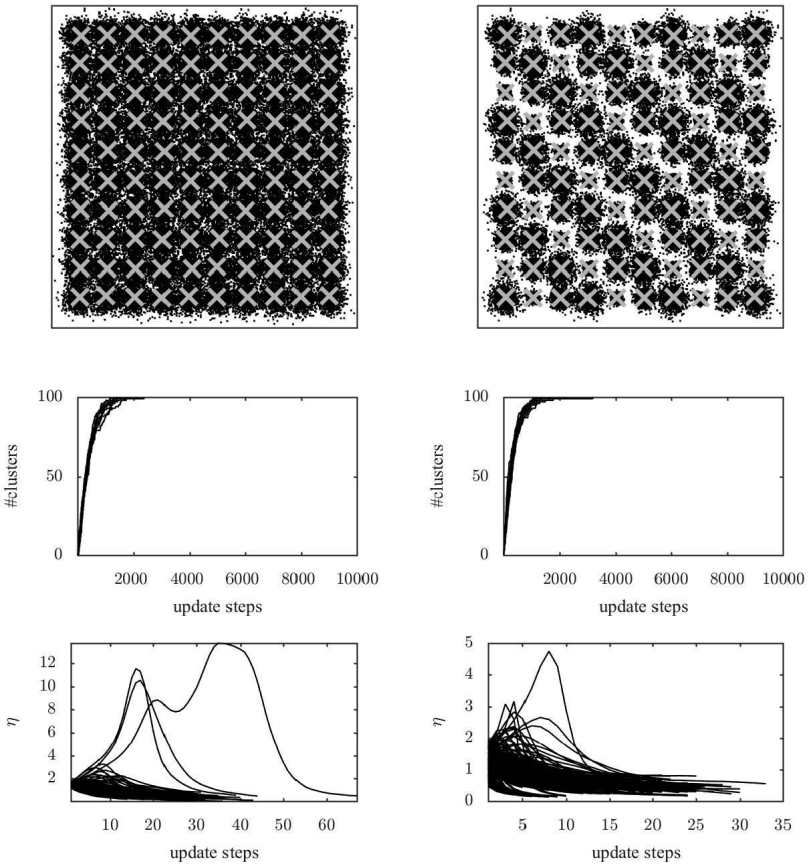


Figure 4: Results for the SP1M–AE algorithm.

References

[1] M. Barni, V. Cappellini, and A. Mecocci. Comments on "A possibilistic approach to clustering". *IEEE Transactions on Fuzzy Systems*, 4(3):393–396, 1996.

[2] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.

[3] R. N. Davé and R. Krishnapuram. Robust clustering methods: A unified view. *IEEE Transactions on Fuzzy Systems*, 5(2):270–293, 1997.

[4] P. Hou, J. Yue, H. Deng, and S. Liu. PCM clustering based on noise level. In *IEEE International Conference on Fuzzy Systems*, Naples, Italy, July 2017.

[5] O. A. Ibrahim, J. Shao, J. M. Keller, and M. Popescu. A temporal analysis system for early detection of health changes. In *IEEE International Conference on Fuzzy Systems*, pages 186–193, Vancouver, Canada, 2016.

[6] J. M. Keller and I. J. Sledge. A cluster by any other name. In *Annual Meeting of the North American Fuzzy Information Processing Society*, pages 427–432, San Diego, 2007.

[7] R. Krishnapuram and J. M. Keller. A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98–110, May 1993.

[8] R. Krishnapuram and J. M. Keller. The possibilistic c–means algorithm: Insights and recommendations. *IEEE Transactions on Fuzzy Systems*, 4(3):385–393, 1996.

[9] T. A. Runkler and J. C. Bezdek. Alternating cluster estimation: A new tool for clustering and function approximation. *IEEE Transactions on Fuzzy Systems*, 7(4):377–393, August 1999.

[10] T. A. Runkler and J. M. Keller. Sequential possibilistic one–means clustering. In *IEEE International Conference on Fuzzy Systems*, Naples, Italy, July 2017.

[11] H. Timm, C. Borgelt, C. Döring, and R. Kruse. An extension to possibilistic fuzzy cluster analysis. *Fuzzy Sets and Systems*, 147(1):3–16, 2004.

[12] M.-S. Yang and C.-Y. Lai. A robust automatic merging possibilistic clustering method. *IEEE Transactions on Fuzzy Systems*, 19(1):26–41, 2011.

[13] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *ACM SIGMOD International Conference on Management of Data*, pages 103–114, 1996.

Zum Optimalen Offline Testsignalentwurf für die Identifikation dynamischer TS-Modelle: Multistufensignale für unsicherheitsminimierte Konklusionsparameter

Matthias Gringard, Andreas Kroll

Fachgebiet Mess- und Regelungstechnik,
Institute for System Analytics and Control,
Fachbereich Maschinenbau,
Universität Kassel
E-Mail: {matthias.gringard,andreas.kroll}@mrt.uni-kassel.de

1 Einführung

In vielen Industriezweigen geht der Trend zum Testen des dynamischen Verhaltens von 100 % der Produkte im gesamten Betriebsbereich. Für den nichtlinearen Fall gibt es nur lokale Ansätze, weshalb es von Interesse ist Testsignale entwerfen zu können, welche den gesamten Betriebsbereich anregen. Weiterhin sind diese Testsignale auch wichtig zur Generierung geeigneter Daten zur Identifikation hochwertiger Modelle, welche zur Simulation oder zum Regelungsentwurf eingesetzt werden können.
In diesem Beitrag wird der optimale Testsignalentwurf für die Identifikation nichtlinearer dynamischer lokal-affiner Fuzzy-Takagi-Sugeno-(TS-)Modelle behandelt. TS-Modelle sind geeignet, da sie universelle Approximatoren sind. Zudem eignen sie sich aufgrund ihrer lokal-affinen Struktur für den Regelungsentwurf.
Der behandelte Ansatz hat zum Ziel, die Unsicherheit der Parameterschätzung zu minimieren. Er basiert auf der Fisher-Informationsmatrix, mit welcher aufgrund der Cramér-Rao-Ungleichung die Kovarianzmatrix abgeschätzt werden kann. Die verwendeten Multistufensignale sind durch Parameter eines Signalmodells kodiert, um eine moderate Anzahl an Optimierungsparametern zu gewährleisten.
Im Vergleich zu den statischen und linear dynamischen FIM-basierten Ansätzen [1] tauchen für nichtlineare dynamische Ansätze Probleme auf, welche in [2, 3, 4, 5] bereits angesprochen wurden. Die FIM eines Modellansatzes, welches nichtlinear in den Parametern ist, enthält die zu diesem Zeitpunkt unbekannten Parameter selbst. Verschiedene Rauschmodelle

haben einen Einfluss auf den Aufbau der FIM. Da die Ausgangsgröße in der FIM enthalten ist, muss die Systemdynamik in der FIM berücksichtigt werden.
In Abschnitt 2 werden entsprechend [6] Modellfamilie und Identifikationsprozess vorgestellt, in Abschnitt 3 werden Problembeschreibung und Lösungsansatz beschrieben und bevor mit einer Zusammenfassung geschlossen wird, wird der Lösungsansatz in Abschnitt 4 anhand einer akademischen Fallstudie unter vereinfachenden Annahmen demonstriert.

2 Modellfamilie und Identifikationsprozess

Es werden zeitdiskrete SISO-TS-Modelle betrachtet. TS-Modelle sind die Superposition lokaler Modelle, welche durch ihre normierten Zugehörigkeitsfunktionen gewichtet werden. Für (N)ARX-Modelle ist das i-te lokale Modell:

$$\hat{y}_i(k) = -\sum_{j=1}^{n} a_{i,j} \cdot y(k-j) + \sum_{l=1}^{m} b_{i,l} \cdot u(k-l-\tau) + c_i \tag{1}$$

$\hat{y}_i(k)$ ist der Ausgang des i-ten Teilmodells, $a_{i,j}$ und $b_{i,l}$ sind die Ein- und Ausgangskoeffizienten, n und m die maximalen Verzögerungen, k die diskrete Zeit und c_i der affine Term. Die Totzeit τ wird im Folgenden nicht betrachtet. Die Differenzengleichung (1) kann umgeschrieben werden zu:

$$\hat{y}_i(k) = \underbrace{\begin{bmatrix} 1 & \boldsymbol{\varphi}_y^{\mathrm{T}}(k-1) & \boldsymbol{\varphi}_u^{\mathrm{T}}(k-1) \end{bmatrix}}_{\boldsymbol{\varphi}^{\mathrm{T}}(k-1)} \underbrace{\begin{bmatrix} c_i & \boldsymbol{\Theta}_{i,y} & \boldsymbol{\Theta}_{i,u} \end{bmatrix}^{\mathrm{T}}}_{\boldsymbol{\Theta}_{\mathrm{LM},i}} \tag{2}$$

$\boldsymbol{\varphi}(k-1)$ und $\boldsymbol{\Theta}_{\mathrm{LM},i}$ sind Regressions- und lokaler Parametervektor. Die lokalen Modelle (2) werden mit ihren von der Schedulingvariable $\mathbf{z}\,(k-1)$ abhängigen Fuzzy-Basisfunktionen $\phi_i(\mathbf{z}(k-1))$ gewichtet und zum Gesamtmodell überlagert. Die Fuzzy-Basisfunktionen sind:

$$\phi_i(\mathbf{z}(k-1)) = \frac{\mu_i\left(\mathbf{z}(k-1)\right)}{\sum_{j=1}^{c} \mu_j\left(\mathbf{z}(k-1)\right)} \tag{3}$$

mit

$$\mu_i\left(\mathbf{z}\,(k-1)\right) = \left[\sum_{j=1}^{c} \left(\frac{||\mathbf{z}\,(k-1) - \mathbf{v}_i||_2}{||\mathbf{z}\,(k-1) - \mathbf{v}_j||_2} \right)^{\frac{2}{\nu-1}} \right]^{-1} \tag{4}$$

im Fall von Zugehörigkeitsfunktionen vom Typ Fuzzy-c-means (FCM). Die Schedulingvariable ist häufig eine Funktion des Regressionsvektors, kann aber auch anders gewählt werden. c ist die Anzahl der lokalen Teilmodelle, die $\mathbf{v}_i$ sind die Partitionsprototypen und $\nu \in \mathbb{R}^{>1}$ ist der Unschärfeparameter. FCM-Zugehörigkeitsfunktionen erfüllen die Orthogonalitätsbedingung, weshalb sie gleich den Basisfunktionen sind:

$$\sum_{i=1}^{c} \mu_i \left(\mathbf{z}\left(k-1\right)\right) = 1 \quad \Rightarrow \quad \phi_i(\mathbf{z}(k-1)) = \mu_i(\mathbf{z}(k-1)) \tag{5}$$

So wird das Gesamtmodell zu:

$$\hat{y}(k|k-1) = \sum_{i=1}^{c} \mu_i \left(\mathbf{z}(k-1)\right) \cdot \boldsymbol{\varphi}^{\mathrm{T}}(k-1) \cdot \boldsymbol{\Theta}_{\mathrm{LM},i} \tag{6}$$

Mit $\mu_i(\mathbf{z}(k-1)) = \mu_{i,k}$ und $\boldsymbol{\varphi}^{\mathrm{T}}\left(k-1\right) = \boldsymbol{\varphi}_{k-1}^{\mathrm{T}}$ folgt aus (6):

$$\hat{y}(k|k-1) = \begin{bmatrix} \mu_{1,k}\boldsymbol{\varphi}_{k-1}^{\mathrm{T}} & \cdots & \mu_{c,k}\boldsymbol{\varphi}_{k-1}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Theta}_{\mathrm{LM},1} & \cdots & \boldsymbol{\Theta}_{\mathrm{LM},c} \end{bmatrix}^{\mathrm{T}} \tag{7}$$

Für N Beobachtungen folgt aus (7):

$$\hat{\mathbf{y}} = \boldsymbol{\Phi}_{\mathrm{E}} \boldsymbol{\Theta}_{\mathrm{LM}} \tag{8}$$

$\boldsymbol{\Phi}_{\mathrm{E}}$ ist die erweiterte Regressionsmatrix, $\boldsymbol{\Theta}_{\mathrm{LM}}$ der Vektor aller lokalen Modellparameter und $\hat{\mathbf{y}}$ der Regressand. Mit bekannten $\mathbf{v}_i$ ist das Schätzproblem linear in den Parametern und die Lösung des OLS-Problems bezüglich einer quadratischen Kostenfunktion ist:

$$\hat{\boldsymbol{\Theta}}_{\mathrm{LM}} = \left(\boldsymbol{\Phi}_{\mathrm{E}}^{\mathrm{T}} \boldsymbol{\Phi}_{\mathrm{E}}\right)^{-1} \boldsymbol{\Phi}_{\mathrm{E}}^{\mathrm{T}} \hat{\mathbf{y}} \tag{9}$$

(9) ist optimal bezüglich des ARX-Modells (Eingangsrauschen) und die Prototypen sind nicht optimal bezüglich des Prädiktionsfehlers. Daher werden in einem folgenden Schritt lokale Modellparameter $\boldsymbol{\Theta}_{\mathrm{LM}}$ sowie die Partitionsparameter $\boldsymbol{\Theta}_{\mathrm{MF}}$ (Prototypen) zu einem Gesamtparametervektor $\boldsymbol{\Theta}$ zusammengefasst und parallel bezüglich einer quadratischen Kostenfunktion in rekursiver Modellauswertung (OE-Modell, Ausgangsrauschen) optimiert:

$$\hat{\boldsymbol{\Theta}} = \arg\min_{\boldsymbol{\Theta}} \sum_{k=1}^{N} \left(y_k - \hat{y}_k\left(\boldsymbol{\Theta}\right)\right)^2 \tag{10}$$

Das Optimierungsproblem wird mit der `MATLAB`-Funktion `lsqnonlin` gelöst, welches einen Trust-Region-Reflective-Algorithmus nutzt. Das Problem wird mit (9) und den in der Clusterung ermittelten Prototypen initialisiert. Die Wahl von ν wird im Detail in [9] diskutiert.

3 Problembeschreibung und Lösungsansatz

In diesem Abschnitt wird die Zielsetzung formuliert und die in Abschnitt 1 angesprochenen Probleme werden erläutert. Im Anschluss daran wird der Lösungsansatz dargelegt. Es ist das Ziel die Stufenhöhen A_l sowie die zugehörigen Haltezeiten $T_{\mathrm{H},l}$ eines Multistufensignals in vorgegebenen Grenzen bezüglich einer Kostenfunktion zu optimieren. Dazu werden die Signalparameter, Stufenhöhen und Haltezeiten, zu einem Signalparametervektor $\boldsymbol{\beta}$ zusammengefasst und folgendes Problem formuliert:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} J(\boldsymbol{\beta}) \tag{11}$$

Die Kostenfunktion J basiert auf der Fisher-Informationsmatrix, mit welcher durch die Cramér-Rao-Ungleichung

$$\mathrm{cov}(\hat{\boldsymbol{\Theta}}) \geq (\mathbf{I}(\boldsymbol{\Theta}))^{-1} \tag{12}$$

die Kovarianzmatrix der Modellparameterschätzwerte $\hat{\boldsymbol{\Theta}}$ bei einem erwartungstreuen Schätzer abgeschätzt werden kann. Somit steht die FIM im Zusammenhang mit der Unsicherheit der Schätzung. Eine „Maximierung der FIM“ führt also zu einer Minimierung der Unsicherheit. In die Kostenfunktion J geht ein skalares Maß auf der FIM ein. Hier wird in einem ersten Schritt üblicherweise die Determinante genutzt (D-Optimalität), da die Determinante der Kovarianzmatrix proportional zum Volumen eines Vertrauensellipsoids ist, welches bei normalverteilten Parameterstreuungen angegeben werden kann.

3.1 Fisher-Informationsmatrix

Das Rauschen wird als additives Messrauschen $n(k)$ angenommen:

$$y(k) = \hat{y}(k) + n(k) \tag{13}$$

Ist $n(k)$ i.i.d.-normalverteilt, kann die FIM formal wie folgt angegeben werden:

$$\mathbf{I} = \frac{1}{\sigma^2} \sum_{k=1}^{N} \left(\frac{\partial \hat{y}(k)}{\partial \boldsymbol{\Theta}} \right) \cdot \left(\frac{\partial \hat{y}(k)}{\partial \boldsymbol{\Theta}} \right)^{\mathrm{T}} \Bigg|_{\boldsymbol{\Theta}=\boldsymbol{\Theta}_0} \tag{14}$$

Hierbei ist σ^2 die Varianz des Messrauschens, $\hat{y}(k)$ die Gesamtmodellgleichung (6), $\boldsymbol{\Theta}$ kennzeichnet alle Parameter bezüglich welcher die FIM

aufgestellt werden soll, $\boldsymbol{\Theta}_0$ ist der wahre Parametervektor und N ist die Anzahl der Beobachtungen.
Bei der Benutzung eines nichtlinearen dynamischen Modellansatzes treten zwei Probleme auf, welche unter anderem in [2] angesprochen wurden. Das erste Problem ergibt sich durch Parameter, welche nichtlinear in die Modellgleichungen eingehen. Dass die Größe, nach welcher abgeleitet wird, verschwindet, ist eine Eigenschaft von Funktionsgleichungen, welche linear in diesen Größen sind. Liegt also ein nichtlinearer Zusammenhang zwischen Modellparametern und Modellgleichung vor, enthält Ableitung die Modellparameter. Dies führt insofern zu einem Problem, dass zur Bestimmung eines optimalen Testsignals, welches zur Generierung geeigneter Daten zur Schätzung der Modellparameter eingesetzt wird, die wahren Modellparameter benötigt werden.
Das zweite Problem ist durch den dynamischen Charakter des Modellansatzes bedingt und ist abhängig von der Wahl des Rauschmodells beziehungsweise der Art der Modellauswertung. Die Problematik soll an einem einfachen Beispiel eines ARX-Modell verdeutlicht werden:

$$\hat{y}_k = \Theta_1 \cdot y_{k-1} + \Theta_2 \cdot y_{k-2} + \Theta_3 \cdot u_{k-2} \tag{15}$$

Der ARX-Modellansatz zeichnet sich dadurch aus, dass zur Prädiktion gemessene Werte genutzt werden. Dadurch sehen die Ableitungen eines ARX-Modellansatzes nach den Parametern dem eines statischen linearen Modells ähnlich.

$$\frac{\partial \hat{y}_k}{\partial \Theta_1} = y_{k-1} \qquad \frac{\partial \hat{y}_k}{\partial \Theta_2} = y_{k-2} \qquad \frac{\partial \hat{y}_k}{\partial \Theta_3} = u_{k-2} \tag{16}$$

Bei der Auswertung eines entsprechenden OE-Modells werden dagegen zur Prädiktion keine Messwerte genutzt sondern vorherige Prädiktionswerte:

$$\hat{y}_k = \Theta_1 \cdot \hat{y}_{k-1} + \Theta_2 \cdot \hat{y}_{k-2} + \Theta_3 \cdot u_{k-2} \tag{17}$$

Wird nun die Ableitung gebildet, so muss entsprechend der Produktregel abgeleitet werden, was exemplarisch für Θ_1 dargestellt ist:

$$\frac{\partial \hat{y}_k}{\partial \Theta_1} = \hat{y}_{k-1} + \Theta_1 \cdot \frac{\partial \hat{y}_{k-1}}{\partial \Theta_1} + \Theta_2 \cdot \frac{\partial \hat{y}_{k-2}}{\partial \Theta_1} + \Theta_3 \cdot \frac{\partial u_{k-2}}{\partial \Theta_1} \tag{18}$$

So gelangen auch bei einem linearen Modellansatz bei einem OE-Modellansatz die unbekannten wahren Parameter in die FIM, weshalb die Kostenfunktion nicht mehr nur von den zu optimierenden Signalparame-

tern $\boldsymbol{\beta}$ sondern auch den geschätzten Modellparametern selbst abhängt. Da die unbekannten wahren Parameter durch Schätzwerte ersetzt werden, ist die Lösung des Optimierungsproblems nur lokal gültig.

3.2 Berücksichtigung der Systemdynamik in der Kostenfunktion

Wie sowohl die Ableitungen eines ARX-Modells (16) als auch die eines OE-Modells (18) zeigen, enthält die FIM für beide Modellansätze die Regressoren, welche im Allgemeinen auch vergangene Werte der Ausgangsgröße y umfassen. Dies bedeutet, dass für jede Iteration der Eingangsgröße u die Ausgangsgröße y in die FIM eingeht. So hängt das Optimierungsproblem (11) nicht nur statisch von den Signalparametern sondern auch von der Systemantwort bezüglich des Testsignals ab:

$$J\left(\boldsymbol{\beta}\right) = J\left(\mathbf{u}\left(\boldsymbol{\beta}\right), \mathbf{y}\left(\mathbf{u}\left(\boldsymbol{\beta}\right)\right)\right) \tag{19}$$

Hierbei ist $\mathbf{u}$ das gesamte Testsignal und $\mathbf{y}$ das entsprechende Ausgangssignal. Für ein technisches System müsste somit in jedem Iterationsschritt ein neues Experiment durchgeführt werden, um die FIM korrekt aufstellen zu können. Da diese Vorgehensweise impraktikabel ist, wird die reale Ausgangsgröße $\mathbf{y}$ durch die Prädiktion eines zuvor geschätzten Modells ersetzt:

$$J\left(\boldsymbol{\beta}\right) = J\left(\mathbf{u}\left(\boldsymbol{\beta}\right), \hat{\mathbf{y}}\left(\mathbf{u}\left(\boldsymbol{\beta}\right), \hat{\boldsymbol{\Theta}}\right)\right) \tag{20}$$

mit

$$\hat{\mathbf{y}} = \left[\hat{y}_k\right] = \left[\sum_{i=1}^{c} \mu_{i,k}(\hat{\boldsymbol{\Theta}}_{\mathrm{MF}}) \cdot \hat{y}_{i,k}(\hat{\boldsymbol{\Theta}}_{\mathrm{LM}})\right] \tag{21}$$

Durch (21) wird die Systemdynamik in der Kostenfunktion berücksichtigt. Auch hier wird deutlich, dass das Optimierungsproblem durch die Verwendung der geschätzten Modellparameter $\hat{\boldsymbol{\Theta}}$ nur lokal gültig ist.

3.3 Testsignalkodierung

Bereits im prozessmodellfreien Testsignalentwurf in [6] wurden die verwendeten Testsignale durch Parameter eines Signalmodells kodiert. Die Parameter dieser Testsignale wurden dann zum einen mit Signaleigenschaften und zum anderen mit der Identifikationsqualität in Verbindung gesetzt.

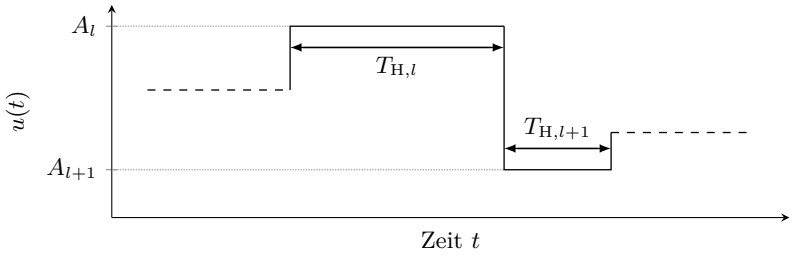


Bild 1: Exemplarische Darstellung der Testsignalparametrierung

Die Kodierung wurde eingesetzt, da Testsignale häufig $N \in \left[10^3; 10^5\right]$ Einzelwerte haben und die interpretierbaren Signaleigenschaften keine hinreichenden Nebenbedingungen bilden, um den Entwurf eines geeigneten Testsignals sicherzustellen. Durch die Kodierung wird die Anzahl der Parameter deutlich reduziert.
Auch für den optimalen Testsignalentwurf (11) werden parametrische Signalmodelle eingesetzt. Selbst wenn die direkte Optimierung der N Einzelwerte des Testsignals zu einer besseren Lösung führen kann, ist sie impraktikabel. Weiterhin wird nicht berücksichtigt, dass es Vorgaben zum Betrieb von Prüfständen geben kann, wie zum Beispiel einer bestimmten Bandbreite oder den Ausschluss bestimmter Signaltypen wie Nicht-Stufensignale. Vorgaben dieser Art können durch die Benutzung eines parametrischen Signalmodells deutlich einfacher erreicht werden. Die Parameter der verwendete Stufensignale sind die Stufenhöhen sowie die Haltezeiten. Damit ist die Signallänge für diese Parametrierung nicht festgelegt. Bild 1 zeigt die Testsignalkodierung exemplarisch.

3.4 Vorgehensweise

Zentrale Probleme, die bei der Verwendung eines FIM-basierten Ansatzes auftreten, lassen sich auf den Einfluss der unbekannten, wahren Modellparameter $\boldsymbol{\Theta}_0$ zurückführen. Diese Parameter werden bei der Optimierung durch zuvor geschätzte Parameter ersetzt, welche aus vorangegangen Identifikationen resultieren, die nicht auf prozessmodellbasiert optimierten Testsignalen beruhen. Damit die Testsignaloptimierung den gewünschten Effekt erzielt, ist es notwendig, dass die geschätzten Modellparameter bereits hinreichend genau sind, wie in [4] für die Partitionsparameter eines statischen TS-Modells bereits gezeigt werden konnte. Übergeordnet

handelt es sich hierbei um einen iterativen Prozess. Mit einem optimierten Testsignal kann ein weiteres Experiment durchgeführt werden und Parameter können geschätzt werden, um eine weitere Optimierung zu veranlassen.
Der vollständige Identifikationsprozess besteht also aus zwei verschachtelten Optimierungen. Die äußere ist die Optimierung der Modellparameter basierend auf einem Identifikationsdatensatz, die innere ist die Optimierung des Testsignals basierend auf dem zuvor geschätzten Modell. Neben den Modellparametern spielen auch strukturelle Entscheidungen bei beiden Optimierungen eine Rolle. Dazu zählen die Anzahl der lokalen Teilmodelle, die Wahl und Parametrierung der Zugehörigkeitsfunktionen und die Wahl der dynamischen Ordnungen. Diese müssen im Vorfeld festgelegt werden. Für die Parameterschätzung und die Testsignaloptimierung werden diese Informationen als bekannt vorausgesetzt und angenommen, dass mit der so entstehenden Modellstruktur das zu identifizierende System im Betriebsbereich hinreichend genau beschrieben werden kann. Eine systematische Abweichung aufgrund struktureller Fehlentscheidungen kann nicht durch die unsicherheitsbasierte Testsignaloptimierung ausgeglichen werden.
Die äußere Optimierung wird entsprechend Abschnitt 2 durchgeführt. Für die innere Optimierung muss eine Kostenfunktion (11) aufgebaut werden, welche das zu minimierende Maß auf der FIM sowie die Nebenbedingung enthält.

3.5 Aufbau der FIM für ein dynamisches TS-Modell

Ausgehend von (14) werde die FIM dargestellt als:

$$\mathbf{I} = \frac{1}{\sigma^2} \sum_{k=1}^{N} \mathbf{I}_k, \quad \mathbf{I}_k = \left(\frac{\partial \hat{y}(k)}{\partial \boldsymbol{\Theta}} \right) \cdot \left(\frac{\partial \hat{y}(k)}{\partial \boldsymbol{\Theta}} \right)^{\mathrm{T}} \Bigg|_{\boldsymbol{\Theta}=\boldsymbol{\Theta}_0} \tag{22}$$

So kann das Kernstück, das dyadische Produkt der Gradienten des Modellansatzes bezüglich der Modellparameter, gesondert betrachtet werden. Da die Partitionsparameter nichtlinear und die lokalen Modellparameter linear in die Modellgleichung eingehen, ist es sinnvoll den Gradienten wie folgt darzustellen:

$$\frac{\partial \hat{y}(k)}{\partial \boldsymbol{\Theta}} = \left[\frac{\partial \hat{y}(k)}{\partial \boldsymbol{\Theta}_{\mathrm{LM}}} \quad \frac{\partial \hat{y}(k)}{\partial \boldsymbol{\Theta}_{\mathrm{MF}}} \right]^{\mathrm{T}} \tag{23}$$

Für einen (N)ARX-Modellansatz lässt sich die Modellgleichung beschreiben als:

$$\hat{y}(k) = \sum_{i=1}^{c} \mu_{i,k}\left(\boldsymbol{\Theta}_{\mathrm{MF}}\right) \boldsymbol{\varphi}_{k-1}^{\mathrm{T}} \boldsymbol{\Theta}_{\mathrm{LM},i} = \underbrace{\left[\mu_{1,k}\boldsymbol{\varphi}_{k-1}^{\mathrm{T}} \cdots \mu_{c,k}\boldsymbol{\varphi}_{k-1}^{\mathrm{T}}\right]}_{\boldsymbol{\varphi}_{\mathrm{E},k-1}^{\mathrm{T}}} \boldsymbol{\Theta}_{\mathrm{LM}} \quad (24)$$

Die Ableitungen bezüglich der lokalen Modellparameter können damit geschrieben werden als:

$$\frac{\partial y(k)}{\partial \boldsymbol{\Theta}_{\mathrm{LM}}} = \boldsymbol{\varphi}_{\mathrm{E},k-1} \quad (25)$$

Für die Beschreibung der Ableitung der Zugehörigkeitsfunktionen nach den Partitionsparametern gibt es keine vereinfachende Schreibweise, da jede Zugehörigkeitsfunktion von allen Partitionsparametern abhängt. Die Ableitungen der Zugehörigkeitsfunktionen von Typ FCM wurden in [2] vorgestellt. Für einen (N)OE-Modellansatz ergibt sich bei der Ableitung bezüglich der lokalen Modellparameter ein weiterer Term:

$$\frac{\partial y(k)}{\partial \boldsymbol{\Theta}_{\mathrm{LM}}} = \boldsymbol{\varphi}_{\mathrm{E},k-1} + \frac{\partial \boldsymbol{\varphi}_{\mathrm{E},k-1}}{\partial \boldsymbol{\Theta}_{\mathrm{LM}}} \cdot \boldsymbol{\Theta}_{\mathrm{LM}} \quad (26)$$

Der Ausdruck $\frac{\partial \boldsymbol{\varphi}_{\mathrm{E},k-1}^{\mathrm{T}}}{\partial \boldsymbol{\Theta}_{\mathrm{LM}}}$ ist die Jacobimatrix des Regressionsvektors bezüglich der lokalen Modellparameter. Was in (18) nicht explizit angesprochen wurde, ist, dass der Regressionsvektor vergangene Werte der Ausgangsgröße enthält, welche selbst wieder von vergangenen Werten der Ausgangsgröße abhängen bis hin zum ersten Wert der Ausgangsgröße. Die Notwendigkeit der Rekursion durch die Zeit ist für die Schätzung der Parameter eines OE-Modells oder eines Künstlichen Neuronalen Netzes schon seit längerer Zeit bekannt. Für den optimalen FIM-basierten Testsignalentwurf ist dieses Problem vergleichsweise neu. Es wird beispielsweise in [5] für einen Online-Testsignalentwurf behandelt. Aufgrund der hohen Komplexität wird daher zunächst die FIM mit einem ARX-Modellansatz aufgebaut:

$$\mathbf{I} = \frac{1}{\sigma^2} \begin{bmatrix} \boldsymbol{\Phi}_{\mathrm{E}}^{\mathrm{T}} \boldsymbol{\Phi}_{\mathrm{E}} & \sum_{k=1}^{N} \boldsymbol{\varphi}_{\mathrm{E},k-1} \left(\frac{\partial y(k)}{\partial \boldsymbol{\Theta}_{\mathrm{MF}}}\right)^{\mathrm{T}} \\ \sum_{k=1}^{N} \left(\frac{\partial y(k)}{\partial \boldsymbol{\Theta}_{\mathrm{MF}}}\right) \boldsymbol{\varphi}_{\mathrm{E},k-1}^{\mathrm{T}} & \sum_{k=1}^{N} \left(\frac{\partial y(k)}{\partial \boldsymbol{\Theta}_{\mathrm{MF}}}\right) \left(\frac{\partial y(k)}{\partial \boldsymbol{\Theta}_{\mathrm{MF}}}\right)^{\mathrm{T}} \end{bmatrix} \quad (27)$$

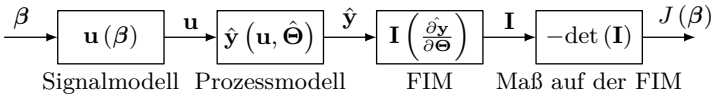


Bild 2: Serieller Ablauf der Berechnung der Kostenfunktion

3.6 Implementierung

Das Optimierungsproblem (11) wurde in `MATLAB` implementiert. Es wurde der Ansatz gewählt die Einzelnen Rechenschritte, welche zur Berechnung der zu minimierenden Größe führen, sukzessive zu berechnen. Zusätzlich müssen der Zielfunktion neben den zu optimierenden Signalparametern $\boldsymbol{\beta}$ noch die Schätzwerte der Modellparameter ($\hat{\boldsymbol{\Theta}}$), die Strukturparameter des Modells c, ν, n, m sowie die Abtastzeit T_A übergeben werden. Durch den Einsatz einer Partikelschwarmoptimierung (PSO) zur Lösung ist es nicht nötig, Initialwerte für die Signalparameter vorzugeben. Bild 2 zeigt das Ablaufdiagramm für die Kostenfunktion. Zuletzt ist die Ausgabe der Kostenfunktion nicht ein Maß auf der inversen FIM sondern das negative Maß. Da die Determinante der FIM sehr große Werte annimmt ist der Kehrwert der Determinante sehr klein, was eine relative Betrachtung von Funktionswerten erschwert. Die Lage der Extrema ändert sich dadurch nicht, die Berechnung jedoch erleichtert. Das Kriterium J kann beliebig erweitert werden. Die Determinante einer Matrix ist das Produkt der Eigenwerte. So wird beim D-optimalen Entwurf nicht berücksichtigt, dass sich kleine und große Unsicherheiten kompensieren und eine optimales Testsignal eine schlechte Schätzung bestimmter Parameter liefern kann. Neben dem D-optimalen Entwurf sind insbesondere die A- und E-optimalen Entwürfe bekannt, welche darauf Abzielen die Summe der Eigenwerte (A) und den größten Eigenwert (E) der Kovarianzmatrix zu minimieren.
Zur Lösung des Optimierungsproblems wird die in `MATLAB` vorimplementierte PSO `particleswarm` genutzt. Eine PSO basiert darauf, dass sich eine Anzahl von Partikeln (Lösungskandidaten) im Lösungs- oder Suchraum bewegen. Aus den Bewegungsrichtungen und Positionen der Partikel ergeben sich in jedem Schritt neue Lösungskandidaten. Hierbei hat jeder Partikel ein Gedächtnis für die individuelle sowie die globale Bestlösung. Die Aktualisierung der Bewegungsrichtung ergibt sich dann aus einer gewichteten Überlagerung der letzten Bewegungsrichtung, der Richtung von der aktuellen Position zur individuellen Bestlösung und der Richtung

von der aktuellen Position zur globalen Bestlösung. Aktualisierung der Bewegungsrichtung des i-ten Partikels $\mathbf{v}_{i,s+1}$:

$$\mathbf{v}_{i,s+1} = \omega_s \cdot \mathbf{v}_{i,s} + \varphi_p r_p \left(\mathbf{p}_{i,\text{best}} - \mathbf{p}_{i,s}\right) + \varphi_g r_g \left(\mathbf{p}_{\text{global}} - \mathbf{p}_{i,s}\right) \quad (28)$$

Hierbei ist ω_s die Gewichtung der Trägheit der Bewegung, φ_p und φ_g heissen kognitiver und sozialer Gewichtungsfaktor, $\mathbf{p}_{i,\text{best}}$ ist die individuelle Bestlösung, $\mathbf{p}_{\text{global}}$ die globale Bestlösung, $\mathbf{p}_{i,s}$ ist die aktuelle Position und r_p und r_g sind Gewichtungsfaktoren, welche in jedem Schritt zufällig gewählt werden. Die Aktualisierung der Position des i-ten Partikels ist dann:

$$\mathbf{p}_{i,s+1} = \mathbf{p}_{i,s} + \mathbf{v}_{i,s+1} \quad (29)$$

Die Trägheit wird adaptiv in Abhängigkeit der Änderung der individuellen Bestlösung angepasst. Neben den Standardabbruchkriterien wie maximaler Rechenzeit, maximale Iterationen oder Erreichen eines vorgegebenen Zielwerts bricht die PSO ab, wenn die Änderung der globalen Bestlösung sich nach einer festzulegenden Anzahl an Iterationen unter einem Schwellenwert befindet oder sich die globale Bestlösung in einem festzulegenden Zeitraum nicht mehr ändert. Nachdem die PSO stoppt wird eine Hybridfunktion (hier: `fmincon`) mit der globalen Bestlösung initialisiert.
Für alle durchgeführten PSO wurden folgende Einstellungen genutzt: Zufällige Initialisierung der Startpositionen und Bewegungsrichtungen, Zielfunktionsschwellenwert: $\epsilon = 10^{-6}$, Anzahl der Iterationen in denen die Änderung des Zielfunktionswerts der Bestlösung unter dem Schwellenwert liegen muss: 20, Wertebereich der Trägheit: $\omega \in [0{,}1; 1{,}1]$, Gewichtungsfaktoren: $\varphi_p = \varphi_g = 1{,}49$, Schwarmgröße $N = 10 \cdot \text{AnzahlDerVariablen}$. Alle globalen Zeit- und Iterationsbegrenzungen wurden entfernt.

4 Fallstudie

In diesem Abschnitt wird die vorgestellte Methode anhand eines akademischen Testsystems demonstriert. Die Nichtlinearität ist der eingangsgrößenabhängigen Dämpfung bei hydraulischen Antrieben nachempfunden und durch [7] inspiriert. Die Identifikationsergebnisse werden basierend auf prozessmodellfrei und -basiert entworfenen Testsignalen verglichen. Bei den verwendeten Testsignalen handelt es sich um Multistufensignale.

Tabelle 1: Parametrierung des Testproblems

Systemeigenschaften	
Typ	Fuzzy-TS-Modell
Zugehörigkeitsfunktionen	FCM-Typ
Anzahl der Teilmodelle	$c = 5$
Scheduling-Variable	Eingangsgröße u
Prototypen	$\{v_i\} = \{-2; -1; 0; 1; 2\}$
Unschärfeparameter	$\nu = 1{,}5$
Dynamikordnungen	$n = 2$, $m = 2$
Eigenkreisfrequenz	$\omega_0 = 10\,\frac{\mathrm{rad}}{\mathrm{s}}$
Abtastzeit	$T_\mathrm{A} = 0{,}01\,\mathrm{s}$
Verstärkungen	$\{K_i\} = \{6; 1{,}5; 3; 7{,}5; 4{,}5\}$
Dämpfungsmaße	$\{D_i\} \approx \{0{,}45; 0{,}71; 0{,}2; 0{,}58; 0{,}32\}$
Ausgangsrauschen	$\mathcal{N} \sim \left(0; 0{,}25^2\right)$

Für die Entwurfsmethodiken werden verschiedene Stufenanzahlen betrachtet. Die Parameter der Testsignale die Stufenhöhen und Haltezeiten. Das Testsystem ist als TS-Modell vorgegeben, so dass die Identifikation anhand der Parameterschätzwerte bewertet werden kann.

4.1 Beschreibung des Testsystems

Beim akademischen Testsystem handelt es sich um ein Fuzzy-TS-Modell, welches sich aus der Überlagerung von $c = 5$ verschiedenen PT_2-Systemen mit eingangsgrößenabhängiger Dämpfung und Verstärkung ergibt. Tabelle 1 zeigt die zusammengestellten Systemeigenschaften.

Die Dämpfungsmaße und Verstärkungen wurden nicht in einer auf- oder absteigenden Reihenfolge gewählt, um den Übergang zwischen zwei benachbarten Teilmodellen signifikanter zu gestalten. Die lokalen Teilmodelle werden durch die Diskretisierung eines kontinuierlichen PT_2-Systems erhalten. Dazu wird ausgegangen von:

$$\ddot{y}(t) + 2D_i\omega_0\dot{y}(t) + \omega_0^2 y(t) = K_i\omega_0^2 u(t) \tag{30}$$

Mit der Abtastzeit T_A und der Umrechnungsvorschrift $u(t_k) = u_k$, $y(t_k) = y_k$, $\dot{y}(t_k) = \left(y_{k+1} - y_k\right)/T_\mathrm{A}$, $\ddot{y}(t_k) = \left(y_{k+2} - 2y_{k+1} + y_k\right)/T_\mathrm{A}^2$ wird (30) diskretisiert. Da der Prädiktionswert im Modellansatz den k-ten Zeitschritt belegt, wird zudem eine diskrete Zeitverschiebung durchgeführt:$u_k \to u_{k-2}$, $y_k \to y_{k-2}$, $y_{k+1} \to y_{k-1}$, $y_{k+2} \to y_k$. Mit

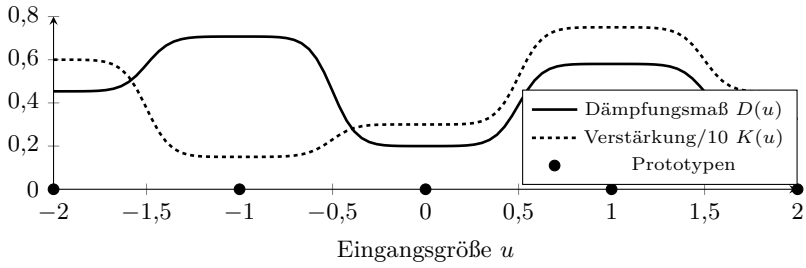


Bild 3: Dämpfung und Verstärkung in Abhängigkeit der Eingangsgröße

der Einführung des Prädiktionswerts $y_k = \hat{y}_{k,i}$ folgt die Gleichung des i-ten Teilmodells:

$$\hat{y}_{k,i} = \underbrace{(2 - 2D_i\omega_0 T_\mathrm{A})}_{a_{1,i}} y_{k-1} - \underbrace{\left(2D_i\omega_0 T_\mathrm{A} - \omega_0^2 T_\mathrm{A}^2 - 1\right)}_{a_{2,i}} y_{k-2} + \underbrace{K_i\omega_0^2 T_\mathrm{A}^2}_{b_{2,i}} u_{k-2} \tag{31}$$

Entsprechend der zuvor eingeführten Schreibweise ergibt sich die Darstellung des i-ten lokalen Teilmodells:

$$\hat{y}_{i,k} = \begin{bmatrix} y_{k-1} & y_{k-2} & u_{k-2} \end{bmatrix} \begin{bmatrix} a_{1,i} & a_{2,i} & b_{2,i} \end{bmatrix}^\mathrm{T} = \boldsymbol{\varphi}_{k-1}^\mathrm{T} \boldsymbol{\Theta}_{\mathrm{LM},i} \tag{32}$$

Bild 3 zeigt die bezüglich der Eingangsgröße u nichtlinear verlaufenden Werte für die Dämpfung und die Verstärkung als Überlagerungen aller mit den Zugehörigkeitsfunktionen gewichteten Werten für Dämpfung und Verstärkung der Teilmodelle.

4.2 Auslegung der Testsignale

Die eingesetzten Testsignale sind Multistufensignale. Die festzulegenden Parameter sind die Stufenhöhen A_l (Amplituden) und die Haltezeiten $T_{\mathrm{H},l}$ entsprechend Bild 1, welche sich in jeweils in Tabelle 2 vorgegebenen Bereichen befinden dürfen. Die Anzahl der Stufen R ist ein weiterer Entwurfsparameter, welcher jedoch vor der Optimierung der Amplituden und Haltezeiten festgelegt wird. Tabelle 2 zeigt die Zusammenstellung der Eckdaten des Testsignalentwurfs. Wie in der Tabelle beschrieben, werden die Testsignale auf drei verschiedene Arten ausgelegt. Zum einen gibt es die heuristische Auslegung (A1) basierend auf Signaleigenschaften, wie

Tabelle 2: Eckdaten des Testsignalentwurfs

Testsignaleigenschaften	
Typ	Multistufensignal
Anzahl verschiedener Stufen R	$R \in \{20; 25; 30\}$
Begrenzung Haltezeiten	$T_\mathrm{H} \in [2; 7]$ s
Begrenzung Amplituden	$A \in [-5; 5]$
Auslegung	A1: heuristisch prozessmodellfrei, A2: optimal prozessmodellfrei, A3: optimal prozessmodellbasiert

in [6], dann die optimale Auslegung (A2) basierend auf Signaleigenschaften und die optimale modellbasierte Auslegung (A3), wie in Abschnitt 3 beschrieben. Die Multistufensignale sind ähnlich zu den Amplitudemodulated Pseudo Random Binary Sequences (APRBS). Jedoch werden die Haltezeiten und Amplituden nun gezielt ausgelegt, weshalb nicht mehr von einem APRBS-Signal gesprochen werden kann.

Die signaleigenschaftbasierten Auslegungen bewerten die Eignung des Testsignals anhand des Scheitelfaktors

$$c_\mathrm{r} = \frac{\max |u_k|}{\sqrt{\frac{1}{N} \sum_{k=1}^{N} u_k^2}} \tag{33}$$

sowie einem Maß für die Gleichverteilung der Amplituden.
Bei der heuristischen Auslegung wird das Multistufensignal mit gleichverteilten Amplituden und Haltezeiten initialisiert und die Parameter nacheinander geändert, sofern die Änderung eine Verringerung des Scheitelfaktors zur Folge hat und die Bewertung der Gleichverteilung der Amplituden einen Schwellenwert nicht überschreitet. Das Suchverfahren stoppt, wenn ein Viertel der Parameteränderungen sukzessive keinen Effekt auf die Zielgröße hat.
Für die prozessmodellfreie Optimierung wird eine Zielfunktion aus Scheitelfaktor und Abdeckungsmaß mit einer PSO minimiert. Der Scheitelfaktor wird als Maß für die Energiedichte des Signals genutzt. Eine hohe Energiedichte ist grundsätzlich gut, jedoch muss für nichtlineare Systeme eine möglichst gleichmäßige Abdeckung des Wertebereichs der Eingangsgröße berücksichtigt werden, um zu vermeiden, dass die Stufenhöhen nur den Signalbegrenzungen entsprechen.
Für den modellbasierten Entwurf wurden vereinfachende Annahmen getroffen beziehungsweise vereinfachende Gegebenheiten ausgenutzt. Da

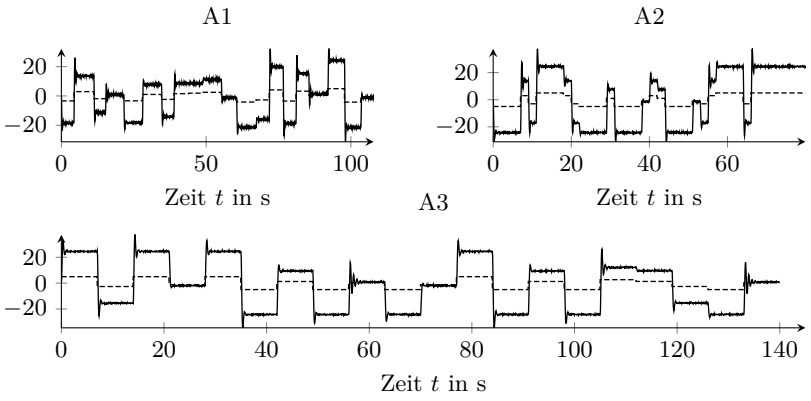


Bild 4: Gestrichelt: exemplarische Testsignale, durchgezogen: Systemantworten

es sich um ein akademisches Testsystem handelt, welches als Fuzzy-TS-Modell vorgegeben ist, sind die wahren Parameter bekannt. Weiterhin wird die Testsignaloptimierung nur bezüglich der lokalen Modellparameter $\boldsymbol{\Theta}_{\mathrm{LM}}$ durchgeführt, so dass die FIM nur die erweiterte Regressionsmatrix $\boldsymbol{\Phi}_{\mathrm{E}}^{\mathrm{T}}$ enthält. Diese hängt zwar von den Partitionsparametern $\boldsymbol{\Theta}_{\mathrm{MF}}$ ab, welche jedoch bekannt sind. Auch die dynamische Nebenbedingung wird präzise eingehalten, da kein geschätztes Modell zur Berechnung der Ausgangsgröße y während der Optimierung genutzt werden muss, sondern das wahre System herangezogen werden kann, was bei der Identifikation eines technischen Systems nicht möglich wäre. Die Berechnung der prozessmodellfreien Testsignale dauert unter einer Sekunde (heuristisch) und bis zu 10 Sekunden mit PSO, wohingegen die Berechnung der prozessmodellbasierten Testsignale länger als einen Tag dauern kann. Bild 4 zeigt exemplarisch die unterschiedlich entworfenen Testsignale. Ein Unterschied, welcher sich bei jedem modellbasiert entworfenen gegenüber den prozessmodellfrei entworfenen zeigt, ist die Haltezeit. Obwohl konstante Ausgangsgrößen den Informationsgehalt im Sinne der FIM nicht erhöhen, da sie auf die Anzahl der Datenpunkte bezogen ist, wird stets die maximal zulässige Haltezeit von $T_{\mathrm{H}} = 7\,\mathrm{s}$ verwendet. In Tabelle 3 sind die Scheitelfaktoren aller entworfenen Testsignale gegenübergestellt. Für die prozessmodellfrei entworfenen Signale ist der Scheitelfaktor ein Gütekriterium, bezüglich welchem die Auslegung stattgefunden hat. Es ist zu sehen, dass die prozessmodellbasiert entworfenen Signale einen

leicht höheren Scheitelfaktor haben, aber insgesamt noch nah am theoretischen Optimum ($c_\mathrm{r} = 1$) liegen. Dies lässt die Annahme zu, dass der Scheitelfaktor **ein** wichtiges Bewertungskriterium ist.

Tabelle 3: Scheitelfaktoren der Testsignale

Stufen	A1	A2a	A2b	A3
20	1,7436	1,1352	1,1463	1,3390
25	1,7883	1,1594	1,1352	1,2956
30	1,5244	1,1476	1,1340	1,2807

4.3 Versuchsdurchführung und Darstellung der Ergebnisse

Für die 12 entworfenen Testsignale werden jeweils $N_\mathrm{exp} = 100$ Experimente durchgeführt, um die Kovarianzmatrix der Parameterschätzwerte empirisch bestimmen zu können. Da das Modell strukturell identisch mit dem akademischen Testsystem ist und das Rauschen am Ausgang (OE-Modell) angenommen wird, sind die Parameterschätzwerte nicht nur näherungsweise sondern auch theoretisch normalverteilt. Die Parameter werden durch die Optimierung eines quadratischen Gütemaßes bezüglich des Prädiktionsfehlers in rekursiver Modellauswertung bestimmt (OE-Schätzung). Für die Optimierung wird die `MATLAB`-Funktion `lsqnonlin` eingesetzt. Aus diesem Grund ist es zulässig die Verteilung der Parameterschätzwerte basierend auf verschiedenen Experimenten in Form einer Gaussglocke darzustellen.
Da die wahren Modellparameter bekannt sind, können sie den Schätzwerten direkt gegenübergestellt werden. Zusätzlich wird die Determinante der Kovarianzmatrix sowie die L_2-Norm der Abweichung des geschätzten vom wahren Parametervektor herangezogen. Bild 5 zeigt am Beispiel des Parameters a_{22}, wie die Schätzwerte des Parameters bezüglich der durchgeführten Versuche um den wahren Wert streuen. Die Abweichung des Mittelwerts ist vernachlässigbar. Mit Ausnahme einer kleinen Abweichung des Mittelwerts sind die Varianzen aller prozessmodellbasierten Experimente zum einen ähnlich und zum anderen deutlich kleiner als die der prozessmodellfrei entworfenen Experimente. Bild 7 zeigt analoge Ergebnisse für alle Parameter. Dargestellt ist das Verhältnis der Varianzen aller Parameterschätzungen der prozessmodellfrei entworfenen Experimente zu den Varianzen der zugehörigen prozessmodellbasiert entworfenen Experimente. Es ist zu sehen, dass durch den modellbasierten Entwurf die Varianz jedes Parameters mindestens halbiert wird.

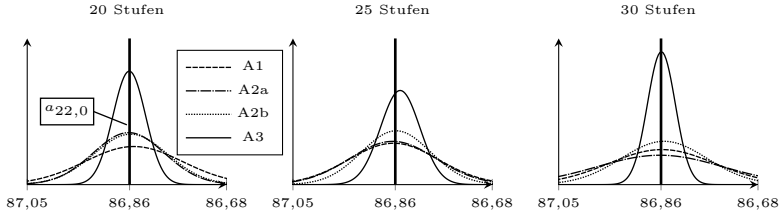


Bild 5: Parameterschätzwerte von a_{22} in -10^{-2}

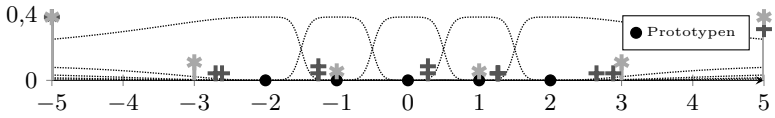


Bild 6: Stufenhöhen bezüglich der Teilmodelle, gestrichelt: skalierte Zugehörigkeitsfunktionen, Kreuz: FIM-basiert, Stern: prozessmodellfrei

Mit Ausnahme des dritten Teilmodells (Parameter a_{13}, a_{23} und b_{23}) ist zwischen den unterschiedlichen prozessmodellfreien Entwürfen bezüglich der Varianz kein signifikanter Unterschied erkennbar. Dass die Parameter des dritten Teilmodells mit den prozessmodellfrei optimierten Signalen für alle Stufenanzahlen deutlich schlechter geschätzt werden als in den anderen Experimenten ist dadurch begründet, dass bei der Optimierung bezüglich des Scheitelfaktors selbst bei Berücksichtigung der Verteilung der Amplituden diese nicht in die Nähe von Null gelegt werden, da dies einen schlechteren Scheitelfaktor bedeutet und der Prototyp des dritten Teilmodells bei $u = 0$ liegt. Tabelle 4 zeigt wieviele Datenpunkte prozentual am stärksten zu den jeweiligen Teilmodellen gezählt haben. Es ist zu erkennen, dass für die Varianten A2a und A2b nicht nur kein Datenpunkt am stärksten dem dritten Teilmodell zugerechnet wurden sondern auch die benachbarten Teilmodelle 2 und 4 vergleichsweise schwach vertreten sind, wie Bild 6 exemplarisch zeigt. Die intuitive Annahme, dass ein längeres Testsignal zu einer besseren Identifikation führt, konnte anhand der durchgeführten Versuche nicht bestätigt werden.

Die L_2-Norm ist ein Maß für die Abweichung des mittleren geschätzten Parametervektors bezüglich des wahren Parametervektors und die Determinante ist ein Maß für die Unsicherheit. In Tabelle 5 und Bild 8 sind diese Größen dargestellt. Für das Balkendiagramm wurden die Werte für die Determinante so angepasst, dass alle Werte größer als eins sind, so

Tabelle 4: Prozentualer Anteil der Datenpunkte im Teilmodell mit der größten Zugehörigkeit

		Prozentuale Zugehörigkeit der Datenpunkte zu den Teilmodellen				
	Stufen	v_1	v_2	v_3	v_4	v_5
A1	20	37,45	4,11	16,89	6,13	35,42
	25	48,64	15,75	7,52	12,10	15,99
	30	40,86	19,30	10,30	6,60	22,95
A2a	20	45,00	5,00	0	5,00	45,00
	25	45,00	4,00	0	6,00	45,00
	30	43,21	8,00	0	1,60	47,19
A2b	20	41,33	2,67	0	8,00	48,00
	25	45,00	2,00	0	8,00	45,00
	30	45,04	3,34	0	6,67	44,94
A3	20	40,00	10,00	10,00	15,00	25,00
	25	30,00	8,24	12,35	12,35	37,06
	30	37.56	10.24	10.24	10.24	31.71

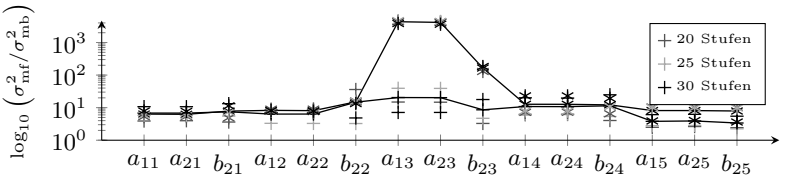


Bild 7: Schätzvarianzen der prozessmodellfreien Identifikationen bezogen auf die prozessmodellbasierten

dass für alle Werte ein positiver Logarithmus resultiert. Es lassen sich drei Dinge erkennen. Zum einen wird kein Trend deutlich, der vermuten lässt, dass eine Erhöhung der Stufenanzahl (Signaldauer) eine Verbesserung des Identifikationsergebnisses zur Folge hat, dies gilt sowohl für die prozessmodellfrei entworfenen als auch für die prozessmodellbasiert entworfenen Experimente. Zwischen dem schlechtesten Ergebnis für die Determinante für den prozessmodellbasierten Entwurf und dem besten prozessmodellfreien Entwurf liegt ein Faktor der Größenordnung 10^{11}. Auch fallen die Experimente bezüglich der Signale mit 25 Stufen bei über der Hälfte der Identifikationsergebnisse deutlich schlechter aus. Zu diesem Zeitpunkt kann dafür kein Grund agegeben werden. Zum anderen sind die optimalen Experimente bezüglich des Unsicherheitsmaßes eine deutliche Verbesserung. Selbst bei dem Maß für die absolute Abweichung liegt der schlechteste Wert der optimalen Experimente nur unwesentlich über den Werten der prozessmodellfreien Experimente, obwohl durch den Ansatz ein Bias-Fehler nicht gezielt reduziert wird. Weiterhin ist

Tabelle 5: Tabellarische Gesamtergebnisse

	Stufen	A1	A2a	A2b	A3
$L_2 \cdot 10^3$	20	0,1732	0,1165	0,1312	0,0701
	25	0,1423	0,6514	1,0170	0,1665
	30	0,4024	0,6970	0,3917	0,0853
det	20	8,4496E−126	6,3481E−127	3,3944E−127	9,0190E−138
	25	5,4914E−121	1,3621E−122	1,1205E−123	1,5639E−138
	30	3,9643E−123	3,8079E−121	7,5622E−124	2,1983E−140

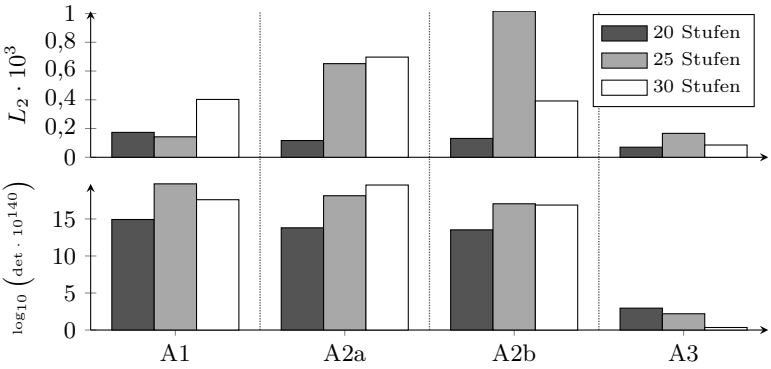


Bild 8: Balkendiagramme der Gesamtergebnisse

die Streuung der prozessmodellfreien Experimente deutlich größer als die der optimalen Experimente. So kann anhand einer ersten einfachen Fallstudie demonstriert werden, dass die grundsätzliche Vorgehensweise eine sichtbare Verbesserung zulässt. Die prozessmodellbasiert optimal entworfenen Multistufensignale sehen anders aus als die bezüglich der Signaleigenschaft entworfenen. Dies kann damit zusammenhängen, dass Multistufensignale dazu gedacht sind, um einen möglichst großen Bereich nichtlinearer dynamischer Systeme anzuregen. Dazu werden aus verschiedenen stationären Zuständen sprunghafte Änderungen aufgeschaltet, da sich bei nichtlinearen Systemen im Allgemeinen bei identischen stationären Anregungen (Stufen) unterschiedliche stationäre Ausgangsgrößen in Abhängigkeit des vorherigen Systemzustands einstellen können. Bei signalmodellbasierten Multistufensignalen wird dies berücksichtigt. Nun ist das nichtlineare Verhalten des Testsystems dieser Fallstudie nur von der Eingangsgröße abhängig, was bedeutet, dass ein stationärer Wert der Eingangsgröße immer zum selben stationären Systemausgang führt.

5 Zusammenfassung und Ausblick

Es wurde eine Vorgehensweise zur Berechnung eines FIM-basierten optimalen Experimententwurfs zur Reduzierung der Unsicherheit der Parameterschätzwerte gegeben und die auftretenden Schwierigkeiten und zugehörigen Lösungsansätze diskutiert. Im Rahmen einer Fallstudie, welche vereinfachenden Annahmen unterlag, konnte die Effektivität des Ansatzes demonstriert werden, indem die prozessmodellbasierten Experimente den prozessmodellfreien gegenübergestellt wurden.
In folgenden Schritten muss untersucht werden, wie robust der Entwurfsprozzes gegenüber Nichterfüllen der vereinfachenden Annahmen ist. Dazu zählen die Annahmen über das Rauschmodell, welche die vereinfachte Berechnung der FIM zulässt, die Benutzung der wahren Ausgangsgröße während der Iterationen der Testsignaloptimierung sowie die Benutzung eines Testsystems, welches in der Modellklasse liegt, was bedeutet, dass zum Aufbau der FIM die bekannten wahren Modellparameter genutzt werden können. Um den Einfluss der Annahmen isoliert betrachten zu können, bieten sich zunächst akademische Testsysteme wie in [8] an. So wird es beispielsweise möglich sein, die Annahmen über das Rauschmodell beizubehalten und die Kostenfunktion weiterhin mit dem wahren System zu aktualisieren. Jedoch würden in die FIM die geschätzten Parameter einfließen. Weitere Arbeiten umfassen die Definition der Zielgröße, welche aktuell die Determinante enthält. Das globale Ziel ist der optimale Entwurf für technische Systeme, welcher robust gegenüber den gebrochenen Annahmen sein muss. Ist die Effektivität gesichert, kann auch die Effizienz mit einbezogen werden. Weiterhin muss auch die Kostenfunktion für den prozessmodellfreien Testsignalentwurf so erweitert werden, dass nicht einzelne Stufenhöhen zu stark benachteiligt werden, wie es bei den prozessmodellfrei optimierten Testsignalen der Fall war.

Literatur

[1] H. Bandemer at. al, *Theorie und Anwendung der optimalen Versuchsplanung*, Berlin: Akademie-Verlag, 1972.

[2] A. Kroll, „Zum optimalen Testsignalentwurf für die Partitionierung und Teilmodellparametrierung dynamischer Takagi-Sugeno-Modelle: Problemstellung und Lösungsansätze“, In: *Proc., 26.*

Workshop Computational Intelligence, Dortmund, Deutschland, S. 97–118, 2016.

[3] T. Heinz und O. Nelles, „Vergleich von Anregungssignalen für Nichtlineare Identifikationsaufgaben“, In: *Proc., 26. Workshop Computational Intelligence*, Dortmund, Deutschland, S. 139–158, 2016.

[4] A. Kroll und A. Dürrbaum, „On joint optimal experiment design for identifying partition and local model parameters of Takagi-Sugeno models“, In: *Proc., 17th IFAC Symposium on System Identification (SysID)*, Beijing, China, S. 1427–1432, 2015.

[5] C. Hametner, M. Stadlbauer, M. Deregnaucourt, M. Jakubek und T. Winsel, „Optimal experiment design based on local model networks and multilayer perceptron networks“, In: *Engineering Applications of Artificial Intelligence*, Band 26, Nr. 1, S. 251–261, 2013.

[6] M. Gringard und A. Kroll, „Zur systematischen Analyse des Einflusses der Parametrierung von Standardtestsignalen für die Identifikation nichtlinearer dynamischer TS-Modelle am Beispiel eines elektromechanischen Stellantriebs“, In: *Proc., 26. Workshop Computational Intelligence*, Dortmund, Deutschland, S. 119–138, 2016.

[7] T. Bernd, M. Kleutges und A. Kroll, „Nonlinear Black Box Modeling - Fuzzy Networks versus Neural Networks“, In: *Neural Computing & Applications*, Band 8, S. 151–162, 1999.

[8] K. S. Narendra und K. Parthasarathy, „Identification and control of dynamical systems using neural networks“, In: *IEEE Transactions on Neural Networks*, Band 1, Nr. 1, S. 4–27, 1990.

[9] A. Kroll, „On choosing the fuzziness parameter for identifying TS models with multidimensional membership functions“, In: *Journal of Artificial Intelligence and Soft Computing Research*, Band 1, Nr. 4, S. 283–300, 2011.

Optimierung von Entwurfsparametern für Takagi-Sugeno Fuzzy Sliding-Mode Beobachter mit Simulated-Annealing Verfahren

Horst Schulte, Eckhard Gauterin

Hochschule für Technik und Wirtschaft Berlin
Ingenieurwissenschaften - Energie und Information
Fachgebiet Regelungstechnik
E-Mail: schulte@htw-berlin.de

Kurzfassung

In dieser Arbeit wird ein Optimierungsalgorithmus zur Bestimmung von Entwurfsparametern für Takagi-Sugeno Fuzzy Sliding-Mode Beobachter basierend auf dem Simulated-Annealing Verfahren vorgestellt. In einem Zweischrittverfahren wird der Lösungsraum mittels Stabilitätskriterien nach Lyapunov eingeschränkt und dem Simulated-Annealing Verfahren als Vorwissen übergeben. Motiviert wird die Vorgehensweise anhand eines bekannten Regelungsproblems aus der Antriebstechnik. Anhand von Simulationen wird die Problemstellung der geeigneten Wahl von Entwurfsparametern erläutert und die Vorteile des Einsatzes von Methoden der Meta-Optimierung aufgezeigt.

1 Einführung

Moderne Verfahren der modellbasierten Regelung ermöglichen es, Vorwissen über den Prozess in das Regelgesetz zu integrieren. Damit können strukturelle Eigenschaften der Regelstrecke z.B. in Form einer Vorsteuerung oder bei der beobachtergestützten Kompensation von Störgrößen berücksichtigt werden. Neben den Streckenparametern, die entweder bekannt oder aus Messdaten geschätzt werden, enthalten Regelgesetze noch Entwurfsparameter, wie Gewichtsfaktoren in Kostenfunktionen [1] oder Eigenwerte des geschlossenen Kreises, welche die Performance einer geregelten Strecke in Abhängigkeit von dem Entwurfsverfahren entscheidend beeinflussen können. Diese frei zu wählenden Entwurfsparameter, unter

Berücksichtigung von zuvor bestimmten Kriterien, stellen den Anwender moderner Entwurfsverfahren vor das Problem, wie diese systematisch bestimmt werden können.
Zunächst wird die allgemeine Struktur von Takagi-Sugeno Sliding-Mode (TS-SM) Beobachtern beschrieben [2], [3], [4]. Danach wird diese Struktur genutzt, um ein bekanntes Regelungs- und Schätzproblem aus der Antriebstechnik zu lösen. Anhand dieser Problemstellung wird gezeigt, welche Schwierigkeiten bei der Parametrierung von Sliding-Mode Beobachtern entstehen können und wie sich diese mit einer geeigneten Meta-Optimierung behandeln und im besten Fall vollständig automatisieren lässt. Abschließend wird der Pseudo-Code eines problemangepassten Simulated-Annealing Algorithmus zur Parametrierung von TS-SM Beobachtern basierend auf dem Standardverfahren [5] vorgestellt.

2 Takagi-Sugeno Sliding-Mode-Beobachter

2.1 Grundstruktur

Die Grundstruktur des Takagi-Sugeno Sliding-Mode Beobachters (TS-SMO) aus [3] enthält eine zeitkontinuierliche und diskontinuierliche Rückführung des Ausgangsfehlers $\mathbf{e}_y = \hat{\mathbf{y}} - \mathbf{y} \in \mathbb{R}^p$:

$$\begin{aligned} \dot{\hat{\mathbf{x}}} &= \sum_{i=1}^{N_r} h_i(\mathbf{z}) \big(\mathbf{A}_i \, \hat{\mathbf{x}} + \mathbf{B}_i \, \mathbf{u} - \mathbf{G}_{l,i} \, \mathbf{e}_y + \mathbf{G}_{n,i} \, \boldsymbol{\nu}_\delta \big), \\ \hat{\mathbf{y}} &= \mathbf{C} \, \hat{\mathbf{x}} \, , \end{aligned} \tag{1}$$

mit dem Schaltterm

$$\boldsymbol{\nu}_\delta = \begin{cases} -\rho \frac{\mathbf{P}_2 \, \mathbf{e}_y}{\| \mathbf{P}_2 \, \mathbf{e}_y \| + \delta} & \text{falls } \mathbf{e}_y \neq \mathbf{0} \\ \mathbf{0} & \text{sonst} \end{cases} . \tag{2}$$

Dieser enthält die positiv-definite Matrix $\mathbf{P}_2$ als Lösung der Lyapunov-Gleichung

$$\mathbf{P}_2 \, \boldsymbol{\mathcal{A}}_{22}^s + \boldsymbol{\mathcal{A}}_{22}^{s\,T} \, \mathbf{P}_2 = -\mathbf{Q}_2 \, , \tag{3}$$

mit $\mathbf{Q}_2 = \mathrm{diag}(1, 1, \cdots, 1)$, vgl. [7]. Bei dem Entwurf des Beobachters müssen die in der Tabelle 1 zusammengefassten Parameter passend gewählt werden. Dabei beeinflussen die Einträge in $\boldsymbol{\mathcal{A}}_{22}^s$ direkt die Verstärkung der Rückführung im zeitkontinuierlichen Term und über (3) indirekt im

Schaltterm (2). Dabei wird die Amplitude über ρ und die Steigung um den Ursprung $\mathbf{e}_y = \mathbf{0}$ mit δ eingestellt [7].

Tabelle 1: Designparameter vom TS-SMO (Grundstruktur)

Beschreibung	Parametersatz
Verstärkung vom Schaltterm (2)	$\rho > 0$
Approximation im Schaltterm (2)	$\delta > 0$
stabile Entwurfsmatrix (3)	$\boldsymbol{\mathcal{A}}_{22}^s = \mathrm{diag}(s_1, \ldots, s_p)$ mit $\mathrm{Re}\{s_i\} < 0$, $i = 1, \ldots, p$

Tabelle 2: Designparameter vom gewichteten TS-SMO (erweiterte Struktur)

Beschreibung	Parametersatz
Verstärkungsvektor d. Schaltterme (4)	$\boldsymbol{\rho}_i = \mathrm{diag}(\rho_{1,i}, \ldots, \rho_{p,i})$
Approximation in d. Schalttermen (4)	$\delta > 0$
stabile Entwurfsmatrix (3)	$\boldsymbol{\mathcal{A}}_{22}^s = \mathrm{diag}(s_1, \ldots, s_p)$ mit $\mathrm{Re}\{s_i\} < 0$, $i = 1, \ldots, p$

2.2 Erweiterte Struktur mit gewichteten Schalttermen

Der erweiterte TS-SMO mit gewichteten Schalttermen unterscheidet sich von der Grundstruktur (2) durch die Gewichtung der Verstärkungsfaktoren $\boldsymbol{\rho}_i = \mathrm{diag}(\rho_{1,i}, \ldots, \rho_{p,i})$ in Abhängigkeit vom Prämissenvektor $\mathbf{z}$ [8] und durch die Gewichtung des vektorwertigen Ausgangsfehlers:

$$\boldsymbol{\nu}_\delta = \begin{cases} -\sum_{i=1}^{N_r} h_i(\mathbf{z}) \boldsymbol{\rho}_i \frac{\mathbf{P}_2 \mathbf{W} \mathbf{e}_y}{\|\mathbf{P}_2 \mathbf{W} \mathbf{e}_y\| + \delta} & \text{falls } \mathbf{e}_y \neq \mathbf{0} \\ \mathbf{0} & \text{sonst} \end{cases} . \tag{4}$$

wobei $\mathbf{W} = \mathrm{diag}(\, w_1, w_2, \ldots, w_p \,)$ mit $w_i = \frac{1}{|y_i|_{\max}}$, vgl. [4].

Bei der Parametrierung des erweiterten Beobachters müssen weit aus mehr Einstellungen vorgenommen werden als bei der Grundstruktur, vgl. Tabelle 1 mit 2. Das heißt, dass man in Abhängigkeit von der Problemstellung abwägen sollte, ob man mehr Flexibilität beim Entwurf und damit einen höheren Aufwand in Kauf nimmt oder nicht. Eine automatisierte Parametrierung kann hier Abhilfe schaffen.

3 Fallbeispiel: Beobachter-Parametrierung zur Rekonstruktion externer Momente

Am Beispiel einer Problemstellung bei Fahrzeugantrieben wird die bisherige Vorgehensweise bei der Parametrierung von Sliding-Mode Beobachtern erläutert. Ziel ist die möglichst genaue, lastunabhängige Rekonstruktion von Drehschwingungen durch die Messung des Antriebsmoments und der Antriebsdrehzahl. Die Schwierigkeit ist, dass man weder das Lastmoment noch die Drehzahl auf der Abtriebsseite kennt. Zur Verbesserung der Antriebsdynamik und zur Vermeidung der Überlastung des Aktuators soll das unbekannte Lastmoment ebenfalls geschätzt werden.

3.1 Modellierung für den Beobachterentwurf

Die Beschreibung der Drehschwingung in einem Antriebsstrang mit externem Lastmoment M_L basiert auf dem mathematischen Modell aus [9]

$$\dot{\mathbf{x}} = \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 0 & -2D_{p1}\omega_{p1} & -\omega_{p1}^2 \\ 0 & 1 & 0 \end{pmatrix}}_{\mathbf{A}} \mathbf{x} + \underbrace{\begin{pmatrix} \frac{1}{J_0} \\ K_1 \\ 0 \end{pmatrix}}_{\mathbf{B}} u + \underbrace{\begin{pmatrix} -\frac{1}{J_0} \\ 0 \\ 0 \end{pmatrix}}_{\mathbf{D}} \xi \qquad (5)$$
$$y := \omega_R = (1\ 1\ 0)\,\mathbf{x}$$

mit dem Zustandsvektor $\mathbf{x} = (\omega_0\,,\,\Delta\omega_1\,,\,\Delta\varphi_1)^T$, dem Antriebsmoment (Moment des E-Motors) als Systemeingang $u := M_{em}$, dem Lastmoment als unbekannten Eingang $\xi := M_L$ und $y := \omega_R$ als die messbare Winkelgeschwindigkeit des Antriebs. Der Zustandsvektor enthält den schwingungsfreien sogenannten Gleichanteil der Antriebsdrehzahl ω_0, die Drehschwingung $\Delta\omega_1$ und den Torsionswinkel $\Delta\varphi_1$. Die Modellparameter in (5) sind die erste modale Dämpfung D_{p1}, die Eigenkreisfrequenz ω_{p1} der Dreh- bzw. Torsionsschwingung, $J_0 = J_1 + J_2$ als Gesamtmassenträgheit mit J_1 als Massenträgheit des Rotors vom E-Motor mit Getriebe und J_2 als auf die Antriebsseite umgerechnete Massenträgheit des Fahrzeugs sowie $K_1 = 1/J_1$.
Zur aktiven Kompensation der Drehschwingung $\Delta\omega_1$ wird üblicherweise die Differenz aus Antriebs- und Abtriebsdrehzahl zurückgeführt. Da die Abtriebsdrehzahl in diesem Fall nicht messbar ist, vgl. (5), wird $\Delta\omega_1$ aus der Messgröße y und dem Eingang u rekonstruiert. Zur Entkopplung des

Antriebsmoments vom Lastmoment wird hierzu der Sliding-Mode Beobachter (1) mit dem Zustandsvektor $\hat{\mathbf{x}} = (\hat{\omega}_0\,,\, \Delta\hat{\omega}_1\,,\, \Delta\hat{\varphi}_1)^T$ für $N_r = 1$ angesetzt. Zur Verbesserung der Rekonstruktionsgüte soll später noch eine drehzahlabhängige Dämpfung mit $D_{p1} = D_{p1,0} + f(\Delta\omega_1)$ integriert werden, wodurch der Einsatz eines TS Sliding-Mode Beobachters mit $N_r = 2$ motiviert ist.

3.2 Bewertungsfunktion zur Beobachter-Parametrierung

Zur Bewertung der Parametrierung (vgl. Tabelle 1) des Beobachters (1) mit der Grundstruktur (2) wird der maximale stationäre Fehler in Prozent zwischen dem simulierten und geschätzten Lastmoment ermittelt:

$$J_P = \frac{\max|\hat{M}_{L,ss} - M_{L,ss}|}{M_{L,ss}} \cdot 100\% \tag{6}$$

Der unbekannte Eingang aus (5), in diesem Fall das externe Lastmoment M_L, wird rekonstruiert durch

$$\hat{M}_L = \boldsymbol{\mathcal{D}}_2^{+}\nu_\delta\,, \qquad \boldsymbol{\mathcal{D}} = \mathbf{T}\,\mathbf{D} = \left(\mathbf{0}^T\ \boldsymbol{\mathcal{D}}_2^T\right)^T \tag{7}$$

mit $\mathbf{T}$ als Transformationsmatrix zur Überführung von (5) in die kanonische Beobachterform [3].
Die Gütewerte in Abhängigkeit von einer händisch vorgenommen Änderung der Parameter s_p und δ sind in Tabelle 3 dargestellt. Man sieht deutlich die nichtlineare Abhängigkeit zwischen der Parametervariation und Bewertungsfunktion. Die kleinste maximale Abweichung erzielt man in diesem Fall bei einem Beobachterpol von $s_p = -80$ und einer Schalttermverstärkung von $\rho = 25.5$. Diese Art der Parametrierung ist jedoch aufwendig und im erheblichen Masse von der Testfunktion, Erfahrung und dem Wissen des Regelungstechnikers abhängig. Zur Automatisierung der Parametrierung wird im folgenden ein Algorithmus basierend auf dem Simulated-Annealing Verfahren vorgestellt.

4 Simulated-Annealing zur Parametrierung von Takagi-Sugeno Sliding-Mode Beobachtern

Beim Simulated-Annealing Verfahren handelt es sich um einen evolutionären Optimierungs-Algorithmus, der den Abkühlvorgang kristalliner

Strukturen nachbildet, weswegen das Verfahren u.a. durch einen abnehmenden Temperaturverlauf gekennzeichnet ist. Ferner zeichnet sich das Verfahren durch zwei stochastische Merkmale aus, wodurch es möglich ist, lokale Optima auf der Suche nach dem globalen Optimum wieder zu verlassen.

Tabelle 3: Beobachter-Parametrierung zur Lastrekonstruktion

s_p	ρ	δ	J_P
-5	25.0	$5.3 \cdot 10^{-4}$	58.33
-10	25.0	$5.3 \cdot 10^{-4}$	58.33
-20	25.0	$5.3 \cdot 10^{-4}$	58.33
-30	25.0	$5.3 \cdot 10^{-4}$	50.50
-40	25.0	$5.3 \cdot 10^{-4}$	41.67
-50	25.0	$5.3 \cdot 10^{-4}$	25.00
-60	25.0	$5.3 \cdot 10^{-4}$	16.67
-70	25.0	$5.3 \cdot 10^{-4}$	7.50
-80	25.0	$5.3 \cdot 10^{-4}$	1.25
-90	25.0	$5.3 \cdot 10^{-4}$	9.17
-80	25.5	$5.3 \cdot 10^{-4}$	**0.42**
-80	30.0	$5.3 \cdot 10^{-4}$	7.50
-80	20.0	$5.3 \cdot 10^{-4}$	11.67
-80	15.0	$5.3 \cdot 10^{-4}$	24.17

Das Simulated-Annealing Verfahren zur Parametrierung von Takagi-Sugeno Sliding-Mode Beobachtern wird in dem Algorithmus 1 beschrieben: Bei der Initialisierung werden die Lösungskandidaten, die Eingangssignale und Parameter des Simulated-Annealing Verfahrens, die u.a. den Temperaturverlauf beim Ausglühen (Annealing) beeinflussen, festgelegt.

Algorithmus 1.: Simulated-Annealing zur Parametrierung von TS Sliding-Mode Beobachtern

Voraussetzung:

- Festlegung der Lösungskandidaten $\boldsymbol{\theta}$ für die Grundstruktur (2) mit $\boldsymbol{\theta} = \{\rho, \delta, s_1, \ldots, s_p\}$ oder der erweiterten Struktur (4)
- Auswahl der geeigneten Eingangssignale Z_i, $i = 1, \ldots, N_d$ für den gesamten Arbeitsbereich $\boldsymbol{\theta} \in \Omega$, $i = 1$
- Festlegung von $\Delta J_{P_{\max}}$ als apriori-Schätzung der maximalen Differenz der Bewertungsfunktion
- Festlegung des Temperaturverlaufs beim Ausglühen (Annealing) mit $T(k) = T_0 e^{-k/\tau_T}$ wobei T_0 die Anfangstemperatur und τ_T die Abklingrate beschreibt
- Festlegung des zulässigen euklidischen Abstands δ_θ bei der Variation der Parametersätze $\boldsymbol{\theta}_k$
- Berechnung der Transformationsmatrix $\mathbf{T}$ und ermittle damit $\boldsymbol{\mathcal{D}}_2$ zur Rekonstruktion des unbekannten Eingänge [3]

Simulated-Annealing

Wahl eines zufälligen Lösungskandidaten $\boldsymbol{\theta}_k \in \Omega, k = 0$

Wähle einen Parametersatz $\boldsymbol{\theta}_- \in \Omega$ (apriori-Schätzung) in der Nähe von $\boldsymbol{\theta}_k$ durch eine zufällige Variation: $\boldsymbol{\theta}_- = \boldsymbol{\theta}_k + \boldsymbol{\Delta_\theta}$ mit $\|\boldsymbol{\Delta_\theta}\| < \delta_\theta$

Berechne $\mathbf{P}_2$ mit (3) und führe die Simulation bzw. das Experiment mit dem TS SMO nach (1), (2) bzw. (4) durch

Auswahl des neuen Lösungskandidaten für die Suche nach einem globalen Minimum mit

$$\boldsymbol{\theta}_{k+1} = \begin{cases} \boldsymbol{\theta}_- & \text{falls} \quad J_P(\boldsymbol{\theta}_-) \geq J_P(\boldsymbol{\theta}_k) \\ \boldsymbol{\theta}_- & \text{falls} \quad J_P(\boldsymbol{\theta}_-) < J_P(\boldsymbol{\theta}_k) \text{ mit der} \\ & \quad \text{Annahmewahrscheinlichkeit } P = e^{-\frac{\Delta J_P(\boldsymbol{\theta}_k)}{\Delta J_{P_{\max}} \, T(k)}} \\ \boldsymbol{\theta}_k & \text{falls} \quad J_P(\boldsymbol{\theta}_-) < J_P(\boldsymbol{\theta}_k) \text{ mit der} \\ & \quad \text{Annahmewahrscheinlichkeit } 1 - P \end{cases}$$

mit $\Delta J_{P_k}(\boldsymbol{\theta}_k)) = J_P(\boldsymbol{\theta}_k) - J_P(\boldsymbol{\theta}_-)$.

k := k + 1

Wiederhole 3. - 7. bis eine zuvor festgelegte Abbruchbedingung (z.B. eine Untergrenze für die Abkühlung) erreicht ist

Speichern von $\hat{\boldsymbol{\theta}}_{Z_i} := \boldsymbol{\theta}_{k-1}$

Wiederhole 2. - 9. bis alle Eingangssignale (d.h. $i = N_d$) abgearbeitet sind, $i := i + 1$

Der eigentliche Algorithmus basiert auf zwei Zufallsprozessen. Im Ersten wird der Lösungskandidat in der Nähe des aktuellen Parametervektors $\boldsymbol{\theta}_k$ zufällig ausgewählt. Der zweite Zufallsprozess wird genutzt, um eine schlechtere Lösung als die aktuelle Lösung mit einer Wahrscheinlichkeit

$$P = e^{-\frac{\Delta J_P(\boldsymbol{\theta}_k)}{\Delta J_{P\max}\, T(k)}} \tag{8}$$

anzunehmen mit

$$T(k) = T_0\, e^{-k/\tau_T} \tag{9}$$

als dem Temperaturverlauf beim Ausglühen.

5 Zusammenfassung

Vorgestellt wurde ein Simulated-Annealing Verfahren zur automatisierten Parametrierung von Takagi-Sugeno Sliding-Mode Beobachtern. Der Einsatz dieses Verfahrens wurde motiviert durch eine Problemstellung aus der Antriebstechnik. Hierbei werden nichtmessbare Drehschwingungen und externe Lastmomente aus dem Antriebsmoment und der -drehzahl rekonstruiert.
In aktuellen und zukünftigen Arbeiten wird das Verfahren erweitert um die Berücksichtigung des Vorwissens eines eingeschränkten Parameterraums und soll ebenfalls angewandt werden auf die Parametrierung von Proportional-Multi-Integral Beobachtern in Takagi-Sugeno Fuzzy Form [10] zur Rekonstruktion von Sensor- und Aktuatorfehlern u.a. bei Windenergieanlagen.

Literatur

[1] E. Gauterin, F. Pöschke und H. Schulte. „Nonlinear Quadratic Estimator with Error State Weighting“. In: *IEEE International Conference on Evolving and Adaptive Intelligent Systems - EAIS*, Ljubljana, Slovenia, 2017.

[2] P. Bergsten, R. Palm und D. Driankow. „Observers for Takagi-Sugeno Fuzzy Systems“, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 32, no. 1, S. 114–121, 2002.

[3] P. Gerland, D. Groß, H. Schulte und A. Kroll. „Design of Sliding Mode Observers for TS Fuzzy Systems with Application to Disturbance and Actuator Fault Estimation“. In: *IEEE Conference on Decision and Control*, Atlanta, USA, S. 4373–4378, 2010.

[4] S. Georg und H. Schulte. „Takagi-Sugeno Sliding Mode Observer with a Weighted Switching Action and Application to Fault Diagnosis for Wind Turbines“. In: *Intelligent Systems in Technical and Medical Diagnostics*, Springer, Berlin, 2013, ISBN 978-3-642-39880-3.

[5] A. Khachaturyan, S. Semenovskaya und B. Vainshtein. „Statistical-Thermodynamic Approach to Determination of Structure Amplitude Phases“ In: *Sov. Phys. Crystallography.* 24 (5), 1979.

[6] Z. Lendek, T. M. Guerra, R. Babuska und B. De Schutter. „Stability Analysis and Nonlinear Observer Design using Takagi-Sugeno Fuzzy Models“, Springer-Verlag Berlin Heidelberg, 2011.

[7] C. Edwards und S. K. Spurgeon. *Sliding Mode Control: Theory and Applications.* Taylor & Francis, Boca Raton, 1998.

[8] F. Pöschke, S. Georg und H. Schulte. „Fault Reconstruction using a Takagi-Sugeno Sliding Mode Observer for the Wind Turbine Benchmark“. In *10th International Conference on Control (UKACC 2014)*, Loughborough University, UK, 2014.

[9] G. Götting. „Dynamische Antriebsregelung von Elektrostraßenfahrzeugen unter Berücksichtigung eines schwingungsfähigen Antriebsstrangs“. Dissertation, In: *Aachener Beiträge des ISEA*, Aachen: Shaker 2004.

[10] H. Schulte und F. Pöschke. „Entwurf von Proportional-Multi-Integral Beobachtern in Takagi-Sugeno Fuzzy Form zur Schätzung unbekannter Eingänge“. In: *at - Automatisierungstechnik.* 65 (3), März 2017.

Automatic Machine Learning: Hierarchical Planning Versus Evolutionary Optimization

Marcel Wever[1], Felix Mohr[2], Eyke Hüllermeier[3]

Paderborn University
Warburger Str. 100, 33098 Paderborn

[1]E-Mail: marcel.wever@uni-paderborn.de
[2]E-Mail: fmohr@uni-paderborn.de
[3]E-Mail: eyke@upb.de

Abstract

These days, there is a growing need for machine learning applications, coming with the quest to automate parts of the process of engineering machine learning tools and algorithms. This development has triggered the emergence of automated machine learning (AutoML) as a new subfield of machine learning. In AutoML, the selection, composition and parametrization of machine learning algorithms is automated and tailored to a specific problem, resulting in a machine learning pipeline. Current approaches reduce the AutoML problem to optimization of hyperparameters. Based on recursive task networks, in this paper we present one approach from the field of automated planning and one evolutionary optimization approach. Instead of simply parametrizing a given pipeline, this also includes the structure optimization of machine learning pipelines. We evaluate the two approaches in an extensive evaluation, finding both of them to have their strengths in different areas. Moreover, the two approaches outperform the state-of-the-art tool Auto-WEKA in many settings.

1 Introduction

While the demand for machine learning functionality is growing quite rapidly these days, end users in application domains are normally not machine learning experts. Therefore, there is an urgent need for suitable support in terms of tools that are easy to use. Ideally, the induction of

models from data, including the data preprocessing, the choice of a model class, the training and evaluation of a predictor, the representation and interpretation of results, etc., could be automated to a large extent [7]. This has triggered the field of automated machine learning (AutoML), which has developed into an important branch of machine learning research in the last couple of years. Given a specific problem (data set), the goal of AutoML is to automatically set up a suitable machine learning pipeline, comprising the aforementioned steps of model induction.

In spite of quite impressive first results, state-of-the-art tools such as Auto-WEKA [13] and Auto-sklearn [4] are still limited in scope and restricted to rather simple learning problems such as classification, essentially because automated machine learning is reduced to the optimization of hyperparameters. In this paper, we address the additional problem of optimizing the structure of a machine learning pipeline, instead of simply parametrizing a given one. We consider structure optimization as an important prerequisite for the application of automated machine learning to learning problems more complex than standard (binary) classification or regression, such as multi-target prediction or structured-output prediction. More specifically, we compare two approaches for optimizing machine learning pipelines, which are based on two different principles for searching the space of configurations.

Our first approach is based on the idea to consider the machine learning problem as a planning task. This idea is arguably quite natural: what the machine learning expert has to do is to devise a plan determining which data processing steps are to be executed in which order, and how the different steps shall be configured or parametrized. More specifically, we make use of recursive task networks for hierarchical planning [9], which offers a versatile approach for the configuration of machine learning pipelines.

As an alternative to the systematic search strategy realized by hierarchical planning, we make use of an evolutionary approach that is based on recursive task networks as well. To this end, the evolutionary algorithm maintains a population of individuals which represent a single path in the network and thus a machine learning pipeline including hyperparameters being set. We apply multi-objective optimization to assess the fitness of ML pipelines considering the generalization behavior of a pipeline in different stages of the learning process, i.e. for different proportions of training-validation splits, in order to prevent the optimized machine learning pipelines from overfitting the provided data.

2 An HTN-Based Search Graph

2.1 Hierarchical Task Networks

A hierarchical task network (HTN) is a partially ordered set T of tasks. A task $t(v_0, .., v_n)$ is a name with a list of parameters, which are variables or constants from $\mathcal{L}$. For example, $configureC45(c)$ could be the task of creating a set of options for a decision tree and assigning them to the concrete decision tree object c. A task named by an operator (e.g., $setC45Options(c, o)$) is called *primitive*, otherwise it is *complex*. A task whose parameters are constants is ground.

We are interested in deriving a plan from a task network. Intuitively, we can refine (and ground) complex tasks iteratively until we reach a task network that has only ground primitive tasks, i.e., a set of partially ordered actions. While primitive tasks can be realized canonically by a single operation, complex tasks need to be decomposed by *methods*. A method $m = \langle name_m, task_m, pre_m, T_m \rangle$ consists of its name, the (non-primitive) task $task_m$ it refines, a logic precondition $pre_m \in \mathcal{L}$, and a task network T_m that realizes the decomposition. Replacing complex tasks by the network of the methods we use to decompose them, we iteratively *derive* new task networks until we obtain one with ground primitive tasks (actions) only.

To get an intuition of this idea, consider the (totally ordered) task networks in the boxes of Figure 1 as an example. The colored entries are the tasks of the respective networks. Orange tasks are complex (need refinement), and green ones are primitive. The tree shows an excerpt of the possible refinements for each task network. The idea is very similar to derivations in context free grammars where primitive tasks are terminals and complex tasks are non-terminal symbols. The main difference is that HTN considers the concept of a state, which is modified by the primitive tasks and poses additional constraints on the possible refinements.

The definition of a simple task network planning problem is then straight forward. Given an initial state s_0 and a task network T_0, the planning problem is to derive a plan from T_0 that is applicable in s_0. A simple task network planning problem is then a tuple $\langle s_0, T_0, O, M \rangle$, where O and M are finite sets of operators and methods, respectively.

The HTN problem definition induces a search graph (a tree) that can be searched with standard search algorithms such as depth first search,

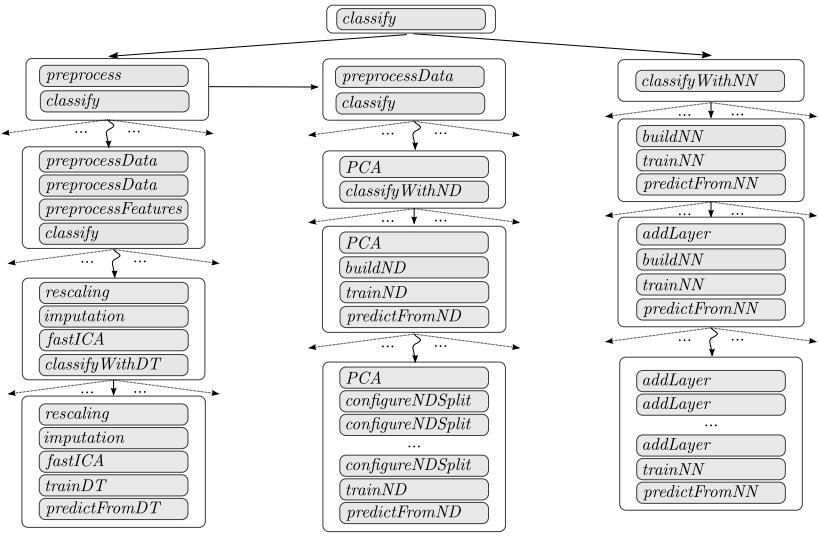


Figure 1: Task networks allow not only more flexible pipelines but even to configure its elements.

best first search, etc.. The graph in Figure 1 sketches (an exerpt of) such a search graph for the AutoML problem. Every node corresponds to a partially defined pipeline (complex tasks are partially defined aspects that still require refinement). The root node corresponds to the pipeline with the initial complex task, and goal nodes are nodes that have fully defined pipelines. Usually, there is a one-one correspondence between search space elements, e.g., the machine learning pipelines, and the goal nodes.

A typical translation of the HTN problem into a graph is to select the *first* complex task in the network of a node and to define one successor for each ground method that can be used to resolve the task. This technique is called forward-decomposition. While it is also possible to derive graphs with other structures for the same HTN problem, we adopt forward-decomposition in this paper.

2.2 The AutoWEKA-Simulation Search Graph

The search graph is induced by the description of a task network problem. As described above, we adopt forward-decomposition, i.e., a node has

successors iff it has at least one complex task, and there is one successor for each method or operator that can be used to refine the *first* task in the network. The complete problem description is very technical, so we focus on giving an intuition. The full formal specification is available with our implementation[2].

The root node is a task network consisting of three tasks `createRawPP`, `setupClassifier`, `refinePP`. The first task can be refined to the existing preprocessing algorithms *without* parametrizing them. There is one method for each classifier to refine the second task `setupClassifier`. Each of these methods refines the task to a network of the form `setupParam1`, ..., `setupParamN` for all of the N parameters of the respective classification algorithm, so the network enforces that a decision is made for each of the parameters. For each of the parameters, there are methods that induce primitive tasks either setting or not setting the respective parameter, i.e., leaving it at the default value. The same technique is then applied to refine `refinePP` and thereby to configure the initially chosen preprocessor (if any). We support the same preprocessors, classifiers, and parameters as used in AutoWEKA.

This modeling technique induces a tree with two regions, which are separated in a relatively low depth of the search tree, say d. This is because the whole construction process encodes the idea of (i) selecting the preprocessor, (ii) selecting the classifier, (iii) parametrizing the classifier, (iv) parametrizing the preprocessor. Partial solutions of nodes in the shallower region (up to a depth of d) define the algorithms that will be used but have not yet made any decision about parametrization, i.e., the steps (i) and (ii). All nodes in depth of at least d deal with the parametrization, i.e. correspond to decisions either in phase (iii) or (iv).

Numerical parameters are discretized either on a linear scale or a log scale. The discretization technique for a parameter is not a choice point but is fixed in advance.

2.3 Potential of HTN Planning for AutoML

Given one concrete example of how to create ML pipelines using HTN planning, we stress that we are not committed to one particular HTN problem definition. In fact, there are many different HTN problems that can cover exactly the same search space. So apart from any questions

[2]Attached as supplementary material during review phase.

related to heuristics, node evaluation, etc., the mere way of how the HTN problem is *formulated* can have a tremendous impact on the search efficiency. Just to give an example, we could use a two-step network where we first choose and configure the preprocessor (or choose to not use any) and then choose and configure the classifier. Yet, we may also interleave these configurations and first choose the preprocessor and the classifier, and *then* configure both of them. While this looks like a trivial change that does not affect the set of constructible pipelines, it has important consequences on the structure of the search tree.

The dominating expressiveness of HTN techniques for AutoML is not only reflected in the ability to construct pipelines of arbitrary lengths but also in that they can *configure complex elements* within it. For example, the right branch of Figure 1 shows that we can configure a neural network. Note, once again, that even though the figure does not show any parameters, we cannot only control the number of layers but also their connections.

Another important advantage of HTN for AutoML is its ability to encode *reduction*. Reduction or decomposition techniques such as one-versus-rest [11], all-pairs [5], or error correcting output codes [3] are quite popular in machine learning. For example, instead of solving the multi-class classification problem directly on the set of k classes, we can first separate two *sets* of classes from each other. This induces two reduced classification problems, which can be solved either directly or again by recursing in the same way (unless there are only two classes left), and this is a very natural use case of HTN planning.

3 ML-Plan: AutoML through HTN Planning

ML-Plan directly solves the HTN problem defined above using an HTN planner. For a specific problem type, e.g., classification, the HTN problem that needs to be solved is usually fixed. The variety for different queries arises from the fact that a plan (composition) performs differently on different data sets. The data set is then used to *guide* the search.

We adopt a best first search algorithm in order to identify good pipelines. A best first search algorithm assigns a number to each node and always chooses the node with the currently lowest known value for expansion.

Since the prediction error as the solution quality does not decompose over the path (necessary for A*), we adopt a randomized depth first search similar to the one applied in Monte Carlo Tree Search to inform the search procedure. Given the node for which we need a score, we choose a random path to a goal node. This is achieved by randomly choosing a child node of the node itself, then randomly choosing a child node of the child node, etc. until a goal node is reached. We then compute the solution "qualities" of each of n such random completions and take the minimum as an estimate for the best possible solution that can be found under that node.

The qualities of the completed solutions are determined by computing a k-step Monte Carlo Cross Validation. That is, a fixed portion of the data initially provided to the search algorithm is allocated for node evaluation; in our implementation we used 70% of the data. To evaluate a single solution, this portion is then partitioned k-times into a stratified training and validation set; here, we also chose a split of 70% for training and 30% for validation and $k = 5$. For each of the k splits, the solution pipeline is trained with the respective training set and tested against the validation set. The mean 0/1-loss of this evaluation is the score of that solution.

Since the node evaluation function actually computes solutions, we propagate these solutions to the search algorithm. More precisely, we propagate the best of the n solutions drawn for each node to the search routine. This way, we can always return solutions even if the main search routine did not already discover any goal node.

In order to make this strategy more reliable, in the upper region of the search graph, we use a biased breadth first search instead. This is to avoid that the randomized depth search averages over too many heavily distinct solutions. The bias within a layer is towards the nodes corresponding to frequently well-performing algorithms: KNN, random forests, voted perceptron, SVM, logistic regression (in this order). This arbitrary preference is hard-coded, but we plan to make it data-dependent in a follow-up version.

Since evaluating the solution candidates is very costly, we use a reduced version of the originally given dataset. For a number n of classes, we reduce the number of examples to at most $250 \times n$ (stratified removal) and the number of features to at most $5 \times n$ using principal component analysis. Of course, this reduction only needs to be computed once in a preprocessing step of the overall search process. In [10], we do not make this simplification.

4 EvoML-Plan: Genetic HTN Planning

As an alternative to the systematic search strategy realized by hierarchical planning, we make use of a so-called messy genetic algorithm [6]. In contrast to classical genetic approaches that take a fixed length of the genotype for granted, messy genetic algorithms allow for variable length strings of genes. The variable length of genestrings is crucial for solving HTN planning problems as the length of plans may vary substantially.

4.1 Genetic Representation

Each individual represents one possible plan derived from the hierarchical task network (HTN). In order to derive plans from the HTN, complex tasks need to be refined to primitive tasks until finally a concrete plan is obtained. As for a refinement different choices might be eligible, we can define our genetic representation to encode these choices. Since the length of concrete plans may have an arbitrary length and moreover may vary for different plans, the number of choices to be made are varying accordingly. Moreover, the genetic representation must not be fixed to a certain length since otherwise it could not represent each possible solution.

Therefore, we choose the genetic representation to be a list of non-negative integers. For each refinement, the possible refinement options are listed and the next gene is taken as the index in this list. If the value of this gene exceeds the number of possible options, genes will get skipped until we finally reach a gene in this range. Note that the genetic representation does not take any semantics into account. Therefore, the i-th gene may represent for instance the choice of a parameter value or a particular algorithm.

Due to the variable length of chromosomes, an individual might be over- or underspecified. While we can deal with over-specification by simply ignoring the remaining part of the chromosome, the other scenario requires additional genetic material to be added dynamically. To this end, we extend the chromosome by randomly drawing new genes until the plan is entirely specified.

Another problem using this genetic representation arises from the application of genetic operators such as mutation and crossover. As a small change in a single gene may lead to dramatic changes since all the choices

made subsequently lose their previous semantics, we divide the overall genotype into so-called *codons*. A codon represents a cohesive sequence of genes that is inherited by offspring en bloc. More specifically, for each complex task, we use a distinct codon of variable length.

Considering our scenario of configuring ML pipelines, the chromosome is divided into codons describing the feature preprocessor, the classifier, the classifier's parametrization, and the parameters chosen for the feature preprocessing. An example is illustrated in Figure 2. Due to this segmentation, changing a gene for the choice of the feature preprocessor does no longer influence the decisions for the classifier and its parametrization at all. Still the codon for the parametrization of the feature preprocessor adopt its semantics according to the feature preprocessor.

The segmentation into codons allows a variable length of the chromosome while isolating semantic structures in order to facilitate the exchange of partial solutions without changing other components. In particular, with this technique it is possible to swap the feature preprocessors of two individuals only.

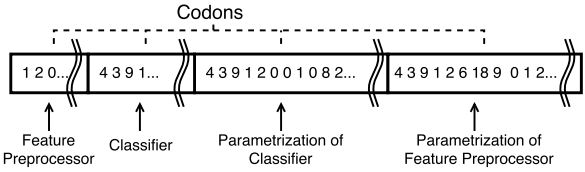


Figure 2: Genotype for the representation of ML pipelines

4.2 Genetic Operators

For the variation of individuals, on one hand, we use standard mutation for single genes. On the other hand, we use an instantiation of n-point crossover fitted to our genetic representation. Meaning, the crossover takes the semantics of the ML pipelines into account, exchanging only entire building blocks of the pipeline. An example for the genetic operator recombining two individuals is presented in Figure 3. In the example figure the crossover is a two-point crossover. Generally, the crossover operator is not fixed to two intersection points. Depending on the order of the codons and depending on how many building blocks a ML pipeline may involve, the crossover can be adapted to exchange the entire feature

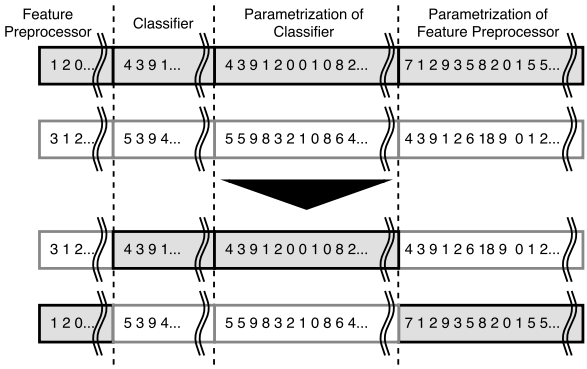


Figure 3: Crossover of two ML pipeline individuals

preprocessing or one to multiple codons. The only important criterion for the crossover is to exchange codons for the selection of an algorithm and its parametrization together.

4.3 Multi-Objective Optimization

Optimizing for the best or most accurate ML pipeline comes also with a high danger for overfitting the data. Therefore, we choose a multi-objective optimization approach in order to have a more differentiated look at the generalization behavior of each evaluated ML pipeline. To this end, we use repeated random sub-sampling validation, aka. Monte Carlo cross-validation, creating repeatedly stratified splits at random [1, 12]. Since the data contained in the training and validation set varies with each split and evaluation, we prevent the ML pipelines from getting to specialized for a certain partitioning.

More specifically, we evaluate each individual performing 5-fold Monte Carlo cross-validation for 50:50, 67:33, and 80:20 splits of training and validation data. Taking the average error rate for the different split proportions as an individual fitness function, we use these as objectives in the multi-objective evolutionary algorithm NSGA-II (see [2]) in general. However, the evaluation of an individual is rather costly, and an individual's evaluation might take several minutes. Due to this and since the main reason for multiple objectives is to prevent the individuals from overfitting, we only use the error rate for the 80:20 splits as a fitness function if a timeout of less the 5 minutes is given.

Selection is done via tournament selection involving 2 individuals, where these two are compared primarily whether the fitness values of one individual dominate the other one's fitness values. If this is not the case for one of the two individuals, a crowding distance comparator is taken into account. The crowding distance comparator calculates how close other individuals are to the considered individuals, and it prefers individuals that are more different from the remaining population.

In order to support diversification of the population, we reinitialize the population every 5 generations with random individuals, keeping only the elite of the current population.

4.4 Solution Selection

The result of NSGA-II is represented by a Pareto set of non-dominated individuals. Therefore, we still need to select a single individual as the final result of the genetic algorithm. As usual in multi-objective optimization, the selection of a particular candidate from a Pareto set is a non-trivial decision. We proposed three fitness functions evaluating an ML pipeline on different splits of the given dataset. Interpreting the three fitness values as a vector in the three-dimensional space, we choose the solution which is closest to the optimum, i.e. the solution with the smallest distance to the origin.

Moreover, this rather simple selection method allows us to always maintain the best solution seen so far. Hence, in the face of timeouts the algorithm is always able to immediately return a solution even if the algorithm is currently processing a generation. However, as a prerequisite for returning a solution, a valid individual, i.e., an ML pipeline that can be applied to the particular problem, must have been already evaluated.

5 Evaluation

We evaluate the two approaches on a selection of 21 datasets from the UCI repository. In our evaluation we compare the two proposed approaches EvoML-Plan and ML-Plan to each other and additionally to the state-of-the-art tool Auto-WEKA.

5.1 Experimental Setup

For the comparison of the three approaches, we carried out 25 runs for each of the approaches on the 21 datasets for timeouts of one minute and one hour, yielding a total number of 3,150 experiments. The timeout for the evaluation of a single individual in EvoML-Plan resp. the internal evaluation of a single solution in ML-Plan was set to 10s for a timeout of one minute. In the case of a timeout of one hour, the internal evaluation timeout was set to 5 minutes. For all the runs, 70% of a stratified split of the entire data were provided to the algorithms as training data and 30% were used for testing. The computations were executed on 200 Linux machines in parallel with 8 cores (Intel Xeon E5-2670, 2.6GHz) and 32GB memory each.

Runs exceeding the timeout limit or the resource limitations of the nodes were canceled and their results are disregarded in the following discussion. Nevertheless, the algorithms were admitted an extra amount of time so that they were killed after taking 110% of the set timeout. Moreover, the algorithms were killed when consuming more resources (CPU and memory) than allocated, which happens because in the implementation controlling the CPU and memory consumption of forked subprocesses is rather hard.

For significance testing, we use the Mann-Whitney-U Test [8], and we denote significant improvements respectively degradation if $p < 0.05$.

5.2 Results

In Tables 1 and 2 the results of the runs with a timeout of one minute respectively one hour are presented. In the tables, for each dataset and each approach, the number of returned solutions (n) and the average test set loss (0/1-loss) plus or minus the standard deviation is shown. Furthermore, for each row the best performing approach is highlighted with bold letters and non-significant degradations are highlighted underlining these values.

The results in Table 1 show that EvoML-Plan as well as ML-Plan clearly outperform Auto-WEKA for all the evaluated datasets. As Auto-WEKA does not even return a solution in the given timeout for some datasets, we can conclude that both proposed approaches find solutions at least faster than Auto-WEKA does. Furthermore, in settings where Auto-WEKA returned solutions, our approaches return solutions that are significantly

Table 1: Experimental results (mean 0/1-losses±std) for a timeout of 1 minute

Dataset	n	EvoML-Plan	n	ML-Plan	n	Auto-WEKA
abalone	23	73.94 ± 0.59	19	**73.33** ± 0.60	12	74.94 ± 1.25
amazon	14	**90.41** ± 5.28	0	?	0	?
car	24	5.05 ± 1.56	19	**3.40** ± 0.74	12	7.18 ± 1.00
cifar10	4	86.34 ± 6.33	16	**68.82** ± 6.15	0	?
cifar10small	16	84.81 ± 7.89	20	**69.98** ± 6.71	0	?
convex	12	47.65 ± 0.34	18	**27.71** ± 0.19	0	?
dexter	20	**24.88** ± 9.90	0	?	0	?
dorothea	17	**26.81** ± 10.32	0	?	0	?
germancredit	24	27.52 ± 1.46	20	**24.83** ± 0.83	12	28.64 ± 0.72
gisette	13	35.65 ± 15.93	17	**3.65** ± 0.15	0	?
kddcup09appe	6	**1.77** ± 0.00	3	2.47 ± 0.92	0	?
krvskp	25	**1.15** ± 0.44	20	5.09 ± 1.76	12	1.60 ± 1.64
madelon	25	**27.74** ± 2.47	18	39.83 ± 2.12	12	30.35 ± 2.49
mnist	6	35.88 ± 11.67	20	**6.50** ± 0.23	0	?
mnistrotatio	11	88.76 ± 0.00	9	**74.88** ± 2.93	0	?
secom	25	**6.42** ± 0.00	19	6.44 ± 0.06	12	6.57 ± 0.27
semeion	24	12.69 ± 2.53	20	**10.49** ± 0.86	12	14.0 ± 0.79
shuttle	24	0.11 ± 0.06	15	**0.02** ± 0.01	12	0.14 ± 0.01
waveform	25	13.92 ± 1.13	16	**13.71** ± 0.27	12	15.14 ± 1.54
winequality	25	38.86 ± 2.89	20	**32.43** ± 0.60	12	36.56 ± 0.30
yeast	22	40.29 ± 1.31	19	**40.14** ± 1.32	12	42.17 ± 1.77

better than Auto-WEKA's solutions in 7 out of 11 cases. Comparing EvoML-Plan and ML-Plan, we can observe that best performances alternate for the different datasets. While EvoML-Plan leads to significantly better performing results in 5 datasets and ML-Plan in 7 datasets, for the remaining the performances are competitive to each other.

After a timeout of one hour (see Table 2), the two proposed approaches still perform better than Auto-WEKA for many datasets. However, the clear dominance is diminishing but still Auto-WEKA yields better results only in 5 out of 21 cases, where a significant improvement over both EvoML-Plan and ML-Plan is achieved only once. Considering the performance of our two approaches, we notice alternating significant improvements. While ML-Plan yields superior results in 8 cases, EvoML-Plan returns significantly better solutions for 6 datasets. The remaining 7 datasets indicate competitiveness of the two approaches.

Table 2: Experimental results (mean 0/1-losses±std) for a timeout of 1 hour

Dataset	n	EvoML-Plan	n	ML-Plan	n	Auto-WEKA
abalone	22	**72.93** ± 0.61	10	73.12 ± 0.13	25	73.49 ± 0.76
amazon	23	51.31 ± 10.84	18	**31.15** ± 1.32	24	51.44 ± 1.61
car	23	2.59 ± 1.12	4	0.83 ± 0.38	24	**0.65** ± 0.23
cifar10	20	77.04 ± 7.03	2	**60.73** ± 1.80	0	?
cifar10small	21	72.19 ± 6.92	4	**60.11** ± 0.77	1	70.23 ± 0.00
convex	24	44.28 ± 5.10	12	**27.7** ± 0.22	25	46.86 ± 0.25
dexter	23	**9.81** ± 2.43	4	19.1 ± 2.32	11	11.24 ± 0.51
dorothea	24	**9.58** ± 2.54	0	?	0	?
germancredit	22	25.5 ± 1.09	9	**24.94** ± 0.42	25	26.81 ± 1.04
gisette	23	4.21 ± 1.49	5	5.00 ± 1.84	23	**3.93** ± 0.29
kddcup09appe	19	1.77 ± 0.00	3	1.78 ± 0.01	17	**1.77** ± 0.00
krvskp	25	**0.73** ± 0.24	10	1.73 ± 0.44	24	2.27 ± 2.28
madelon	25	26.65 ± 2.99	9	39.89 ± 0.44	24	**26.11** ± 2.31
mnist	17	12.59 ± 1.40	5	**7.21** ± 1.70	25	7.21 ± 0.12
mnistrotatio	22	73.27 ± 6.65	3	**62.95** ± 0.38	25	78.61 ± 0.27
secom	25	**6.42** ± 0.00	8	6.53 ± 0.11	25	6.52 ± 0.25
semeion	19	**7.44** ± 0.87	13	10.0 ± 0.48	18	13.01 ± 2.02
shuttle	24	0.06 ± 0.05	5	**0.05** ± 0.05	25	0.12 ± 0.04
waveform	24	**13.24** ± 0.57	11	14.51 ± 0.50	25	13.26 ± 0.40
winequality	22	35.66 ± 2.27	15	**32.89** ± 0.69	22	33.34 ± 1.11
yeast	21	40.26 ± 1.59	11	40.23 ± 0.68	23	**39.87** ± 1.36

To sum up, we notice that the two approaches already lead to promising results comparing to the state-of-the-art Auto-WEKA. Especially, in a timeout of one minute, it becomes clear that the two proposed approaches perform faster. Even for a timeout of one hour, EvoML-Plan] and ML-Plan continue to perform better than Auto-WEKA. For both timeouts and all the datasets, we observe competitiveness for the two approaches and find that both have their individual strengths, albeit we notice ML-Plan to have a slight edge over EvoML-Plan.

6 Conclusion and Future Work

We proposed two new approaches to the AutoML problem, both being based on hierarchical planning. While on one hand, we made use of a classical planning approach, as an alternative search strategy, we applied a multi-objective evolutionary algorithm to the same problem. In our evaluation, we found significant performance improvements over the state-of-the-art. Moreover, we have seen that both strategies have their advantages becoming evident in better performance compared to the other approach.

In our evaluation, we limited the configured machine learning pipelines to a length of two, i.e. incorporating a feature preprocessor and a classifier, in order to remain comparable to Auto-WEKA. However, by this limitation we did not even leverage the full potential of HTN, and thus, exploiting the latter is an important point for future work. Moreover, there were datasets where worse results have been obtained in the run with a timeout of one hour compared to the ones returned after one minute. This indicates that the solutions returned after one hour tend to overfit the data, requiring a mechanism to deal with this problem.

Acknowledgment

This work is part of the Collaborative Research Centre "On-the-Fly Computing" at Paderborn University, which is supported by the German Research Foundation (DFG).

References

[1] P. Burman. A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. *Biometrika*, 76(3), 1989.

[2] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolutionary Computation*, 6(2):182–197, 2002.

[3] T. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[4] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2962–2970. Curran Associates, Inc., 2015.

[5] J. Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

[6] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5), 1989.

[7] J. R. Lloyd, D. K. Duvenaud, R. B. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Automatic construction and natural-language description of nonparametric regression models. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 1242–1250, 2014.

[8] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Statist.*, 18(1):50–60, 03 1947.

[9] F. Mohr, T. Lettmann, and E. Hüllermeier. ITN planning: Planning with independent task networks. In *Proceedings KI-2017, 40th German Conference on Artificial Intelligence, Dortmund, Germany*, 2017.

[10] F. Mohr, M. Wever, and E. Hüllermeier. ML-Plan: Automated machine learning via hierarchical planning. Submitted for publication, 2018.

[11] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[12] J. Shao. Linear model selection by cross-validation. *Journal of the American Statistical Association*, 88(422):486–494, 1993.

[13] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-weka: Automated selection and hyper-parameter optimization of classification algorithms. *CoRR*, abs/1208.3719, 2012.

Untersuchungen zur Verwendung von Takagi-Sugeno Fuzzy Systemen in modellprädiktiven Dual-Mode-Reglern

Alessio Cavaterra, Steven Lambeck

FB Elektrotechnik und Informationstechnik, Hochschule Fulda
Leipziger Str. 123, 36037 Fulda
E-Mail: {alessio.cavaterra,steven.lambeck}@et.hs-fulda.de

Einleitung

Modellprädiktive Regler (MPC) eignen sich insbesondere für träge Prozesse mit großen Abtastzeiten, weil in diesen Anwendungen ausreichend Rechenzeit zwischen zwei Abtastzeitpunkten zur Verfügung steht. Im Rahmen des F-&-E-Projektes „Dezent – Dezentrale Klimageräte“[3] eignen sich MPC-Strategien für den Einsatz in Luftbe- und -entfeuchtungsprozessen mit klimatischen Beschränkungen zur Erhaltung von Kulturgütern. Der MPC-Algorithmus muss einfach auf einer Hardware zu implementieren und die Stabilität des MPC einfach nachzuweisen sein. Ein möglicher Ansatz zur Erfüllung der genannten Anforderungen ist der sogenannte Dual-Mode MPC [1].
Die Erweiterung des Dual-Mode MPC-Ansatzes um Takagi-Sugeno (TS) Fuzzy Systeme erfordert eine Reihe von Voruntersuchungen. Dies betrifft zum Einen die Implementierung der TS Fuzzy Systeme im Regelgesetz (s. Abschnitt 1). Zum Anderen ist die Approximation der maximal zulässigen Menge (MAS, engl.: „maximal admissable set“) aller Anfangszustände der TS Fuzzy Modelle notwendig (s. Abschnitt 2). Der Beitrag beschränkt sich auf TS Fuzzy Systeme, die mit mehreren LQ-Zustandsrückführungen stabilisiert werden. Die Untersuchungen setzen zeitdiskrete TS Fuzzy Systeme voraus, deren Ruhelage im Ursprung liegt. Die Abtastzeit ist T_A. Ein Abtastschritt wird abgekürzt mit $k = kT_A$. Die in diesem Beitrag betrachteten TS Fuzzy Systeme werden formuliert mit den Gleichungen

[3] `https://www.hs-fulda.de/elektrotechnik-und-informationstechnik/forschung/dezent/`

$$x_{k+1} = \sum_{i=1}^{N} A_i h_i(z) x_k + \sum_{i=1}^{N} B_i h_i(z) u_k \tag{1}$$

$$y_k = \sum_{i=1}^{N} C_i h_i(z) x_k \tag{2}$$

mit $C_i = E, \forall i$, wobei E die Einheitsmatrix darstellt. Schedulingvektor z enthält eine oder mehrere Zustandsgrößen $x_{i,k}, i = 1 \dots n_x$ und für die Fuzzy-Basisfunktionen $h_i(z) = h_i$ gelten $\sum_{i=1}^{N} h_i(z) = 1, 0 \leq h_i(z) \leq 1$. Für N Teilmodelle werden entsprechend N LQ-Zustandsrückführungen entworfen. Diese werden nach dem „Parallel Distributed Compensator"-Prinzip (PDC) miteinander verknüpft [2]. Damit ergibt sich das Regelgesetz

$$u_k = -\sum_{i=1}^{N} K_i h_i(z) x_k \,. \tag{3}$$

1 Strukturerweiterung mit TS Fuzzy Systemen

Für lineare, zeitdiskrete Systeme der Form $x_{k+1} = Ax_k + Bu_k$, $y_k = Cx_k$ lautet das Regelgesetz eines Dual-Mode MPC

$$u_k = \begin{cases} -Kx_k + c & \text{wenn } 0 \leq k < n_c \\ -Kx_k & \text{wenn } k \geq n_c \end{cases} \tag{4}$$

Der affine Term c repräsentiert hierbei den ersten Eintrag im Vektor $c_{\text{vec}} = [c_k, c_{k+1}, \cdots, c_{k+n_c}]^T, c_{\text{vec}} \in \mathbb{R}^{(n_c \times 1)}$, der zugleich die Optimierungsvariablen beinhaltet. c ist der Ausgang des MPC. Die Länge des Prädiktionshorizonts wird mit n_c abgekürzt. Die quadratische Kostenfunktion

$$J = \sum_{k=0}^{\infty} x_{k+1}^T Q x_{k+1} + u_k^T R u_k \tag{5}$$

$$= x_k^T S_X x_k + 2 x_k^T S_{XC} c_{\text{vec}} + c_{\text{vec}}^T S_C c_{\text{vec}}$$

wird über die diskrete algebraische Riccati-Gleichung gelöst und anschließend dem Optimierer übergeben. Wichtig ist hierbei, dass die LQ-Zustandsrückführungen nach der gleichen Kostenfunktion berechnet werden.

Die Implementierung von TS Fuzzy Systemen in diese Struktur bietet nun zwei Möglichkeiten. Die Erste belässt den affinen Term c im Regelgesetz. Dies ist gleichzusetzen mit der Verwendung eines einzigen MPC im Regelkreis. Gleichzeitig muss das PDC-Regelgesetz anstelle der einfachen Zustandsrückführung eingesetzt werden.

$$u_k = \begin{cases} -\sum_{i=1}^{N} K_i h_i(z) x_k + c & \text{wenn } 0 \leq k < n_c \\ -\sum_{i=1}^{N} K_i h_i(z) x_k & \text{wenn } k \geq n_c \end{cases} \tag{6}$$

Die zweite Variante orientiert sich an dem PDC-Prinzip und verwendet N parallel angeordnete MPC. Der affine Term c_i stellt den Ausgang des i-ten MPC dar.

$$u_k = \begin{cases} -\sum_{i=1}^{N} K_i h_i(z) x_k + \sum_{i=1}^{N} h_i(z) c_i & \text{wenn } 0 \leq k < n_c \\ -\sum_{i=1}^{N} K_i h_i(z) x_k & \text{wenn } k \geq n_c \end{cases} \tag{7}$$

Für beide Varianten gilt, dass die quadratische Kostenfunktion für alle N Teilmodelle gelöst werden muss. Dabei ist eine gemeinsame positivsemidefinite Kostenmatrix zu finden, die für alle Teilmodelle Gültigkeit besitzt. Dieses Problem kann als Optimierungsproblem mit linearen Matrixungleichungen formuliert und mit effizienten Lösungsmethoden gelöst werden [3]. An dieser Stelle wird auf die genaue Aufführung der einzelnen Schritte verzichtet.

2 Methoden zur Approximation des MAS

Ein wichtiger Schritt in der Dual-Mode MPC-Methode ist die Bestimmung der MAS, die alle Anfangs- und Folgezustände beinhaltet, sodass die Stell- und Zustandsgrößenbeschränkungen eingehalten werden. Die MAS wird als Ungleichungsnebenbedingung (UNB) im Solver verwendet. Diese polyedrische Menge kann bei vorhandenem linearen System in einer effizienten Art und Weise mit dem Algorithmus von Pluymers bestimmt werden (s. Alg. 1 und [4]). Die Funktionsweise des Algorithmus wird in [4] detailliert beschrieben. Die Stell- und Zustandsgrößenbeschränkungen $\underline{u} \leq u_k \leq \overline{u}, \underline{x} \leq x_k \leq \overline{x}$ werden üblicherweise beschrieben mit $\mathbb{X} = \{x : x \in \mathbb{R}, \underline{x} \leq x_k \leq \overline{x}, k = 0, 1, \ldots, \infty\}$ und $\mathbb{U} = \{u : u \in \mathbb{R}, \underline{u} \leq u_k \leq \overline{u}, k = 0, 1, \ldots, \infty\}$. Diese Mengen werden als Polyeder in die Beziehungen $A_{y,i} x_k \leq b_{y,i}$ überführt.

$$x_k^T \left[E^T, -E^T, -K_i^T, K_i^T\right]^T \leq \left[\overline{x}^T, -\underline{x}^T, \overline{u}^T, -\underline{u}^T\right]^T \tag{8}$$

In dieser Formulierung ist bereits die für das i-te TS Fuzzy Teilmodell zugeteilte Zustandsrückführung K_i aufgeführt. Die MAS wird definiert als $\mathbb{S} = \{x_k : x_k \in \mathbb{X}, u_k \in \mathbb{U}\} \Leftrightarrow \{x_k : A_S x_k \leq b_S\}$ und repräsentiert damit eine positiv-invariante Menge. Ein Teilmodell $\Phi_i = A_i - B_i K_i$ gilt als vollständig aktiviert, wenn die zugeteilte Fuzzy-Basisfunktion h_i vollständig aktiviert ist ($h_i = 1$, alle Anderen sind gemäß Definition null). Nachfolgend werden zwei Methoden zur Approximation von $\mathbb{S}$ für TS Fuzzy Systeme erläutert.

Die erste Methode verwendet den bereits erwähnten Algorithmus 1 wie folgt: Anstelle einer Matrix Φ, werden diesem $i = 1, 2, \ldots, N$ Teilmodelle Φ_i übergeben. Analog wird mit den $j = 1, 2, \ldots, N$ Beschränkungen $A_{y,j} x_k \leq b_{y,j}$ verfahren. In Zeile 4 von Algorithmus 1 wird dann für jedes i-te Teilmodell Φ_i ein lineares Optimierungsproblem mit der j-ten UNB $A_{y,j} x_k \leq b_{y,j}$ gelöst. Daraus resultieren N MAS $\mathbb{S}_i$. Die Schnittmenge aller $\mathbb{S}_i$ liefert eine genäherte MAS $\mathbb{S}_{M1} = \bigcap_{i=1}^{N} \mathbb{S}_i$. Die Methode 1 ignoriert den Einfluss der Fuzzy-Basisfunktionen h_i. Im Gegensatz hierzu berücksichtigt die zweite Methode die $h_i(s)$ derart, dass diese an festgelegten Stellen $s = \underline{z}, \underline{z} + \delta, \underline{z} + 2\delta, \ldots, \overline{z}$ ausgewertet werden. δ ist so zu wählen, dass $P = \frac{\overline{z} - \underline{z}}{\delta} + 1$ ganzzahlig ist. Hierdurch erhält man P gewichtete Teilmodelle

$$\Phi_q = \sum_{i=1}^{N} A_i h_i(s) - \sum_{i=1}^{N} B_i h_i(s) \cdot \sum_{i=1}^{N} K_i h_i(s),\ q = 1, 2, \ldots, P \qquad (9)$$

Weiterhin werden die Zustandsrückführungen K_i in der Ungleichung (8) entsprechend durch $\sum_{i=1}^{N} K_i h_i(s)$ ersetzt. Die so ermittelten $\Phi_q, A_{S,q}, b_{S,q}$ werden dann dem Algorithmus in gleicher Art und Weise wie in Methode 1 übergeben. Als Zwischenergebnis werden P MAS $\mathbb{S}_q$ zurückgegeben. Diese werden gemäß den Stützstellen s nachträglich beschränkt und zur Menge $\mathbb{S}_{M2} = \bigcup_{q=1}^{P} \mathbb{S}_q$ vereinigt. Es ist ersichtlich, dass der Rechenaufwand mit kleiner werdendem δ erheblich zunimmt. Allerdings ergibt sich dadurch eine höhere Approximationsgüte der MAS.

3 Ergebnisse

Die beiden Methoden werden im Folgenden anhand eines Beispiels aus [4] simulativ untersucht. Hierbei handelt es sich um ein LQ-zustandsgeregeltes ($Q = E, R = 1$) Doppelintegrator-System mit zwei Teilmodellen, dessen

Algorithmus 1.: MAS

Voraussetzung: lineares, zeitdiskretes System, asymptotisch stabil, Beschränkungen $\mathbb{X}, \mathbb{U}$

1: $A_S = A_y,\ b_S = b_y,\ i = 1,\ r = \text{rows}(A_S)$
2: **while** $i \leq r$ **do**
3: $\quad a = (A_S)_{(i,:)},\ b = (b_S)_{(i,:)}$ ▷ Wähle i-te Zeile aus
4: $\quad c = \max_x a\Phi x_k - b$, s.t. $A_S x_k \leq b_S$ ▷ Suche nach einer Beschränkung $a\Phi x_k \leq b$, welche $\mathbb{S}$ verkleinert.
5: $\quad$ **if** $c > 0$ **then**
6: $\quad\quad A_S = \begin{bmatrix} A_S \\ a\Phi \end{bmatrix},\ b_S = \begin{bmatrix} b_S \\ b \end{bmatrix}$ ▷ Wenn eine solche Beschränkung gefunden ist, füge diese A_S und b_S hinzu.
7: $\quad i = i + 1$

zweiter Zustand $x_{2,k}$ nichtlinear mit dem ersten Zustand verkoppelt ist. Die Abtastzeit beträgt $T_A = 1\,\text{s}$.

$$A_1 = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 1 & 0.2 \\ 0 & 1 \end{bmatrix} \tag{10}$$

$$B_1 = B_2 = B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad C_1 = C_2 = C = E \tag{11}$$

$$h_1(x_2) = -0.1x_2 + 1, \quad h_2(x_2) = 1 - h_1(x_2) \tag{12}$$

$$K_1 = \begin{bmatrix} 0.589 & 0.712 \end{bmatrix}, \quad K_2 = \begin{bmatrix} 0.563 & 0.795 \end{bmatrix} \tag{13}$$

In Abbildung 1 werden die approximierten MAS in der Zustandsebene mitsamt drei Anfangzuständen dargestellt. Es fällt auf, dass $\mathbb{S}_{M1} \subset \mathbb{S}_{M2}$ gilt. Die Menge $\mathbb{S}_{M2}$ aus Methode 2 ist „größer“ als $\mathbb{S}_{M1}$. $\mathbb{S}_{M2}$ beinhaltet demnach mehr zulässige Anfangszustände. Die Berücksichtigung der Fuzzy-Basisfunktionen $h_{1,2}(x_2)$ in der zweiten Methode liefert hier eine bessere Approximation. Die zwei Anfangszustände $x_{0,1}$ und $x_{0,2}$ beginnen auf dem Rand der MAS $\mathbb{S}_{M2}$. Sie halten die Beschränkungen $-0.4 \leq u_k \leq 1$ sowie $-10 \leq x_k \leq 10$ ein. Befindet sich ein Anfangszustand, wie z.B. $x_{0,3}$ außerhalb der MAS, so werden die Beschränkungen verletzt (s. Abtastschritt $k = 0$). Die parallel zur x_1-Achse eingezeichneten Linien in $\mathbb{S}_{M2}$ grenzen die Teilmengen $\mathbb{S}_q, q = 1, 2, \ldots, P = 21$ voneinander ab.

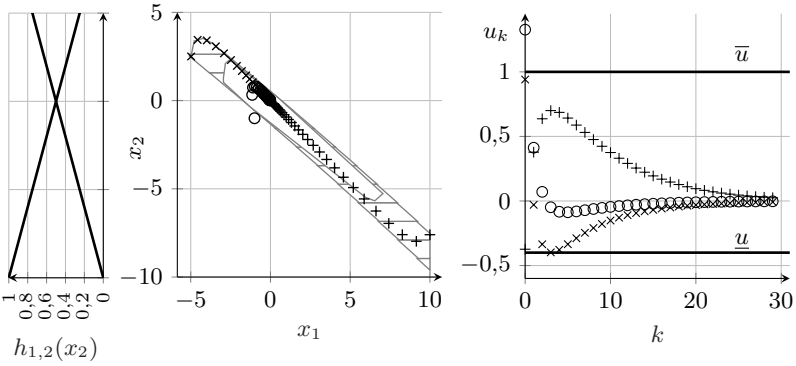


Bild 1: Links: Fuzzy-Basisfunktionen $h_1(x_2), h_2(x_2)$. Mitte: Approximierte MAS ($\mathbb{S}_{M1} \subset \mathbb{S}_{M2}$) und Anfangszustände $x_{0,1} = [-1, -1]^T$ ($\circ$), $x_{0,2} = [-4.975, 2.5]^T$ ($\times$), $x_{0,3} = [10, -7.5]^T$ ($+$). Rechts: Zugehörige Stellgrößentrajektorien $-0.4 \leq u_k \leq 1$. Mengenplots erstellt mit [5].

4 Fazit

Die Untersuchungen in diesem Beitrag befassen sich mit Vorarbeiten zur Implementierung von TS Fuzzy Systemen in die Dual-Mode MPC-Struktur. Hierzu werden auf die zwei möglichen Erweiterungen der Regelgesetze eingangen. Es werden zwei Methoden zur Approximation der maximal zulässigen Mengen (MAS) speziell für PDC-geregelte TS Fuzzy Systeme vorgestellt. Die MAS werden später als Ungleichungsnebenbedingungen im quadratischen Solver benötigt und umfassen alle Anfangszustände, welche die Stell- und Zustandsgrößenbeschränkungen einhalten. Die beiden Methoden werden anhand eines Beispiels simulativ untersucht. Zukünftige Arbeiten werden die hier vorgestellten Methoden für Dual-Mode MPC mit TS Fuzzy Modellen anwenden.

Literatur

[1] P.M.O. Scokaert und J.B. Rawlings. „Infinite Horizon Linear Quadratic Control with Constraints“. San Francisco: 13th IFAC Triennal World Congress. 1996.

[2] H.O. Wang, K. Tanaka und M. Griffin. „Parallel distributed compensation of nonlinear systems by Takagi-Sugeno fuzzy model“.In: *Proceedings of IEEE International Conference on Fuzzy Systems* 2 S. 531-538. 1995.

[3] S. Boyd und L. Vanderberghe. „Convex Optimization“. Cambridge University Press. 2004.

[4] B. Pluymers et al. „The Efficient Computation of Polyhedral Invariant Sets for Linear Systems with Polytopic Uncertainty“ In: *Proceedings of the American Control Conference*, Portland, S. 804-809. 2005.

[5] M. Herceg et al. „Multi-Parametric Toolbox 3.0.“ In: *Proceedings of the European Control Conference*, Zürich, S. 502-510. 2013.

Benchmark-Untersuchung zur Regelung schwach gedämpfter Systeme bei industriepraktischen Anwendungen

Tobias Loose, Thomas Pospiech

Hochschule Heilbronn, Fakultät für Technische Prozesse
Max-Planck-Str. 39, 74081 Heilbronn
E-Mail: tobias.loose@hs-heilbronn.de, thomas.pospiech@hs-heilbronn.de

1 Einführung

Es existieren einige Industrieanlagen, bei denen Teilsysteme einen kleinen Dämpfungsgrad besitzen. Durch eine üblicherweise hohe Prozesstaktung werden diese zum Schwingen angeregt. In diesem Beitrag werden verschiedene schwach gedämpfte Systeme aus der Industriepraxis vorgestellt. Zudem werden hierfür klassische Methoden aus der Industriepraxis gezeigt, um die Schwingungen zu dämpfen. Erste Ansätze für CI-Methoden werden beschrieben, die als Alternative zum Einsatz kommen können. Die Vor- und Nachteile der Methoden sollen als zukünftige Benchmark-Untersuchung dienen. Der Beitrag soll damit Anregungen und Ideen geben, um industriepraktische Fragestellungen anschaulich in der Hochschullehre einzubinden.

Folgende Industrieanwendungen werden u. a. gezeigt:

- Schwappfreie Positionierung von Flüssigkeitsbehältern bei Abfüllanlagen,
- ventilgesteuerte fluidtechnische Anlagen wie hydraulische, umformende Werkzeugmaschinen, z. B. Pressen, oder Pneumatikanlagen,
- weitere Anwendungen wie z. B. Regalbediengeräte, Kran-Pendeldämpfungen.

Klassische etablierte Methoden sind u. a. Einstellung der Rampenzeit in Abhängigkeit der Eigenfrequenz bzw. deren Periodendauer, Verwendung einer Beschleunigungs- oder Geschwindigkeitsrückführung, Implementierung eines schaltenden I-Anteils. Neue CI-Methoden, die zum Einsatz kommen können, sind u. a. Optimierungsansätze mit a-priori-Wissen, um das Schwappverhalten zu erkennen oder um charakteristische Parameter

Tabelle 1: Erfahrungswerte aus der Industriepraxis, typische Eigenfrequenzen f_0 für den Maschinenbau, siehe auch [9]

Eigen-frequenz	Bewertung	Beisiel
< 4 Hz	(sehr) schwer handhabbar	Regalbediengerät mit $f_0 \approx 1 \ldots 2$ Hz
> 15 Hz	solider Maschinenbau, recht „steife Maschinen“	Werkzeugmaschinen, z. B. Drehmaschinen mit $f_0 \approx 30 \ldots 40$ Hz
$\gg 15$ Hz	Sondermaschinen „supersteif“	Nibbelmaschinen mit $f_0 \approx 200 \ldots 600$ Hz

einer Anlage automatisiert zu ermitteln. Damit soll ein besseres Verständnis, insbesondere bei jungen Ingenieuren, für die zu regelnden Anlagen erzielt, als auch der Einsatz bzw. die passende Auswahl bestimmter Methoden gezeigt werden.

2 Industrieanwendungen mit kleinem Dämpfungsgrad

Konstruktionsbedingt weisen Maschinen und Anlagen bestimmte Eigenfrequenzen auf. Von besonderer Bedeutung ist meistens die niedrigste, erste Eigenfrequenz f_0 (des ungedämpften Systems), die ein sehr charakteristischer Anlagen-Parameter ist, siehe Tabelle 1. Zwar liegt die Taktungsfrequenz der Anlagen i.d.R. unterhalb der Eigenfrequenz (üblicherweise durch die Anwendung vorgegeben, dadurch ist eine weitere konstruktive Erhöhung der Eigenfrequenzen nicht notwendig), dennoch treten bei schwach gedämpften Anlagen Schwingungen durch Anregungs- und Prozessfunktionen auf. Derartige Industrieanlagen werden nachfolgend exemplarisch gezeigt.

2.1 Schwappfreie Positionierung von Flüssigkeitsbehältern bei Abfüllanlagen

Moderne vollautomatische Abfüllanlagen von flüssigen Lebensmitteln oder Pharmaka lassen sich in kontinuierlich und diskontinuierlich ar-

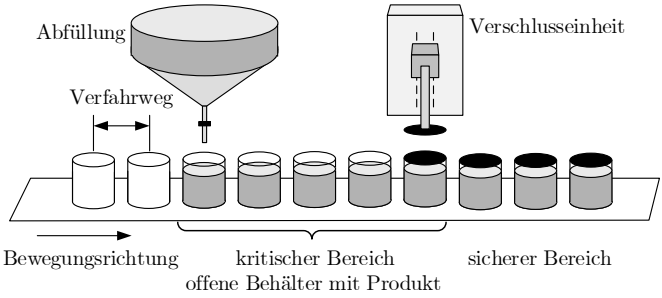
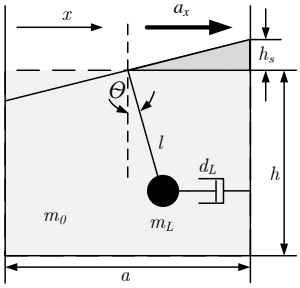


Bild 1: Prinzip einer getakteten Abfüllanlage mit kritischem Bereich

beitende Anlagen einteilen. Bei kontinuierlichen Anlagen stoppt das Transportband nicht, um die einzelnen Arbeitsschritte durchzuführen, wie z. B. die Behälterreinigung, das Befüllen und Verschließen. Der mechanische und elektrische Aufwand solcher Maschinen ist erheblich höher und entsprechend kostenintensiv, weil die Arbeitsstationen mitgeführt werden müssen. Bei diskontinuierlichen (getakteten) Maschinen sind die oben genannten Arbeitsschritte stationär. Das Transportband und somit auch die Behälter müssen bei jedem Abfüllvorgang gestoppt und wieder gestartet werden. Dem kostengünstigeren Aufbau steht somit ein kritischer Arbeitsbereich gegenüber, siehe Bild 1.

Der kritische Arbeitsbereich ergibt sich durch das Starten und Stoppen des Transportbandes bzw. der Behälter, wodurch die Flüssigkeit in den Behältern zu einer Bewegung angeregt wird. Ist diese Bewegung zu groß, so wird die Flüssigkeit aus den Behältern austreten. Zum einen bedeutet dies Produktverlust und zum anderen eine Verunreinigung des Behälterrandes, wodurch ein luftdichtes Verschweißen im nachfolgenden Produktionsschritt unmöglich wird. Beide Szenarien sind bei einer industriellen Abfüllung nicht akzeptabel. Zudem wird eine maximale Produktionsmenge angestrebt, was zu einer minimalen Positionierzeit führt. Zusammen mit dem eigentlichen Abfüllvorgang, stellt somit das Verfahren der Behälter die größte Prozesszeit bei derartigen Anlagen dar. Zusammenfassend lässt sich folgende Problemstellung formulieren: Unmittelbar nach dem Stoppen des Transportbands muss die Flüssigkeit in Ruhe sein und darf keine Bewegungen mehr durchführen, da ansonsten ein „Aufschwappen“ eintreten kann (Resonanz). Des Weiteren darf eine maximale Auslenkung der Flüssigkeit im Behälter, bei minimaler Verfahrzeit, nicht überschritten werden.



a	Behälterdurchmesser
a_x	Behälterbeschleunigung
d_L	Dämpfungsparameter (Viskosität)
g	Erdbeschleunigung
h	Füllhöhe der Flüssigkeit
h_s	Auslenkung der Flüssigkeit (Behälterwand)
l	Länge des virtuellen Pendels
m_0	„Ortsfeste“ Masse der Flüssigkeit
m_L	Schwingender Masseanteil der Flüssigkeit
Θ	Winkel der Auslenkung der Flüssigkeit

Bild 2: Approximiertes Modell: Verhalten von Flüssigkeiten in einem Behälter

Das Verhalten von Flüssigkeiten in einem Behälter kann durch ein gedämpftes Pendelmodel angenähert werden, siehe [2, 14] und Bild 2.

Die Flüssigkeiten sind mit folgenden Eigenschaften angenähert:

- Keine Viskositätsänderung durch Temperatur ϑ, Druck p, Zeit t,
- konstante Dichte ρ (inkompressible Flüssigkeit),
- keine Übertragung von Schubspannungen (Newton-Fluid).

Die Bewegungsdifferentialgleichung lässt sich mit

$$\ddot{\theta} = -\frac{d}{m_L} \cdot \dot{\theta} - \frac{g}{l} \cdot \theta + \frac{1}{l} \cdot \ddot{x} \tag{1}$$

für kleine Flüssigkeit-Auslenkungswinkel θ berechnen. Die Behälterbeschleunigung $a_x(t) = \ddot{x}(t)$ steht mit der Flüssigkeitsauslenkung

$$h_s = -\frac{a_x}{2} \cdot \theta \tag{2}$$

in Zusammenhang. Hiermit lässt sich die Übertragungsfunktion

$$G(s) = \frac{\mathcal{L}\{h_s\}}{\mathcal{L}\{a_x\}} = \frac{H_s}{A_x} = \frac{K \cdot \omega_0^2}{s^2 + 2 \cdot D \cdot \omega_0 \cdot s + \omega_0^2} \tag{3}$$

bestimmen, dabei gelten folgende Parameter

$$K = \frac{a}{2 \cdot g},\ \omega_0 = \sqrt{\frac{g}{l}} \text{ und } D = \frac{d_L}{2 \cdot m_L} \cdot \sqrt{\frac{l}{g}}. \tag{4}$$

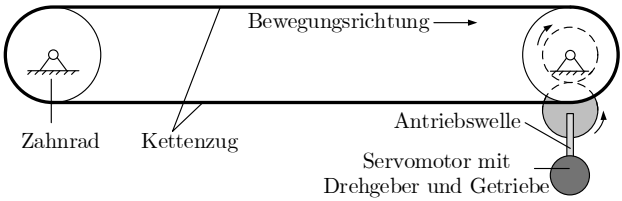


Bild 3: Schematischer Aufbau eines Transporteurs mit einem Kettenzug

2.2 Kettenzugmittel für Abfüllanlagen

Eine getaktete Betriebsart wie in Abschnitt 2.1 beschrieben, führt unter gewissen Umständen auch zu Schwingungen am Transporteur selbst. Speziell im Lebensmittelbereich werden hierfür oft Kettenzugmittel verwendet, auf denen Platten montiert sind in die dann letztendlich die Behälter eingelassen werden. Auch wenn in Bild 1 nur eine Füllstelle dargestellt ist, existieren in Realität mehrere Füllstellen an dieser Station. Das bedeutet, dass der Transporteur entsprechend breit sein muss, was zu bewegten Massen von mehreren 100 kg führt. Als Zugmittel werden hierfür massive Ketten verwendet, die pro Seite (also Vor- und Rückseite der Maschine) bis 50 m lang sein können. Kettenzugmittel können die Eigenschaft haben, dass bei hohen Geschwindigkeiten und einer geringen Zähneanzahl des Zahnrades longitudinale Schwingungen hervorgerufen bzw. angeregt werden. Im vorliegenden Fall kann dieser Polygoneffekt ausgeschlossen werden, da zum einen die oben genannten Gegebenheiten nicht vorhanden sind (rel. langsame Geschwindigkeit und hohe Zähnezahl) und zum anderen keinerlei Schwingung bei unterschiedlichen aber jeweils konstanten Geschwindigkeiten zu erkennen ist. Aus diesem Sachverhalt lässt sich schließen, dass die Schwingungen in der Transporteinheit letztendlich nur durch eine positive oder negative Beschleunigung des Antriebsmotors hervorgerufen werden. Wenn sowohl der Drehgeber, als auch das Antriebszahnrad (vergleiche Bild 3 rechte Seite) unmittelbar nach dem Starten und Stoppen keine Schwingungskomponenten enthält, kann davon ausgegangen werden, dass letztendlich nur die Materialflexibilitäten des Zugmittels für ein entsprechend oszillierendes Verhalten verantwortlich sind. Eine derartige Transporteinheit besitzt mehrere Schwingungsfrequenzen mit größtenteils unterschiedlichen Amplituden A, siehe Bild 4. Auffällig ist die Tatsache, dass die niedrigste Schwingfrequenz die größte Amplitude am entferntesten Ort des Antriebskettenrades besitzt. Das liegt zum einem am erheblichen Massenträgheitsmoment

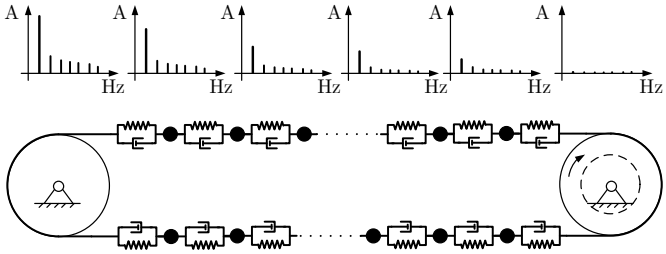


Bild 4: Interpretation der einzelnen Kettenglieder als gedämpfte Feder-Masse-Schwinger

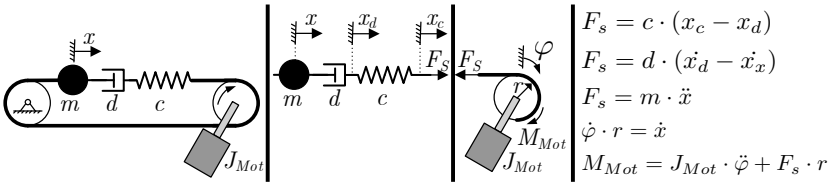


Bild 5: Vereinfachtes Modell eines Kettenzugs (links) mit den Teilmodellen (Mitte) und Hinweise zur Modellbildung (rechts)

des linken Kettenrades und zum anderen (unabhängig des linken Kettenrads) an der Reihenschaltung der einzelnen Feder-Masse-Elemente. Die Amplitude dieser Grundfrequenz geht mit geringerem Abstand zum Antriebsrad zurück, ist aber bei der dargestellten Verteilung nahezu über die gesamte Transporteinheit am größten.

Grundsätzlich gilt folgender Zusammenhang: Die Frequenz der Grundschwingung hängt zum einen vom Massenträgheitsmoment des linken Zahnrades und zum anderen von der Maschinenlänge und dem verwendeten Material der Kette ab. Alle drei „Parameter“ stellen in der Realität Maschinenkonstanten dar, die zwar für unterschiedliche Maschinen andere Werte besitzen können, sich aber für die eine und selbige Maschine nicht verändern. Ausgehend davon lässt sich nun ein vereinfachtes Modell nach Bild 5 erstellen, das den notwendigen Ansprüchen genügt und sich auch in einem Labor umsetzen lässt. Dieser gedämpfte Ein-Masse-Schwinger kann durch die Differentialgleichung

$$M_{Mot} = J_{Mot} \cdot \left[\frac{m}{c \cdot r} \dddddot{x} + \frac{m}{d \cdot r} \dddot{x} + \frac{J_{Mot} + m \cdot r^2}{r \cdot J_{Mot}} \ddot{x} \right] \tag{5}$$

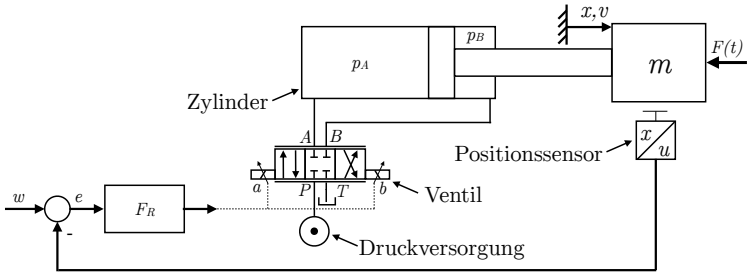


Bild 6: Prinzip einer ventilgesteuerten Hydraulikachse mit Einbindung in einen Regelkreis mit dem Sollwert w, der Regeldifferenz e, den Ventil-Magneten a und b sowie den Ventilanschlüssen Pumpe P, Tank T und den Arbeitsanschlüssen A und B, den Zylinderdrücken p_A und p_B, die Masse der Anwendung m, die Prozesskräfte $F(t)$, Position x und das dazugehörende Spannungssignal u, die Geschwindigkeit v

mit der Masse m, der Federkonstante c und dem Dämpfer d sowie dem Radius r und dem Antriebs-Massenträgheitsmoment J_{Mot} berechnet werden. Diese Gleichung lässt sich für die Beschleunigung $a = \ddot{x}$ in eine Übertragungsfunktion analog zur Gleichung (3) überführen.

2.3 Ventilgesteuerte fluidtechnische Anlagen

In der Hydraulik existieren mehrere Antriebskonzepte, wie z. B. Antriebe mit Motorregelung bzw. Sekundärregelung, pumpengesteuerte Antriebe, ventilgesteuerte Antriebe, siehe [4]. Letztgenannte werden in der Industriepraxis aufgrund der hohen Dynamik meistens verwendet, siehe Bild 6. Der Regler gibt eine Stellgröße an das Ventil, welches die translatorische Bewegung des Zylinders einstellt. Dieses Antriebskonzept zeichnet sich durch geringe Massenträgheiten im Antrieb selbst aus. Massenträgheiten des Ventilkolbens, der Ölsäule und des Zylinderkolbens mit -stange sind ggü. der Anwendung nahezu vernachlässigbar. Hinzu kommt eine extrem hohe Dynamik durch das Ventil (moderne Servoventile öffnen geregelt unterhalb von 10 ms) und ggf. einen Hydrospeicher (der Kurzzeitig enorm viel Leistung freisetzen kann) sowie eine sehr hohe Kraftdichte der Hydraulik. Somit ist die Hydraulik von der Dynamik und der Kraftdichte ggü. einem elektromechanischen Antrieb weit überlegen. Allerdings stehen demgegenüber eine Reihe von Nachteilen, wie z. B. geringer Wirkungsgrad (insbes. durch Druckverluste am Ventil), Aufwändige Inbetriebnahme (u.a. durch viele Nichtlinearitäten), Kompressibilität des Öls (dadurch entste-

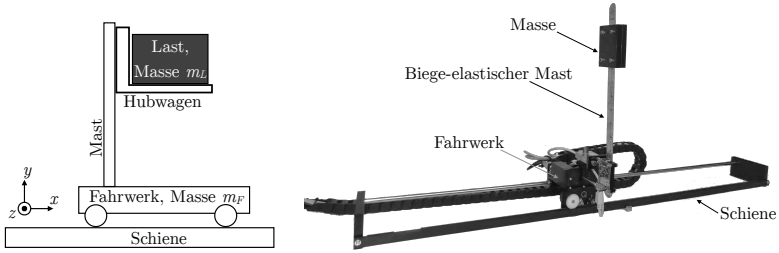
hen niedrige Eigenfrequenzen, die Anlage wird schwingungsfähig), geringe Dämpfung (Schwingungsenergie wird im System durch minimale Leckagen kaum abgebaut). In der Pneumatik verschärfen sich die Probleme insbesondere bei Positionsregelungen wegen der höheren Kompressibilität von Luft. Typische Werte des Dämpfungsgrads bei realen hydraulischen Anlagen sind $D \approx 0,15 \ldots 0,4$. Durch die nachfolgend beschriebenen Methoden lassen sich die Werte etwas anheben. Die Eigenfrequenz hingegen hängt sehr stark von der Konstruktion und der Anlage ab, u.a. lässt sie sich sich durch den Zylinderdurchmesser und -länge, Getriebefaktor, Ventilposition beeinflussen. Sie lässt sich in gewissem Maße allerdings auch durch regelungstechnische Maßnahmen anheben, wie nachfolgend gezeigt. Typische Industrie-Erfahrungswerte für die Eigenfrequenz von $f_0 > 15\,\mathrm{Hz}$ sollten dabei angestrebt werden, siehe auch Tabelle 1. Anwendungsbeispiele sind umformende Werkzeugmaschinen (z. B. hydraulische Pressen), Druckgießmaschinen, Rohrbiegemschinen, Hütten- und Walzwerktechnik (z. B. Stranggießanlagen). Die dynamischen Vorzüge der Hydraulik lassen sich durch Simulationen relativ einfach in die Hochschullehre einbeziehen, wie nachfolgend gezeigt wird. Entsprechende Hydraulik Prüfstände sind demgegenüber relativ aufwändig.

2.4 Regalbediengeräte

Regalbediengeräte (RBG) haben konstruktionsbedingt durch eine große Masse und langen Masten eine niedrige Eigenfrequenz ($f_0 = 0,5 \ldots 2\,\mathrm{Hz}$) und niedrigen Dämpfungsgrad ($D = 0,05 \ldots 0,2$), siehe Bild 7a und [1, 8]. Die Fahrwerke erreichen zwar eine relativ hohe Endgeschwindigkeiten von typischerweise $5\,\mathrm{m/s}$, sie sind aber durch die langen Fahrwege keiner hohen wechselnden Anregung ausgesetzt, d.h. die Taktfrequenz ist gering. Dennoch besteht beim Anhalten und Positionieren der Bedarf an dämpfenden Maßnahmen. Lösungen inkl. deren Vor- und Nachteile dazu wurden bereits in [8] ausführlich dargestellt, wie z. B. Zustandsregelung oder über spezielle Rampenzeiten. Diese Anwendung lässt sich auch sehr anschaulich in die Hochschullehre mit relativ einfachen Modellen und Simulationen integrieren, siehe Bild 7b und [8].

2.5 Krananlagen

Eine Krananlage mit einer hängenden Last ist eine weitere Anwendung schwach gedämpfter Systeme. Modellbildung inkl. regelungstechnischer



(a) Prinzip RBG (b) Labormodell

Bild 7: Regalbediengerät (RBG) als Labormodell, nach [8]

Lösungen zur Dämpfung des Last-Auslenkungswinkels sind in einigen Lehrbüchern beschrieben, z. B. mit Zustandsregelung, siehe [5]. Als Ergänzung theoretischen und simulativen Auslegung lässt sich diese Anwendung sehr gut als Labormodell in die Hochschullehre integrieren, siehe [10].

3 Etablierte Lösungen zur Schwingungsdämpfung

Eine Sammlung geeigneter Lösungen zu konkreten Industriebeispielen wird in diesem Beitrag als Ergänzung zu Lehrbüchern gezeigt, was darin oftmals zu kurz kommt, siehe z. B. [3, 12]. Zudem haben sich bei bestimmten Anwendungen gute, pragmatische Lösungen etabliert, die sich auf andere Anwendungen übertragen lassen und noch nicht durchgesetzt haben. Dieser Beitrag soll dazu ebenso anregen, um Lösungen zu übertragen.

3.1 Vorgabe von Eingangssignalen

Bis ca. Mitte der 90er Jahre waren Industrieanlagen konzeptionell mit einer mechanischen Königswelle aufgebaut und sämtliche Bewegungen wurden über diese (mechanisch gekoppelt) miteinander synchronisiert. Heutzutage sind mechanische Königswellen nur noch sehr selten zu finden, da die Antriebstechnik durch die immer leistungsfähige Servotechnik einen Wandel erlebt hat, auch wenn der konzeptionelle Aufbau der gleiche geblieben ist: Die Königswelle wurde durch einen sog. „virtuellen Master“ ersetzt und sämtliche Bewegungsvorgänge werden auf diesen mittels elektronischer Getriebe oder elektronischen Kurvenscheiben synchron

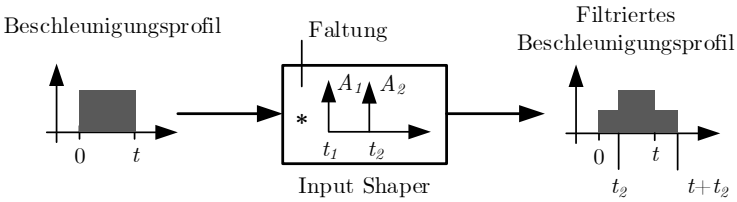


Bild 8: Auswirkung des Input Shaper auf ein rechteckförmiges Beschleunigungsprofil

gehalten (Slave). „Rast-in-Rast“ Bewegungen sind dabei typische, durch die Anwendung notwendige Bewegungsabläufe, siehe VDI-Richtlinien [17]. Werden schwingungsfähigen Systeme durch ruckoptimierte „Rast-in-Rast“ Bewegungen positioniert, so werden sie zu Schwingungen angeregt. Das liegt daran, dass die Anregungsfunktion bestimmte Frequenzen enthält die „in der Nähe“ der System-Eigenfrequenzen liegen. Die normierte geneigte Sinuslinie ist mit

$$x(z) = z - \frac{1}{2 \cdot \pi} \cdot sin\,(2 \cdot \pi \cdot z) \tag{6}$$

für $0 \leq z \leq 1$ definiert, siehe [17].

Zur Verbesserung der Schwingungsanregung kann ein sog. Input Shaper (IS) als spektraler Filter für beliebige Eingangssignale verwendet werden, siehe Bild 8 und [6, 7, 16]. Häufig werden vereinfacht die Beschleunigungssignale in Rechteckform angegeben, siehe [3]. Derartige Beschleunigungsformen sind aber für industrielle Positionierungen nicht geeignet (Geräusche, Verschleiß, ...). Der ZV-Input Shaper (zero vibration) ist über zwei Impulse mit den Amplituden

$$A_1 = \frac{1}{1+k} \quad \text{und} \quad A_2 = \frac{k}{1+k} \quad \text{mit} \quad k = e^{-\frac{D \cdot \pi}{\sqrt{1-D^2}}} \tag{7}$$

und den Zeitpunkten $t_1 = 0$ sowie $t_2 = T_D/2$ definiert. Dadurch erfährt das System zwei definierte Beschleunigungsänderungen und daraus implementierte Kraft-Impulse, siehe Bild 8. Werden diese Impulse im zeitlichen Abstand als ganzzahliges Vielfaches der Schwingungsperiodendauer $T_0 = 1/f_0$ gesetzt, dann werden die Schwingungen reduziert. Der gleiche Effekt kann auch über die gezielte Einstellung einer Rampenzeit als Anregungsfunktion erreicht werden, siehe auch [8].

Hier wird gezeigt, wie ein Input Shaper wirkungsvoll mit einer Rast-in-Rast Bewegung nach VDI-Richtlinien (geneigte Sinuslinie) kombiniert

werden kann und somit die Schwingungen nach dem Positioniervorgang eliminiert werden, ohne dabei die Synchronität zu anderen Maschinenachsen zu verlieren. Ausgehend von der mechanischen Königswelle beträgt ein Maschinentakt 360° (Königswelle dreht sich einmal im Kreis). Für die kalkulatorische Produktionsmenge wird nun die Zeit für einen Takt festgelegt, z.B. $T_{Takt} = 1{,}5\ s$. Sämtliche Bearbeitungsvorgänge beziehen sich nun auf diese 360°. Bei Abfüllanlagen kann für den Transport oftmals nur 120° zur Verfügung gestellt werden, da die anderen Prozessvorgänge (z.B. Abfüllung des Produkts, die Kontrolle der Füllung, usw.) die restlichen 240° benötigen. Diese 120° entsprechen im vorliegenden Beispiel einer tatsächlichen Transportzeit von 0,5 s. Wie jeder Filter, verzögert auch der Input Shaper das Eingangssignal, nämlich hier um die Hälfte der Periodendauer T_D des schwingungsfähigen Systems. Diese Zeit muss wieder aufgeholt, bzw. kompensiert werden, indem die Transportzeit mit Filter um die Hälfte der Periodendauer verkürzt wird, d.h. das Eingangssignal des Filters wird um $T_D/2$ gestaucht. Es spielt dabei keine Rolle, ob der Positionsverlauf (als mechanische Kurvenscheibe), die Geschwindigkeit oder die Beschleunigung entsprechend gefiltert wird, siehe Bild 9. Die Positionierung beträgt 150 mm bei einer Positionierzeit von 0,5 s. Die Antwort des Systems ist in Bild 10 wiedergegeben, indem die entsprechenden Antworten des Systems (Flüssigkeitsauslenkung) mit und ohne Filtrierung durch den Input Shaper dargestellt sind. Wie in diesen Aufzeichnungen zu erkennen ist, schwingt bzw. schwappt die Flüssigkeit nach Beendigung der Positionierung ohne Input shaper deutlich und aufgrund der schwachen Dämpfung des Systems ($D \approx 0,02$) über einen langen Zeitraum. Das hat zur Folge, dass für den nächsten Maschinentakt entsprechend lange gewartet werden muss. Mit der Ansteuerung durch die Filtrierung des Input Shapers zeigt sich eine deutliche Verbesserung bei gleicher Positionierzeit, da bereits nach sehr kurzer Zeit nahezu keine Restschwingungen mehr vorhanden sind und somit für den nächsten Maschinentakt nicht gewartet werden muss.

3.2 Geschwindigkeits- und Beschleunigungsrückführung

Die Geschwindigkeits- und Beschleunigungsrückführung bzw. Lastdruckrückführung sind gängige Methoden bei ventilgesteuerten hydraulischen Systemen, um das Verhalten der Regelstrecke zu verbessern, siehe [4]. Gute Ergebnisse werden bei schwach gedämpften Systemen erzielt, wozu üblicherweise hydraulische Systeme zählen. Früher wurden die beiden Zylinderkammern durch einen Bypass verbunden, um die Dämpfung zu

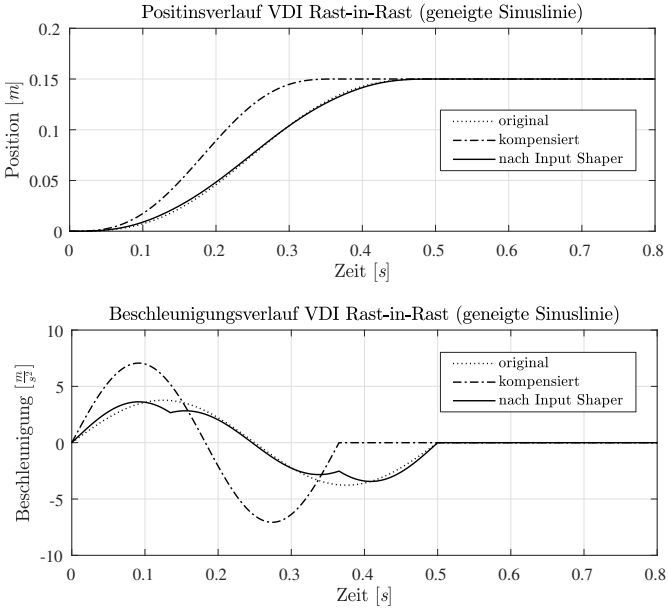


Bild 9: Filtrierte „geneigte Sinuslinie“ nach VDI mit Input Shaper (inkl. zeitlicher Kompensation)

erhöhen (mit dem Nachteil einer künstlichen inneren Leckage zwischen beiden Zylinderkammern). Das ist aber mittlerweile unüblich, da insbesondere durch elektronische Regelungen der Dämpfungsgrad ebenso angehoben werden kann, u.a. mit nachfolgend gezeigten Methode. Eine adäquate hydraulisch-mechanische Realisierung der Methode ist im Vergleich zu aufwändig und schwer einstellbar. Der früher angewandte Bypass ist zwar eine geschwindigkeitsabhängige Gegenkopplung auf auf das System, allerdings geht die Geschwindigkeit quadratisch ein und ist somit schwieriger handhabbar.

Die Methode wird hier exemplarisch an einer schwach gedämpften Strecke

$$F_S = \frac{\mathcal{L}\{v\}}{\mathcal{L}\{y_2\}} = \frac{K_S \cdot \omega_0^2}{s^2 + 2 \cdot D \cdot \omega_0 \cdot s + \omega_0^2} \tag{8}$$

in einem Positionsregelkreis mit dem Regler F_R gezeigt, siehe Bild 11. Die Strecke soll eine stark vereinfachte hydraulische Achse repräsentieren, die bei einer bestimmten Ventilöffnung y_2 eine Zylinder-Geschwindigkeit v

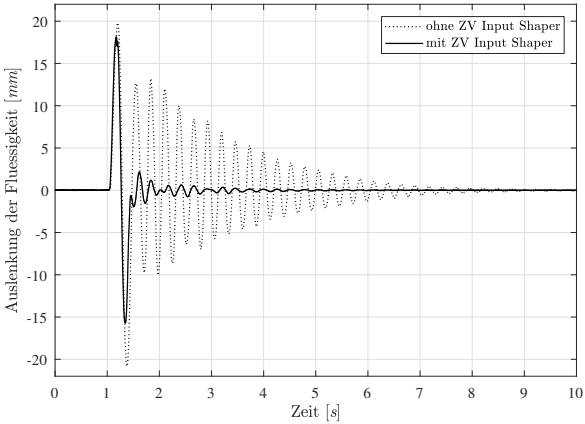


Bild 10: Flüssigkeitsauslenkung nach einer Rast-in-Rast Bewegung (Geneigte Sinusline) mit und ohne ZV Input Shaper

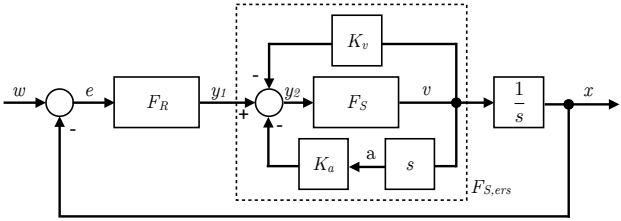
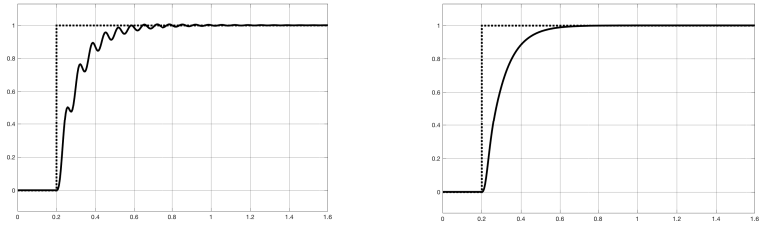


Bild 11: Prinzip der Beschleunigungsrückführung

aufbaut. Über den Integrator $\frac{1}{s}$ wird dann die Position x zurück geführt. Die Streckenparameter Verstärkungsfaktor K_S, Dämpfungsgrad D und Eigenkreisfrequenz des ungedämpften Systems ω_0 sind Maschinen- bzw. Anlagenwerte und über die Konstruktion teilweise beeinflussbar, wie z.B. die Wahl des Zylinder-Durchmessers, der sich auf K_S und ω_0 auswirkt.



(a) ohne Beschleunigungsrückführung (b) mit Beschleungigungsrückführung

Bild 12: Exemplarische simulierte Wirkung einer Beschleungigungsrückführung

Die Geschwindigkeit v lässt sich nun über einen Verstärkungsfaktor K_v als Gegenkopplung ($K_v > 0$) oder Mitkopplung ($K_v < 0$) auf das Stellsignal y_1 zurück führen. Ebenso kann die Beschleunigung a über den Verstärkungsfaktor K_a zurück geführt werden. Damit soll das Streckenverhalten positiv beeinflusst werden. Dadurch entsteht eine Ersatz-Übertragungsfunktion

$$F_{S,ers} = \frac{K_S \cdot \omega_0^2}{s^2 + \underbrace{\left(2 \cdot D \cdot \omega_0 + K_a \cdot K_S \cdot \omega_0^2\right)}_{=2 \cdot D_{ers} \cdot \omega_{0,ers}} s + \underbrace{\left(\omega_0^2 + K_v \cdot K_S \cdot \omega_0^2\right)}_{=\omega_{0,ers}^2}} \quad (9)$$

der Regelstrecke, siehe auch Bild 11.

Durch die Rückführung der Geschwindigkeit v über den Verstärkungsfaktor K_v kann bei Gegenkopplung ($K_v > 0$) die Eigenkreisfrequenz des ungedämpften Systems ω_0 vergrößert werden. Es entsteht eine Ersatz-Eigenkreisfrequenz $\omega_{0,ers} = \omega_0 \cdot \sqrt{1 + K_v \cdot K_S}$. Im Industriepraxis-Jargon heißt es dann, dass das System dadurch steifer wird. Dies geht leider einher mit einer Reduzierung des Dämpfungsgrades, der dann mit $D_{ers} = \frac{D}{\sqrt{1+K_v \cdot K_S}}$ kleiner als D ist (K_a sei dabei noch zu Null gesetzt). Eine Mitkopplung ($K_v > 0$) wird ebenso praktiziert, um die Dämpfung zu erhöhen, leider zu Lasten von ω_0.

Mit einer Beschleunigungsrückführung kann gezielt der Dämpfungsgrad angehoben werden, siehe Bild 12. Die Eigenkreisfrequenz bleibt dabei unverändert, siehe $\omega_{0,ers}$ in Gleichung (9), hier geht der Verstärkungsfaktor K_a nicht mit ein. Der Dämpfungsgrad vergrößert sich dann mit $D_{ers} = D + \frac{1}{2} \cdot K_a \cdot K_S \cdot \omega_0$ (K_v sei dabei wegen der Übersichtlichkeit zu Null gesetzt). Anhand dieser Formeln lässt sich die Methode auch vor

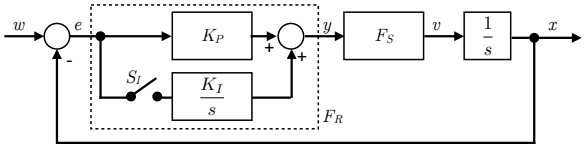


Bild 13: Prinzip des schaltenden I-Anteils

Ort an einer Anlage sehr gut überschlägig anwenden, vorausgesetzt die Streckenparameter wie K_S und ω_0 sind bekannt. Somit sind die Methoden für den Industrieeinsatz gut geeignet, weil sie gut interpretierbar und damit heuristisch vor Ort einstellbar ist.

3.3 Schaltender I-Anteil

Bei ventilgesteuerten hydraulischen Systemen ist der schaltende I-Anteil eine sehr pragmatische Industriepraxis-Methode, siehe [4, 11]. Zwar wird dabei der Dämpfungsgrad des Regelkreises nicht angehoben, dennoch kann aber die Dynamik des Systems durch eine höhere Wahl der Reglerverstärkung erhöht werden. Da ventilgesteuerte Hydraulikanlagen selbst integrierendes Verhalten haben (Ventilöffnung baut Zylindergeschwindigkeit auf) sowie schwach gedämpft und schwingungsfähig sind, kann ein I-Anteil des Reglers hinderlich sein. Sicherlich lässt sich das über regelungstechnische Lösungen überwinden, wie z. B. mit dem symmetrischen Optimum, das ursprünglich für elektromechanische Antriebe entwickelt wurde, siehe [5, 12, 13, 15]. Allerdings setzt das bei der Inbetriebnahme eine genaue Kenntnis des Streckenmodells voraus, was in der Praxis im Anlagenbau nicht immer zugänglich ist. Zudem ist die Hydraulik durch einige Nichtlinearitäten gekennzeichnet, wie z. B. die Durchfluss-Charakteristik von Ventilen. Daher wurde die pragmatische Lösung mit dem schaltenden I-Anteil für die Hydraulik entwickelt, um unterschiedliche Lastzustände auszugleichen.

Bei dieser Methode wird der I-Anteil des Reglers über einen Schalter S_I dazu geschaltet, wenn drei Bedingungen erfüllt sind, siehe Bild 13:

- Die Geschwindigkeit v der Hydraulikachse sowie
- die Regeldifferenz e unterschreiten jeweils einen bestimmten Wert und
- eine bestimmte Zeit ist seit dem Unterschreiten verstrichen.

4 Ansätze für CI-Methoden als Benchmark-Untersuchung für die Hochschullehre

Für die Inbetriebnahme industrieller Anlagen stehen pragmatische, interpretierbare Lösungsansätze im Vordergrund, siehe [10] und die hier gezeigten Methoden. Voraussetzung für deren Einsatz ist allerdings die Kenntnis wichtiger Systemparameter wie z. B. Eigenfrequenz f_0, Dämpfungsgrad D. Eine robuste Lösung zur automatisierten Erkennung des Systemverhaltens inkl. interpretierbarer Systemparameter konnte sich für viele Industrieanwendungen noch nicht durchsetzen. Es bedarf dabei immer noch einem Experten, der das Systemverhalten inkl. Parametern interpretiert. Selbstverständlich existieren auch automatisierte Methoden die eine Systemidentifikation als mathematisches Modell abbilden, z.B. ARMA-Modelle [12]. Allerdings ist die Interpretation für einen industriepraktischen Einsatz sehr schwer. Dieser Beitrag liefert an einem Beispiel erste Ideen, wie die wichtigen Systemparameter und später auch ein Systemmodell automatisiert extrahiert werden können.

In Kapitel 3.1 wurde gezeigt, wie (beliebige) Eingangssignale mit Hilfe eines ZV Input Shapers, so verformt werden können, dass die Restschwingungen von einem schwach gedämpften System (hier: Flüssigkeit in einem Behälter) reduziert bzw. im Idealfall eliminiert werden können. Der ZV Input Shaper wird durch die zwei Impulse eingestellt, d.h. über deren Amplituden und Zeitpunkten, siehe Gleichung (7). Zur Einstellung müssen die Systemparameter Dämpfungsgrad D und Periodendauer T_D bekannt sein. Prinzipiell lassen sich die Systemparameter mit zwei unterschiedlichen Arten ermitteln.
Ein theoretischer Ansatz ist die Parameterermittlung über das approximierte mathematische Modell. Ein praktischer Ansatz hingegen erfolgt über eine Parameteridentifikation mit Hilfe eines Sensors (z.B. zum Zeitpunkt des Einrichtbetriebes der entsprechenden Maschine), indem die Impulsantwort des Systems ausgewertet wird. Es kann aber auch die Systemantwort eines Maschinentaktes ausgewertet werden, wie nachfolgend gezeigt wird.
Für die Berechnung der Systemparameter ω_0 und D müssen nach [7] mindestens zwei aufeinanderfolgende Extrema der Impulsantwort gefunden werden, die nach dem Einschwingvorgang auftreten. Im einfachsten Fall kann von zwei aufeinanderfolgenden Messwerten der größere Wert über einen entsprechenden Vergleich ermittelt werden. Der Zeitpunkt

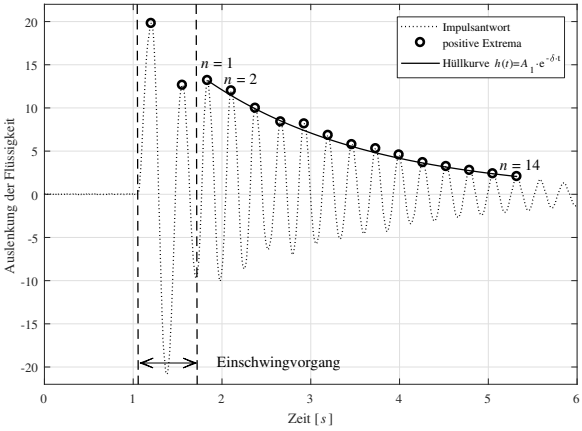


Bild 14: Ermittlung positiver Extrempunkte und Darstellung der Hüllkurve

des Messwertes ergibt sich über die deterministische Funktionsweise der Steuerung und der dabei eingestellten Zykluszeit.

Messfehler werden pragmatisch durch einen gleitenden Mittelwert über mehrere Extrempunkte ausgeglichen, siehe Bild 14. Zudem wird über mehrere m Perioden eine mittlere Periodendauer berechnet, um weitere Messfehler zu reduzieren. Über das logarithmische Dekrement

$$\Lambda = ln\left(\frac{A_n}{A_{n+m}}\right) = \delta \cdot m \cdot T_d \tag{10}$$

wird der Dämpfungsgrad

$$D = \frac{1}{\sqrt{1 + \left(\frac{2 \cdot m \cdot \pi}{\Lambda}\right)^2}} \tag{11}$$

berechnet, siehe [12, 7]. Darin sind mehrere Perioden m einbezogen, um Messfehler zu reduzieren. Die Berechnung der Hüllkurve

$$h\,(t) = A_n \cdot e^{-\delta \cdot t} \tag{12}$$

erfolgt über den Abklingkoeffizienten δ. Diese beschriebene Prozedur lässt sich auf Maschinensteuerungen problemlos realisieren, sodass zur Laufzeit des Steuerungsprogrammes die charakteristischen Parameter eines schwingungsfähigen Systems (ω_0 und D) bestimmt und somit die

Einstellparameter für z.B. einen ZV Input Shaper automatisiert vorgegeben werden können. Außerdem können die entsprechend gefundenen Systemparameter mit Gleichung (10) und einer graphischen Anzeige kontrolliert werden.

Die Identifikation der Eigenfrequenz inkl. deren Abhängigkeit von Parametern ist ein aufwändiger Prozess, indem viel Expertenwissen einfließen muss, siehe [7]. Daraus wurde erkannt, dass z. B. die Eigenfrequenz bei Flüssigkeiten i. W. von der Füllhöhe und dem Behälterdurchmesser abhängt, siehe Bild 15. Automatisierte Identifikationen existieren noch nicht, sollen aber eine Anregung für zukünftige Arbeiten sein.

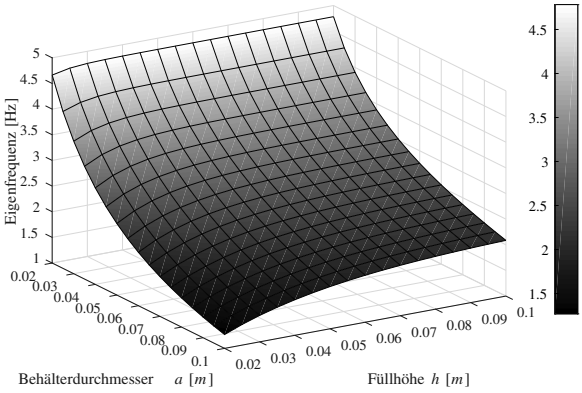


Bild 15: Identifikation der Eigenfrequenzen

5 Zusammenfassung

Dieser Beitrag zeigt einige industriepraktische Beispiele von schwach gedämpften, schwingungsfähigen Systemen, wie z. B. das Schwapp-Verhalten von Flüssigkeiten in einer Abfüllanlage, ventilgeregelte Hydraulikanlagen, Regalbediengeräte. Es wurden typische, pragmatische Lösungen aus den einzelnen Anwendungen gezeigt, die sich in der Industriepraxis etabliert haben, z. B. Generierung spezieller Eingangssignale um die Schwingung zu reduzieren. Hauptgründe für die Etablierung sind die gute Interpretierbarkeit und somit Einstellbarkeit vor Ort an der Anlage während der Inbetriebnahme. Allerdings sind hierzu fundierte Prozess- und Anlagenwissen Voraussetzung, um die Einstellung vorzunehmen, z. B. die Kenntnis der Eigenfrequenz.

Dieser Beitrag soll für zukünftige Arbeiten Anregungen liefern, um

- automatisiert Systemmodelle inkl. Parameter zu ermitteln,
- weitere Lösungsmethoden zu etablieren bzw.
- zwischen den Anwendungen die Methoden auszutauschen.

Zudem wurde in diesem Beitrag gezeigt, wie die hier gezeigten Industrieanwendungen sich mit relativ wenig Aufwand als Labormodelle bzw. Simulationen in der Hochschullehre einbinden lassen.

Literatur

[1] M. Bachmeyer, M. Schipplick, T. Thümmel, S. Kessler, W. A. Günther, H. Ulbrich. „Nachschwingungsfreie Positionierung elastischer Roboter durch numerische und analytische Trajektorienplanung am Beispiel Regalbediengerät“. In: *FB 113 Elektrisch-mechanische Antriebssysteme - Innovationen - Trends - Mechatronik*. S. 199-205. 2008.

[2] F. Dodge. „The New Dynamic Behavior of Liquids in Moving Containers“. Research Institute San Antonio, Texas. 2000.

[3] H. Dresig, A. Fidlin. „Schwingungen mechanischer Antriebssysteme“. Springer Vieweg. 2014.

[4] D. Findeisen, S.Helduser. „Ölhydraulik“. Springer Vieweg. 2015.

[5] O. Föllinger. „Regelungstechnik“. VDE Verlag. 2016.

[6] P. Hubinský, T. Pospiech „Input Shaping for Slosh-Free Moving Containers with Liquid“. International Journal of Mechanics and Control, Vol. 09, No. 02. S. 27-34. 2008.

[7] P. Hubinský, T. Pospiech „Slosh-free Positioning of Containers with Liquids and Flexible Conveyor Belt“. Journal of Electrical Engineering Vol. 61, No. 2. S. 65-74. 2010.

[8] T. Loose: „Benchmark-Anwendung zur Schwingungsanalyse und -dämpfung von Regalbediengeräten am Beispiel eines Labormodells“. In: *Proc., 25. Workshop Computational Intelligence* (Hoffmann, F.; Hüllermeier, E., Hg.), S. 127–144. Universitätsverlag Karlsruhe. 2015.

[9] T. Loose: „Projektierung mechatronischer Anlagen als Laborarbeit in der Hochschullehre am Beispiel LEGO® MINDSTORMS®“. In: *Proc., 2. IFToMM D-A-CH Konferenz*, S. 1–8 (Tagungsband auf CD-ROM). Universität Innsbruck. 2016.

[10] T. Loose: „Projektierung mechatronischer Anlagen in der Hochschullehre am Beispiel von Labormodellen“. In: *Proc., Workshop 2017 ASIM/GI-Fachgruppen* (Commerell, W.; Pawletta, T., Hg.), S. 162–167. ARGESIM Verlag Wien, Hochschule Ulm. 2017.

[11] J. Lunze. „Schaltende PI-Regler“. In: *at – Automatisierungstechnik* 63(12). S. 941–952. De Gruyter Oldenbourg. 2015.

[12] H. Lutz, W. Wendt. „Taschenbuch der Regelungstechnik“. Europa-Lehrmittel. 2014.

[13] W. Oppelt. „Kleines Handbuch technischer Regelvorgänge“. Weinheim / Bergstr.: Verlag Chemie. 1972.

[14] R. Ibrahim. „Liquid Sloshing Dynamics“. University Press, Cambridge. 2005.

[15] D. Schröder. „Elektrische Antriebe – Regelung von Antriebssystemen“. Springer Vieweg. 2015.

[16] W. Singhose, L. Pao „A comparison of input shaping and time-optimal flexible-body control“. In: *Control Engineering Practice*, Volume 5, Issue 4, S. 459-467. 1997.

[17] VDI-Richtlinien „Bewegungsgesetze für Kurvengetriebe - theoretische Grundlagen, VDI 2143 Blatt 1“. VDI. 1980.

Surrogate-Assisted Learning of Neural Networks

Jörg Stork, Martin Zaefferer, Andreas Fischbach, Frederik Rehbach, Thomas Bartz-Beielstein

SPOTSeven Labs, TH Köln
Steinmüllerallee 1, 51643 Gummersbach
E-Mail: firstname.lastname@th-koeln.de

1 Introduction

Surrogate-assisted optimization has proven to be very successful if applied to industrial problems. The use of a data-driven surrogate model of an objective function during an optimization cycle has many benefits, such as being cheap to evaluate and further providing both information about the objective landscape and the parameter space. In preliminary work, it was researched how surrogate-assisted optimization can help to optimize the structure of a *neural network* (NN) controller [7]. In this work, we will focus on how surrogates can help to improve the direct learning process of a transparent feed-forward neural network controller. As an initial case study we will consider a manageable real-world control task: the *elevator supervisory group problem* (ESGC) using a simplified simulation model [3]. We use this model as a benchmark which should indicate the applicability and performance of surrogate-assisted optimization to this kind of tasks. While the optimization process itself is in this case not considered expensive, the results show that surrogate-assisted optimization is capable of outperforming metaheuristic optimization methods for a low number of evaluations. Further the surrogate can be used for significance analysis of the inputs and weighted connections to further exploit problem information.

2 Motivation

Recent advancements in robotics and control have shown, that methods from the field of computational intelligence are becoming more and more significant. Robot control policies are no longer just trained by machine learning algorithms. Rather, robots learn how to solve a certain task

by themselves, e.g., by methods of evolutionary robotics [4]. In an real world environment evolutionary learning of control policies can be costly, as the fitness of a certain robot action can only be evaluated after a sequence of time steps, which can easily be in minutes or hours. Thus, these learning processes pose a difficult optimization problem and standard learning methods are not suitable for the time requirements of these tasks. Neural networks are a well established type of controller in evolutionary robotics. Here, the set of coefficients and the topology of the network need to be optimized for optimal performance. More recent and sophisticated approaches for developing and learning of controllers, such as *neuroevolution of augmenting topologies* [18] were invented to handle these optimization processes, but they still need many evaluations to adapt the neural networks.

- Our hypothesis is that assisting this learning process by means of surrogate-assisted optimization, which has proven to be able to perform significantly well in expensive industrial optimization tasks [14, 15], should be beneficial.
- As a second hypothesis, we assume that these surrogate models can help to retrieve additional useful information about the objective function, e.g., importance of certain inputs.

We want to test this hypothesis based on experiments with a small real-world task simulator, which is implemented as a simple neural network and not expensive to evaluate. The results can be transferred to more sophisticated tasks, like a real world learning process. The results should indicate the applicability and basic performance of surrogate-assisted optimization methods in comparison to state-of-the-art optimization algorithms. The variable importance information provided by the surrogate models is also analyzed with regard to their usefulness. For instance, variable importance could be helpful to identify especially important or defective inputs sensors of a physical controller, e.g., in an evolutionary robotics task.

3 The Elevator Supervisory Group Problem

3.1 General Description

Today, elevator systems are present everywhere in urban areas. They need to be optimized to achieve the desired service quality in terms of

waiting time for the customers, as well as in terms of energy efficiency. They are controlled by an elevator group controller, which assigns the elevator cars to certain floors and destinations on basis of the customer service calls. The ESGC problem as introduced by [3] is a so-called destination call system, where the customer can choose their desired destination on the floor level outside the elevator cars. In the introduced problem instance, the controller is implemented as a sophisticated *neural network* NN, where the specific structure and weights depict a certain control strategy. The optimization of these weights imposes a set of challenges, which render this task highly complex:

- The topology of the fitness function is to a high extent non-linear as well as multi-modal.
- The traffic load is dynamic and stochastic, as customers do not arrive in a deterministic manner.
- Gradient-based methods cannot be applied successfully to this optimization problem.
- The simulator is computational expensive, which limits the number of function evaluations.

As consequence of the complexity of such simulators [3] introduced a simplified validation model of an ESGC system, the *sequential ring*(S-Ring).

3.2 S-Ring Perceptron Simulator

The S-Ring was introduced to benchmark different ESGC algorithms independent of a certain elevator/floor configuration. It uses a simplified NN to control the elevators, where the connection weights can be modified and represent the variables of an optimization problem. Each weight setting will result in a certain control strategy which is tested on simulations of different traffic situations. The S-Ring has low computational costs, which allows us to use an ESGC instance as a benchmark for a large variety of algorithms. Using different traffic situations will lead to a fitness function which is subject to noise. The S-Ring optimization problem can be defined as follows [3]:

$$F(n, m, p, \vec{w}) = E\left(\sum_{i}^{t} \vec{c}_i\right) \tag{1}$$

where n is the number of elevators, m the number of floors, p the probability of an arriving customer per floor and $\vec{w}$ the NN weight vector, which depicts the control policy. This objective function evaluates the average waiting time of all customers $\vec{c}_i$ during a simulated traffic situation with t steps. For a given set of n, m, p the performance is only influenced by the weight vector of the NN controller. Thus, the simplified problem, as further used during this paper, can be written as $F = F(\vec{w})$. The parameters n, m, p were set as follows: Table 1 also displays the number of time steps for a single simulation run, which was set rather high to simulate a longer period. For each simulation run, the exact same period was used, resembling a certain fixed time-frame, e.g. a certain day in a year. By choosing a fixed time frame, we removed the noise of the problem, which renders the problem simpler to optimize. Moreover, the problem was adapted by setting the desired customer service quality of the ground floor to a high priority, while the second floor was set to a lower priority. This should simulate a typical real world hotel scenario, where it is wanted that arriving customers in the lobby are fast served. The second floor displays an internal service area, which is of low priority for the quality of service. As a side effect, this reduces the dimensionality of the optimization problem from 12 to 10.

Table 1: S-Ring Configuration

nFloors	nElevators	probNewCustomer	nIterations
6	2	0.3	10000

4 Methods for Learning of Neural Networks

A standard method for learning NN controller is *backpropagation*. Backpropagation optimizes the weights by utilizing a set of training data that contains input values with corresponding outputs. In case of the S-ring optimization, our task is not machine learning, but to find the (single) global optimum for the given fitness topology of a time dependent simulation problem. We receive a fitness value only after evaluating the weights in a designated simulation run. This means, there is no clear mapping from input to output data, as the output only defines a certain control policy and the final action changes dynamically in every time step.

Thus we will need to use more sophisticated methods: metaheuristic optimization and surrogate-assisted optimization.

4.1 Metaheuristic Optimization

Metaheuristics are sophisticated heuristics, which are often inspired by nature. They utilize stochastic processes (randomization) and usually do not require any gradient information. Metaheuristics are known to be general solvers which apply to a large variety of *global* problems without needing a priori information. They are suitable for highly non-linear and multi-modal problems, as well as so-called *black-box* problems, where no information about the topology of the objective function is known. No algorithm is able to deliver their best performance for every problem without adapting their control parameters; By parameter tuning [2, 6], we can exploit beneficial parameter settings, but it is very time-demanding. To provide reliable results without putting a lot of effort into algorithm tuning, we selected four different state-of-the-art R implementations of common metaheuristics from the range of *simulated annealing* methods and *evolutionary algorithms* for our comparison. Simulated annealing [9] is inspired by annealing processes in metallurgy, where materials are heated and cooled to change their physical structure. Simulated annealing follows the base principle of an greedy stochastic algorithm, but implements a control strategy which allows to accept also solutions with lesser fitness. This allows to escape local optima and establishes a global search strategy. Evolutionary algorithms [1] are based on the principles of natural selection: in each generation, a population of individuals (e.g. solutions $\vec{w}$) is evolved by mutation, recombination and selection steps. The selected packages are *DEoptim*, *GA*, *GenSA* and *genoud*. DEoptim and GA were chosen due to personal preference, while the two latter were chosen based on the survey on *Continuous Global Optimization in R* by Mullen [12]. GenSA and genoud performed best on a set of different optimization problems.

- **DEoptim** [13] is an R-implementation of the differential evolution algorithm [19], which belongs to the class of evolutionary algorithms. It is designed for global optimization using real vectors.
- **GA** [16] is a package which implements an genetic algorithm an allows optimization of real and integer problems.
- **GenSA** provides a version of generalized simulated annealing [20].

- **rgenoud** [11, 17] is an R-package which provides and implementation of a so-called *hybrid* algorithm. This algorithm combines evolutionary algorithms with the derivative-based quasi-Newton method *Broyden-Fletcher-Goldfarb-Shanno* (BFGS).

4.2 Surrogate-Assisted Optimization

Surrogate-assisted optimization algorithms employ data driven models to lighten the burden of expensive objective function evaluations. One framework for surrogate-assisted optimization is Sequential Parameter Optimization (SPO) [2]. SPO provides a flexible framework that employs methods from the fields of design of experiment, optimization, and statistics. In essence, SPO starts by generating an initial design of experiment, then builds a surrogate model (e.g., a linear model or Kriging). Then, the surrogate model is optimized to suggest a promising candidate solution, which is afterwards evaluated by the expensive objective function. These last steps (model building, optimization and evaluation) are iterated until some budget of evaluations is exhausted. Figure 1 shows the optimization cycle for the underlying ESGC problem. The experiments make use of SPOT, the R implementation of SPO. For this study, we have chosen to investigate three different surrogate models within the SPO framework.

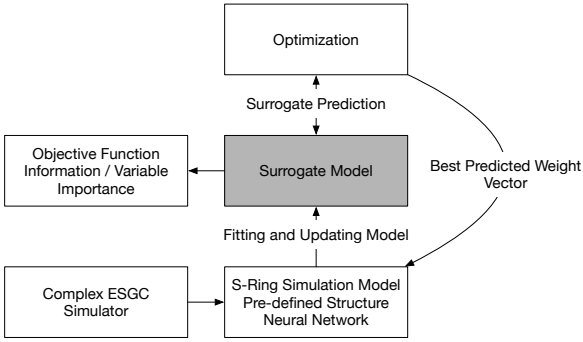


Figure 1: Surrogate-Assisted Optimization Cycle. The ESGC Simulator is approximated by the S-Ring simulator. The fitness topology is fitted by the surrogate on basis of the initial design and sequential updates. The sequential weight vectors are computed by an optimization of the surrogate.

- **Second order model with step-wise regression:** Firstly, we build second order linear regression models. The model is first build with all first order effects, quadratic effects as well as second order interactions. E.g., for two parameters x_1 and x_2 a model of the form $y(x) = c_1x_1 + c_2x_2 + c_3x_1^2 + c_4x_2^2 + c_5x_1x_2$ is determined. This full model is further refined by backwards, stepwise variable selection based on the Akaike information criterion. The stepwise variable selection is skipped whenever the data size is insufficient. While the resulting models are comparatively simple, one advantage is the comparatively low computational effort.
- **Random Forest:** Secondly, we use a Random Forest [5] model. Random Forests are ensembles of decision trees. We use the default settings of the randomForest R-package [10]. Random Forests are able to learn non-linear dependencies in the data, are typically numerically robust and fast to compute, and can handle discrete input variables.
- **Kriging:** Thirdly, a Kriging model (also known as Gaussian process regression) is employed. Kriging assumes that the observed data is the result of a stochastic process. We use an implementation from the R-package SPOT. The implementation is loosely based on Matlab code by Forrester et al. [8]. The correlation of samples is modeled via an exponential correlation function $\mathrm{cor}(x, x') = \exp(-\sum_{i=1}^{n} \theta_i |x_i - x'_i|^{p_i})$. The vectors x and x' are samples, or candidate solutions of the optimization problem. The parameters $\theta_i > 0$ and $1 \leq p_i \leq 2$ are determined by maximum likelihood estimation. Forrester et al. provide a detailed and easy to follow description of Kriging and related methods [8]. Of the three models, Kriging requires the largest computational effort, yet produces the potentially most accurate model.

4.3 Variable importance

Most black-box optimizers tend to deliver only the best found parameter settings found within the available budget of objective function evaluations. Hence, these optimizers do no provide any information on what they have learned about the importance of the input variables during the optimization process. An advantage of the surrogate model techniques applied in this study is the provision of some knowledge beyond the best found parameter setting. For example, the estimated model coefficients or

the change of the prediction accuracy (i.e. the model error) by permuting variable values may provide strong indicators for the importance of each input variable.

- **Linear Regression Models:** In linear regression models, p-values can be taken to analyze the significance of a variable. However, statistical significance does not automatically mean a large impact of the variable on the result. The values of the regression coefficients can be compared instead. Larger coefficient values account for larger impact of the corresponding variable for the outcome. This has to be used carefully, especially if the scales of variables differ. The data must be standardized first, to enable a safe comparison of the regression coefficients to judge the variables importance.
- **Random Forest:** Variable importance can be estimated by computing the out-of-bag error first. Afterwards a permutation of the values of one variable is computed randomly. This keeps the distribution of the values equal. In the next step the out-of-bag error is computed again and the result compared to the initial out-of-bag error. If the error (e.g., mean squared error) varies a lot, the variable can be seen as important. With this process, a ranking of the importance of the variables can be computed.
- **Kriging:** The width parameter θ_i determines how far the influence of each sample point spreads in dimension i. In detail, the larger the width parameter is, the faster are the potential changes in the predicted value. The smaller the width parameter is, the slower are the potential changes in the prediction. The descending order of the θ values give an indicator of the variable importance.

5 Experiments

Our stated hypothesis is tested by performing a benchmark on the S-Ring problem. The performance of a *random search* algorithm will be added as a baseline comparison. All four metaheuristic algorithms and further the surrogate-assisted optimization with three different models are compared on basis of their total objective function evaluations to simulate the performance on a possible expensive function. Thus, the smallest number of evaluations is set to 100, which is a common limit for optimization of expensive objective functions. The maximum number of tested evaluations for the metaheuristics is set to 1_{e+5} to see the

convergence behavior on the objective function. The maximum number of surrogate-assisted optimization runs are limited to 1000 total objective function evaluations and 10 percent of this budget are used for the initial *latin hypercube sampling*. A single optimization iteration can, due to the model fitting, model optimization and prediction, become very computationally expensive for large sample sizes. As all tested algorithms are stochastic, each experiment is repeated 20 times. As previously mentioned, for all tested algorithms the parameter settings, beside the iterations and populations size to set an exact number of evaluations, are not changed and use the pre-set default settings. The experimental setup is summarized in table 2.

Table 2: Experimental Setup

Algorithm	No. Evaluations	popSize	maxIter
RandomSearch	100	100	1
RandomSearch	$1_{e+3}, 1_{e+5}$	$1_{e+3}, 1_{e+5}$	1
Metaheuristics:			
GenSA	$100, 200, 500$	/	/
GenSA	$1_{e+3}, 1_{e+5}$	/	/
DEoptim	$100, 200, 500$	5,5,10	9,19,24
DEoptim	$1_{e+3}, 1_{e+5}$	10,100	49,499
GA	$100, 200, 500$	10,10,20	10,20,25
GA	$1_{e+3}, 1_{e+5}$	50,50	$20,2_{e+3}$
genoud	$100, 200, 500$	10,10,20	10,20,25
genoud	$1_{e+3}, 1_{e+5}$	$50,1_{e+3}$	20,100
Surrogates: Model	No. Evaluations	initDesign	Optimizer
SecondOrderLM	$100, 200, 500, 1_{e+3}$	10,20,50,100	DEoptim
RandomForest	$100, 200, 500, 1_{e+3}$	10,20,50,100	DEoptim
Kriging	$100, 200, 500, 1_{e+3}$	10,20,50,100	DEoptim

6 Results and Discussion

6.1 Benchmark Comparison

Figure 2 shows the benchmark results for the different combinations of algorithms and number of evaluations:

- **Metaheuristics:** For 100 evaluations, *GA* and *genoud* perform better than random search, while *DEoptim* and *GenSA* show inferior results. The methods significantly improve with a rising number of evaluations and are able to outperform random search. For instance, for 1000 evaluations, most method (except GenSA) perform better than the random search method, which is also the case for 1_{e+5} evaluations. The long run results (1_{e+5}) also indicate that *GenSA* and *DEoptim* seem to converge to a global optimum, while *genoud* and *GA* show significantly inferior results. A large number of evaluations seems to benefit *GenSA* the most, as with less evaluations it is in all cases outperformed by the other algorithms. This can be strongly connected to the chosen default parameter settings.
- **Surrogate-Assisted Optimization:** For 100 evaluations, the surrogate-assisted optimization outperforms the metaheuristics significantly. Particular *Kriging* is performing on a level equivalent to this of 1000 random search evaluations and near 500 metaheuristic evaluations. This picture becomes even clearer for 200 evaluations, where *Kriging* again improves and surpasses random search with 1000 evaluations. For 1000 evaluations, *Kriging* reaches the level of 1_{e+5} random search evaluations and thus outperforms all metaheuristic algorithms on this level. The *second-order linear model* also performs well, but is not able to show considerable improvements in comparison to the metaheuristics.*Random forest* shows poor results for larger evaluations sizes, where it is inferior to random search.

The good results of the *Kriging* surrogates can be explained by their strong ability of fitting non-linear landscapes and their general good interpolation ability. While the *second-order linear model* can fit non-linear behavior so a certain extend, they are not fit to build global surrogates of highly multi-modal landscapes. The poor performance of *random forest* is due to the pure continuous nature of the problem and the bad interpolation abilities of these models.

Table 3: Variable importance

Weight	RF Mean MSE Decrease	Kriging theta values
1	117.31366	3.82746
2	9.83143	1e-04
3	59.79332	1.15356
4	17.86245	0.4643613
5	11.06656	0.2711343
6	14.05440	0.001178616
7	64.32869	2.221088
8	64.86968	1.164819
9	34.63966	1.8983
10	47.50249	3.45285

6.2 Model Variable Importance

To check the usefulness of the variable importance, we used three of the best models, which were fitted with 1000 evaluations.

- The *second-order linear model* has a multiple R-squared of 0.869 and an adjusted R-squared 0.8611, thus it is able to explain a high extent of the underlying variance. It contains a large number of 60 model terms, including main effects, interactions and quadratic effects, where the p-values indicate a high importance. Due to this high number of terms, an additional analysis is needed to extract the importance variables. At this point, as we are not able to validate the results in terms of their usefulness, we decided not to conduct any further analysis.
- The random forest model has a mean of squared residuals of 0.129 and explains 71.44 percent of the variance. The mean decrease in MSE is shown in table 3 and compared to the results of *Kriging*.
- The estimated activity parameters (theta) are shown in table 3.

Table 3 indicates, that at least the most important weight (No. 1) and the least important weight (No. 2) are the same for both the *Kriging* and *random forest* model. The other weights also show some correlation. The variable importance comparison shows, that the models are able to extract knowledge beyond the best found parameter setting. To validate the given results, we will need to use a designated experimental design, which will be part of future research.

Table 4: Algorithm Computation Time. All values are approximated.

Algorithm	No. Evaluations	Computation Time
S-Ring Problem C	1	< 0.001 seconds
S-Ring Problem R	1	< 1 second
Metaheuristics	100	0.1 second
Metaheuristics	1000	1 second
Metaheuristics	1_{e+5}	1-2 minutes
surrRF	100	1 minute
surrSO	100	4 minutes
surrKR	100	8 minutes
surrRF	1000	1 hour
surrSO	1000	4 hours
surrKR	1000	> 1 day

6.3 Computation Time Comparison

An important aspect of every optimization technique is the total computation time. Table 4 shows approximated values for the metaheuristics and the surrogate-assisted optimization with the respective models. As the problem itself has nearly no computation time, the indicated values are mainly caused by the optimization algorithms. As the values indicate, surrogate-assisted optimization is in comparison very expensive. The model fitting, updating and optimization process is computationally expensive, particularly for a higher number of samples. This is especially visible for Kriging, which is very sensitive to higher sample sizes. At this point, we also have to consider that the *SPOT* implementation is a R-framework, which is in terms of computation time much inferior to C or C++ based implementations. For instance, a re-implementation of the S-Ring simulator in R, which is normally implemented in C, is about 1000 times slower. We can assume, that an optimized version would be significant faster. Moreover, *SPO* performs sequential optimization, while the metaheuristics are able to conduct parallel evaluations.

7 Conclusion

In accordance to our hypotheses, the results show that surrogate-assisted optimization is a beneficial approach for the underlying NN control optimization task. The tested algorithms were capable of outperforming metaheuristic optimization methods. Furthermore, the surrogate can be used for significance analysis of the inputs and weighted connections to further exploit problem information. We can thus assume that surrogate-assisted optimization is able to provide a greater understanding of the learning process. The clear downside of the surrogate-assisted optimization is the large computation time, which is more than 10000 times larger than these of metaheuristic optimization. However, this huge downside becomes less significant in scenarios where the objective function evaluations become very expensive, e.g., in the area of several minutes. The model fitting and optimization process could be conducted simultaneously to the real-time evaluations. Also, not considered here are optimized and parallel surrogate-assisted approaches, which could considerable improve the computation time. In this study, we used the default parameters for all given algorithms, whereby no extensive research was made to optimize the *SPOT* default parameters, while the metaheuristic implementations default parameters are commonly optimized or include self-adaptive procedures to show comparable results. An extended study to identify generally good settings for large set of problems could be helpful to further improve general performance. In future research, we will test the applicability to a larger range of difficult problems from the area of artificial intelligence and evolutionary robotics. Moreover, we will study the usefulness of the extracted variable importance for evolutionary robotics.

Acknowledgements

This work is part of a project that has received funding from the European Unions Horizon 2020 research and innovation program under grant agreement no. 692286.

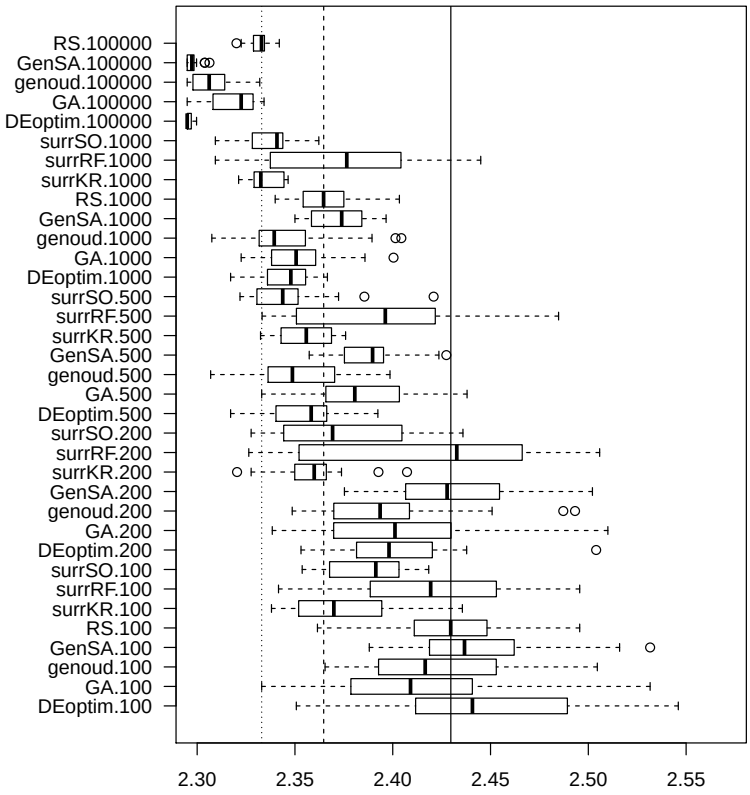


Figure 2: S-Ring Simulator Benchmark Results. The algorithms with their respective number of evaluations are shown on the y-axis, the x-axis shows the achieved fitness. SurrSO uses the second order model, SurrRF the random forest model and SurrKR the Kriging model. The vertical lines represent the baseline, where the solid line is the median random search fitness for 100, the dashed line for 1000 and the pointed line for 100000 evaluations

References

[1] T. Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[2] T. Bartz-Beielstein, C. W. Lasarczyk, and M. Preuß. Sequential parameter optimization. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 773–780. IEEE, 2005.

[3] T. Bartz-Beielstein, M. Preuss, and S. Markon. Validation and optimization of an elevator simulation model with modern search heuristics. *Metaheuristics: Progress as Real Problem Solvers*, pages 109–128, 2005.

[4] J. C. Bongard. Evolutionary robotics. *Communications of the ACM*, 56(8):74–83, 2013.

[5] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[6] Á. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on evolutionary computation*, 3(2):124–141, 1999.

[7] O. Flasch, T. Bartz-Beielstein, A. Davtyan, P. Koch, W. Konen, T. D. Oyetoyan, and M. Tamutan. Comparing ci methods for prediction models in environmental engineering. In *Proc. of CEC*, 2010.

[8] A. Forrester, A. Sobester, and A. Keane. *Engineering Design via Surrogate Modelling*. Wiley, 2008.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[10] A. Liaw and M. Wiener. Classification and Regression by randomForest. *R News*, 2(3):18–22, 2002.

[11] W. R. Mebane Jr and J. S. Sekhon. Genetic optimization using derivatives: the rgenoud package for r. *Journal of Statistical Software*, 42(11):1–26, 2011.

[12] K. M. Mullen. Continuous global optimization in r. *Journal of Statistical Software*, 60(6):1–45, 2014.

[13] K. M. Mullen, D. Ardia, D. L. Gil, D. Windover, and J. Cline. Deoptim: An r package for global optimization by differential evolution. 2009.

[14] Y. S. Ong, P. B. Nair, and A. J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA journal*, 41(4):687–696, 2003.

[15] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker. Surrogate-based analysis and optimization. *Progress in aerospace sciences*, 41(1):1–28, 2005.

[16] L. Scrucca. Ga: a package for genetic algorithms in r. *Journal of Statistical Software*, 53(4):1–37, 2013.

[17] J. S. Sekhon and W. R. Mebane. Genetic optimization using derivatives. *Political Analysis*, 7:187–210, 1998.

[18] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[19] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[20] Y. Xiang, S. Gubian, B. Suomela, and J. Hoeng. Generalized simulated annealing for global optimization: The gensa package. *R Journal*, 5(1), 2013.

Design of Experiments – Combining Linear and Nonlinear Inputs

Tim Oliver Heinz, Julian Belz, Oliver Nelles

Mess- und Regelungstechnik – Mechatronik, Universität Siegen
Paul-Bonatz-Straße 9-11, D-57068 Siegen
E-Mail: tim.heinz@uni-siegen.de

Abstract

The complexity of real-world processes has increased in recent years. By modeling processes with many inputs, the *curse of dimensionality* appears. For most modeling approaches, this is a huge problem. By using local model networks (LMNs), the inputs of the process may be assigned to either the rule premises ($\underline{z}$-inputs), the rule consequences ($\underline{x}$-inputs) or both. This contribution focuses on experimental designs that fully exploit the possibility of LMNs to distinguish between the x- and z-input space. We propose to create experimental designs that are space-filling in the z-input space, while the values of the x-inputs are chosen to fulfill a user-specified optimality criterion. In addition to a Fisher-information-matrix-based optimality criterion a loss function based on geometric considerations is used. The proposed approach is superior to space-filling designs for all inputs. A comparison to a Fisher information based optimization of the $\underline{x}$-input values and a space-filling design for all inputs shows the properties of the new approach.

1 Introduction

Measurements play the key role in experimental modeling. The quality of the data used to build models restricts the generalization performance that can be achieved. The goal of design of experiments (DoE) techniques is to use available resources in the most efficient way [7]. The DoE specifies at which locations in the input space data should be gathered. Loosely speaking, these locations are chosen to yield the most informative data.

In this contribution we focus on experimental designs that are customized for so-called local model networks (LMNs). LMNs follow a divide-and-conquer strategy by partitioning the input space into several subregions and estimating rather simple local models that are only valid within these subregions [17]. As a result of this model structure it is possible to distinguish between two types of variables since the input space partitioning might depend on other variables than the local models. For example, a process might well be described by a dynamic linear model as long as there are only moderate changes in temperature. If the temperature change becomes too big, time constants and/or gains of the true system might change. In this example the temperature acts as a scheduling variable and would be responsible for the input space partitioning while the local models might not explicitly depend on the temperature. Typically the local models are chosen to be of polynomial type whereas functions describing the partitioning are more complex and nonlinear in the parameters [18].

The DoE strategy that is proposed here exploits the different characteristics of the two types of input variables. Optimal parameters for the partitioning of an LMN are usually obtained by nonlinear optimization techniques or clustering algorithms, but the shapes and sizes of the corresponding subregions are unknown a-priori. Therefore it is suggested to choose a space-filling experimental design for the variables responsible for the input space partitioning, which are named $\underline{z}$-inputs in the following. For the variables on which the local models depend, called $\underline{x}$-inputs for short, we propose to exploit the a-priori known local model structure. If for example the local models are chosen to be polynomials of degree one, optimal experimental designs can be generated by optimizing some summary statistic of the Fisher information matrix (FIM) [6]. For example, maximizing the determinant of the FIM (D-optiomal design), maximizing the trace of the FIM (A-optimal design), or maximizing the smallest eigenvalue of the FIM (E-optimal design) [6].

One drawback of this proposal is the necessity for prior knowledge about the system under investigation. This information might come from experts or from previous investigations on similar systems. Automatic procedures exist that are able to unveil the necessary information based on measured data (from previous investigations) as described in [2, 3]. The main advantage of the proposed DoE strategy highly depends on the number of $\underline{z}$-inputs, spanning the $\underline{z}$-input space. The lower the dimensionality of the $\underline{z}$-input space is, the greater the advantage should

be. Far less measurements are necessary to realize a space-filling design in lower dimensional input spaces. Once the space-filling design for the $\underline{z}$-input space is generated the values for the remaining $\underline{x}$-inputs can be determined subsequently, but before measurements are actually carried out.

There is some literature that already suggested experimental designs specifically generated for the properties of LMNs. For example, Kroll et al. [13, 4, 12] proposed sequential designs for this specific model type. The idea is to identify a proper partitioning of the LMN in a first step and to determine additional measurements afterwards considering the structure of the local models such that optimal experimental designs arise. For this proposal measurements have to be carried out before the DoE for the local models can be generated whereas in our proposed DoE strategy the whole experimental design is set up in advance to any measurements. Additionally, the possibility to distinguish between the $\underline{x}$-and $\underline{z}$-inputs is not exploited. The same differences apply for the publication of Hartmann et al. [9]. They use LMNs to adaptively add measurements to the experimental design in order to improve the model's generalization performance in a very efficient way. Hametner et al. [8] generate D-optimal designs for LMNs based on the FIM. However, this approach does also not exploit the separability between the $\underline{x}$-and $\underline{z}$-inputs as it is done in our approach.

Section 2 describes some fundamentals about DoE in general and about the used space-filling and optimal experimental designs. Section 3 provides more detailed information about LMNs before Section 4 presents the proposed experimental designs that take account of the two different LMN input variables as described previously. Results of some case studies are presented in Section 4 and Section 5 concludes the paper.

2 Design of Experiments

The quality of a black-box model depends strongly on the data used for the identification. Thus, the DoE is most important. Without any prior knowledge of the process space-filling designs are preferable to gather all nonlinear effects of the process. If information about the process structure is available, space-filling designs may be suboptimal. In this case an optimized design according to the structure is preferable.

2.1 space-Filling Designs

Basically there are two desirable properties for the DoE, if no prior knowledge about the function or process of interest is available. The design should cover the whole input space and it should be non-collapsing [21]. Non-collapsing means, that if all data points are projected onto one axis, all values along that axis will only occur once. This property is especially appealing if the relevance of the input variables is unknown a-priori. In case some inputs are identified to be irrelevant, the values of each remaining relevant input are still uniformly spread across each single axis without repetitions.

2.1.1 Latin Hypercube

Latin hypercube (LH) designs were originally proposed for computer experiments by [15] and [10] and automatically fulfill the requirement of being non-collapsing. They are computationally cheap to construct, cover the design region without replications and pursue a geometrical approach to be build [20]. The number of samples N has to be determined a-priori and can be chosen arbitrarily. Once N is defined, each axis of the input space is partitioned into N levels. For n inputs a n-dimensional grid with $N \cdot n$ grid points emerges. Each level of each input is only occupied once, such that a non-collapsing design is guaranteed. The property of being an LH design does not automatically guarantee a space-filling data distribution. Therefore an optimization of the LH design has to be performed. In order to achieve good space filling properties, an optimization has to be performed. A coordinate exchange-based optimization algorithm to produce space-filling LH designs is presented in [5] and used throughout this work to generate the LH designs.

2.1.2 Sobol Sequences

Sobol sequences were introduced by Ilya M. Sobol [22]. These low-discrepancy sequences subdivide each dimension successively in halves and reorder the coordinates in each dimension. For $N = 2^x$ with $x \in \mathbb{N}$ the data points are equidistant distributed for each input, similar to LH designs. In contrast to LH designs, the calculation of a Sobol sequence is computationally cheap. But the quality w.r.t. the achievable model

quality may be worse compared to an optimized LH design due to the randomness incorporated in the Sobol sequence generation.

2.2 Optimal Designs

If the model structure is known a priori, the design may be optimized to a given quality function (see e.g. [14]). For polynomial models maximizing the determinant of the Fisher information matrix results in an D-optimal design. In case of affine regression problems the design points are optimized to the extreme values (minimum and maximum values) of each input dimension.

3 Local Model Networks

A local model network (LMN) can be divided in (i) the partitioning and (ii) the local models. In a fuzzy interpretation, the partitioning describes the rule premises (IF part) and the local models describe the rule consequents (THEN part). The input space is partitioned using validity functions $\Phi_i(\underline{z})$. The local models are described by $\hat{y}_i(\underline{x})$. The inputs $\underline{x}$ and $\underline{z}$ are subsets from all available process inputs. Each process input can be assigned to (i) $\underline{x}$, (ii) $\underline{z}$, (iii) $\underline{x}$ and $\underline{z}$, or (iv) can be completely omitted. This property is beneficial especially for processes with many inputs, where the operating point can be described by a few model inputs [1, 3]. The output of the LMN can be calculated as sum of M local models $\hat{y}_i$, weighted with their validity function Φ_i:

$$\hat{y} = \sum_{i=1}^{M} \hat{y}_i(\underline{x})\Phi_i(\underline{z})\,, \quad \sum_{i=1}^{M} \Phi_i(\underline{z}) = 1\,. \tag{1}$$

All n inputs $\underline{u} = [u_1, u_2, \ldots, u_n]$ can be assigned to either $\underline{x}$-and/or to $\underline{z}$. With n_x inputs in $\underline{x}$-and n_z inputs in $\underline{z}$. The structure of a LMN is presented in Fig.1.

The LMNs in this contribution are constructed using the hierarchical local model tree (HILOMOT) algorithm [19]. This incremental growing tree construction algorithm divides the input space with axes-oblique splits. All local models are chosen to be of affine type in this paper. The validity functions are generated by sigmoid splitting functions that are linked in a hierarchical, multiplicative way, see [19] for more details.

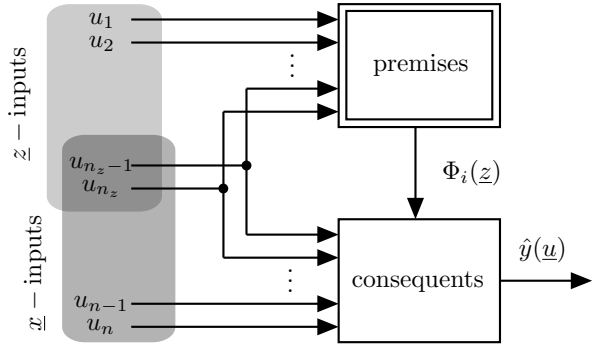


Figure 1: Block diagram of a local model network assigning the inputs to $\underline{x}$-inputs and $\underline{z}$-inputs.

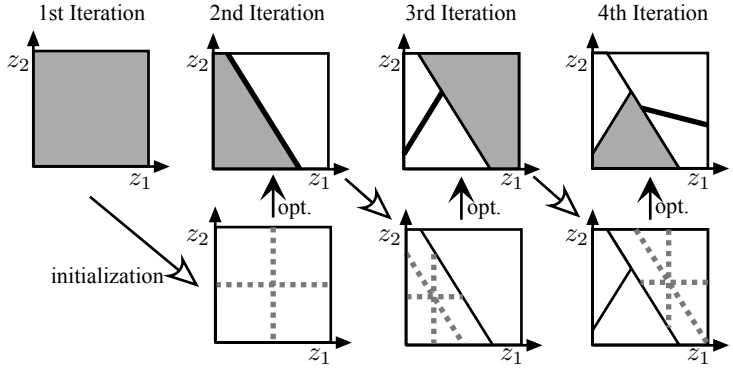


Figure 2: The first Iterations of HILOMOT using a two-dimensional $\underline{z}$-input space.

The procedure of the HILOMOT algorithm can be explained with the help of Fig. 2. Starting with a global affine model, in each iteration an additional local affine model is generated. The local model with the worst local error measure (gray areas in Fig. 2) is split into two submodels, such that the spatial resolution is adjusted in an adaptive way. The linear parameters of the new submodels are estimated locally by a weighted least squares method. This is computationally extremely cheap and introduces a regularization effect which increases the robustness against overfitting, as stated in [18]. The axes-oblique partitioning is achieved by optimizing the current split direction and position in each iteration. Only the new split is optimized, all already existing splits are kept unchanged. The

initial split direction for the optimization is either one of the orthogonal splits or the direction of the parent split (dotted lines in Fig. 2). In this contribution, the $\underline{x}$-input space consists of all process inputs $\underline{x} = \underline{u}$.

4 DoE for Nonlinear and Linear Inputs

By modeling real-world processes, the number of input variables which influence the output are plenty. In general, each input contributes more or less nonlinear to the process output thus a space-filling design for all inputs is desirable. With increasing number of input dimensions, the volume of the input space grows exponentially thus for fixed N, the data point density decays to zero. This is one manifestation of the *curse of dimensionality*. One way to weaken the *curse of dimensionality* is the separation of nonlinear and linear input variables. For LMNs this is possible by assigning an input to either the $\underline{x}$-and/or the $\underline{z}$-inputs. The number of inputs contained in the $\underline{z}$-input space can be significantly smaller compared with the $\underline{x}$-input space.

Since the partitioning of the LMN is performed in the $\underline{z}$-input space, this space needs a uniform data point coverage to unveil all nonlinearities. Since the structure of the local models is assumed to be known a-priori, the data points of the $\underline{x}$-input space can be optimized e.g. according to an optimality criterion. The matrix containing the locations of all data points in the $\underline{z}$-input space is defined as follows:

$$\underline{Z} = \begin{bmatrix} \underline{z}(1) \\ \underline{z}(2) \\ \vdots \\ \underline{z}(N) \end{bmatrix}, \quad \text{with} \tag{2}$$

$$\underline{z}(i) = [u_1(i), u_2(i), \ldots, u_{n_z-1}(i), u_{n_z}(i)] \; . \tag{3}$$

Most important for the identification of the nonlinear process is the data distribution of the $\underline{z}$-input space. Thus the data points in $\underline{Z}$ are chosen space-filling. The focus of this contribution is the determination of the best values of the data points in the *pure* $\underline{x}$-input space.

$$\underline{\xi} = \underline{X} \backslash \underline{Z} = \begin{bmatrix} u_{n_z+1}(1) & u_{n_z+2}(1) & \dots & u_n(1) \\ u_{n_z+1}(2) & u_{n_z+2}(2) & \dots & u_n(2) \\ \vdots & \vdots & \ddots & \vdots \\ u_{n_z+1}(N) & u_{n_z+2}(N) & \dots & u_n(N) \end{bmatrix} . \tag{4}$$

There are some remarks to this approach: (i) The order of the optimization (first $\underline{Z}$ and $\underline{\xi}$ afterwards) is arbitrarily chosen and in general the order can be switched. But in comparison with the $\underline{x}$-inputs, the data distribution of the $\underline{z}$-input space is more important. Therefore the optimization of the $\underline{z}$-inputs should be preferred. (ii) Variables contained in both the $\underline{x}$- and $\underline{z}$-input space are treated as if they only occur in the $\underline{z}$-input space. This seems to be arbitrary chosen. But for the determination of the nonlinearity in the $\underline{z}$-input space, a uniform data distribution along each axis is a necessity. For an affine local model the extreme values of the input are optimal w.r.t. the variance of the estimated parameters. However, having only extreme values of an input variable is somehow a worst-case scenario for the identification of nonlinear effects. Thus the treatment as $\underline{z}$-inputs of variables contained in both input spaces is justified.

4.1 Fisher Information Based Optimization

With a given $\underline{Z}$ Matrix, the remaining $\underline{\xi}$ data points may be optimized according to an optimality criterion. A common optimality criterion is the D-optimality. The regression matrix $\underline{\xi}_R$ is built according to the chosen local model type. In case of local affine models only $\underline{x}$-inputs and ones (for the offset) occur. For higher-degree polynomials the regressors have to be chosen accordingly. By maximizing

$$\det\left(\left[\underline{Z}\,\underline{\xi}_R\right]^T\left[\underline{Z}\,\underline{\xi}_R\right]\right), \quad \text{with:} \quad \left[\underline{Z}\,\underline{\xi}_R\right] \in \mathbb{R}^{N\times n} \tag{5}$$

a D-optimal design is achieved. This optimization is non-convex. To achieve reasonable optima, a coordinate exchange algorithm is common [16]. In this contribution, the `dcovary` function of the MATLAB Software is used to generate optimal designs.

4.2 Maximin Sparse Level Optimization

For local affine models, the best values for the $\underline{x}$-inputs w.r.t. the parameter estimation variance are the extreme values. The proposed approach places the values of $\underline{\xi}$ to their extreme values. In contrast to the optimal design based on the FIM, the data points are optimized according to the maximin distance [11] in the whole input space. This objective maximizes the nearest neighbor distance between all design points. This makes the data points uniformly spread across the whole input space. The data points in $\underline{\xi}$ are restricted to the extreme values thus the design is kind of *sparse* in $\underline{\xi}$. The optimized design is referred to as a maximin sparse level design (MSL design).

The optimization problem of a MSL design is non-convex as the optimization of an FIM based design, but the coordinate exchange algorithm as stated in [5] can be easily extended to optimize the values in $\underline{\xi}$. This algorithm is restricted to swap only the points in $\underline{\xi}$. But the quality function (nearest-neighbor distance) is calculated in the complete design $\left[\underline{Z}\,\underline{\xi}\right]$ (for affine local models, $\underline{\xi}$ and $\underline{\xi}_R$ differ only in a column filled with ones.). For a thorough discussion of the algorithm see [5].

5 Results

The DoE are analyzed (i) visually based on the data point distribution of a three dimensional input space and (ii) the achievable model quality is determined with artificial processes.

5.1 Data Point Distribution

For a DoE containing one nonlinear input and two linear inputs, the data along the nonlinear input axis are equidistant distributed. Along the remaining axes, the levels are at the extreme values. In Fig. 3, the data points for the $\underline{x}$-inputs are optimized according to *a*) D-optimal, *b*) MSL and, *c*) space-filling for all three inputs. It is obvious that the levels for *a*) and *b*) are optimized to the extreme values. As can be seen in Fig. 3 *b*) the data points Fig. 3are more evenly spread in the input space. Visually, this design is preferable over the classical D-optimal design. In *c*) the design is space-filling for all inputs. This is expected to be worst for processes with the given structure.

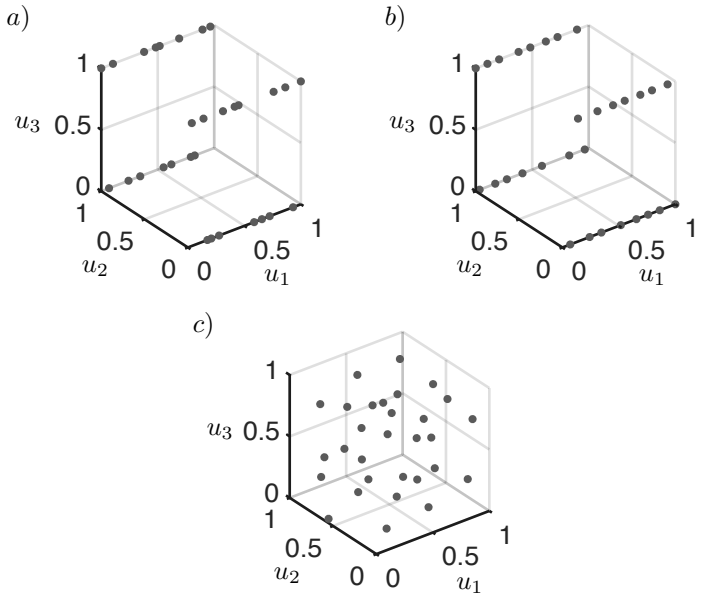


Figure 3: Input spaces for different DoE strategies with u_1 being nonlinear and u_2, u_3 being linear: *a*) D-optimal, *b*) MSL and, *c*) space-filling for all inputs.

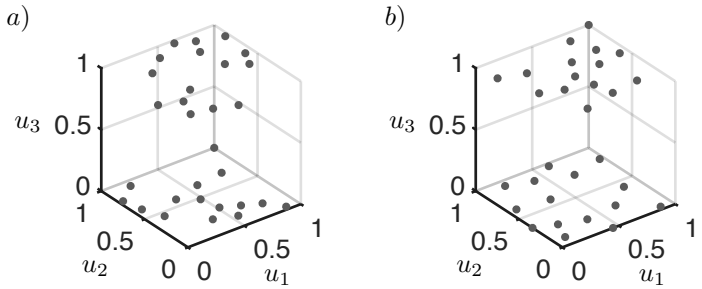


Figure 4: Input spaces for different DoE strategies with u_1, u_2 being nonlinear and u_3 being linear: *a*) D-optimal and, *b*) MSL.

For a DoE with two nonlinear inputs and one linear input the data points are scattered in two planes at the extreme values of u_3. For the space-filling design in u_1 and u_2 a LH optimized according to [5] is used. Figure 4 *a*) shows the D-optimal design with data points missing in the edges of the upper plane. The lower plane of *a*) shows similar data

holes. The data points of the MSL are more evenly spread, thus both planes (upper and lower) are comparable scattered with data points. A space-filling design is the same as in Fig. 3 *c*) and thus omitted in Fig. 4.

5.2 Achievable Model Quality

The performance of the proposed approach is underlined with two artificial processes with three inputs. The first process is only nonlinear in one of the three inputs and the remaining inputs are linear. The second process is nonlinear in two inputs with one linear input:

$$f_1(\underline{u}) = \frac{0.1}{u_1 + 0.1} + u_2 - u_3 + 1 \tag{6}$$

$$f_2(\underline{u}) = \frac{0.1}{u_1 + 0.1} - 4u_2 + 4u_2^2 + u_3 + 1 \tag{7}$$

For 50 different data set sizes in the range of $N = 10$ to $N = 300$ the designs are optimized. For each input data set size 100 output data sets are generated by using the functions above. To each data set a realization of white Gaussian noise with variance $\sigma = 0.05$ is added. Thus, for each design 5000 input-output data sets are generated. The model performance is achieved by an independent noise free test data set containing $N_{\text{test}} = 50\,000$ data points from a Sobol sequence.

5.2.1 Comparison with Space-Filling Designs

For the three designs (D-optimal, MSL and, LH), the 5000 test errors are calculated ($e_{\text{D-opt.}}$, e_{MSL} and, e_{LH}). To compare the optimized design with the space-filling design, the relative errors are used in Fig. 5 and Fig. 6. Values smaller than one indicate a better model performance of the optimized design. In Fig. 5 and Fig. 6 most of the relative errors are smaller than one, thus the optimized designs are superior to the space-filling LH design. The behavior is similar for the system with one nonlinear input (Fig. 5) and two nonlinear inputs (Fig. 6). The comparison of the optimized designs shows that an optimal design is always preferable over a space-filling design, if the process structure is known.

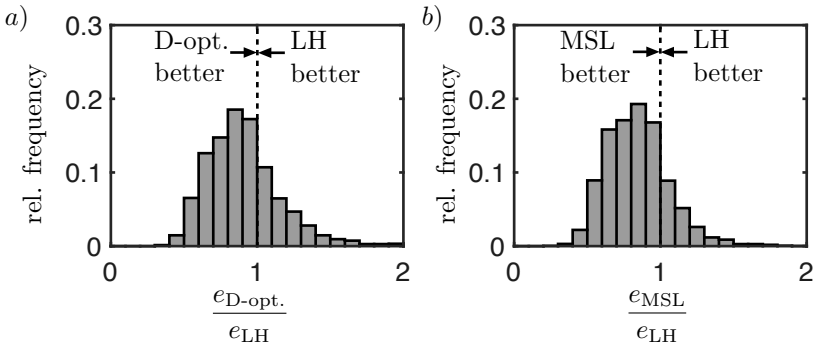


Figure 5: Relative test error of the model compared to the error achieved with a LH design (function f_1): *a*) D-optimal design and *b*) MSL design.

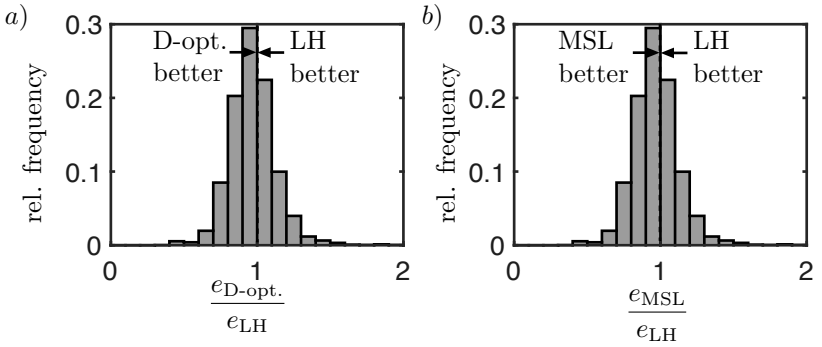


Figure 6: Relative test error of the model compared to the error achieved with a LH design (function f_2): *a*) D-optimal design and *b*) MSL design.

5.2.2 Comparing D-optimal and MSL Designs

In Fig. 7 the relative errors $e_{\text{D-opt.}}/e_{\text{MSL}}$ are shown. In *a*) the results for function f_1 are given. The distribution of the relative errors looks like a normal distribution with mean slightly greater than one. This indicates that the MSL is superior over the D-optimal design in many cases. In fact there are 62% of the MSL with a better model performance compared to the D-optimal design.

For function f_2 in *b*) the results are similar. The model performance based on a MSL design is in 61% of the test cases better compared to the D-optimal design.

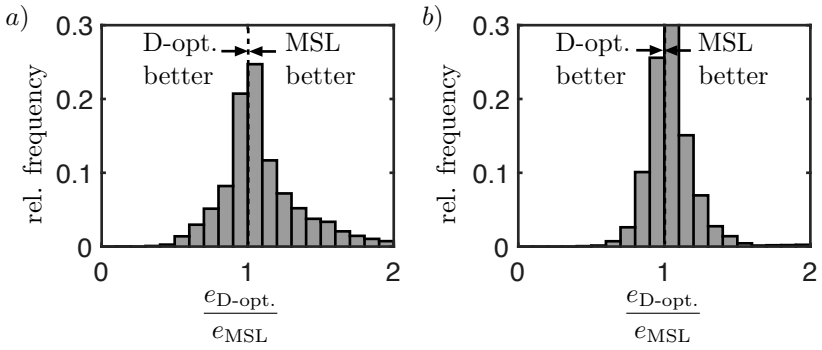


Figure 7: Relative test error of the model compared to the error achieved with a LH design (function f_2): *a*) D-optimal design and *b*) MSL design.

6 Conclusion and Outlook

This contribution focuses on experimental designs that fully exploit the possibility of local model networks (LMNs) to distinguish between the $\underline{x}$-(local models) and $\underline{z}$-input (partitioning) space. Knowing the structure of the local models that will be used together with the information about the process inputs contained in the $\underline{x}$-input space allows to create highly customized experimental designs for LMNs. In our proposal a space-filling design is created for the $\underline{z}$-input space. Afterwards the values of the remaining variables contained only in the $\underline{x}$-input space are optimized. Two different optimality criteria for the $\underline{x}$-inputs are used, namely the maximin sparse level design and a D-optimal design. Both alternatives are compared to each other and to an experimental design that is space-filling in the input space spanned by all process inputs. The comparison criterion is the achieved model performance of LMNs trained with the three different experimental designs. It turns out that the customized designs outperform the space-filling design. Among the customized designs the MSL design turns out to be superior compared to the D-optimal design.

In this contribution, the structure of the processes are given, thus the DoE matches perfectly to the given structure. For complex processes, the structure may not be given a priori. So the determination has to be performed with an input selection using an initial DoE for all inputs. The analysis of an optimized design based on an initial input selection is subject of future work.

References

[1] Julian Belz, Tim Oliver Heinz, and Oliver Nelles. Automated order determination strategies for nonlinear dynamic models. In *Computational Intelligence, 2015 IEEE Symposium Series on*. IEEE, 2016.

[2] Julian Belz and Oliver Nelles. Input selection using local model network trees. *IFAC Proceedings Volumes*, 47(3):4128–4133, 2014.

[3] Julian Belz, Oliver Nelles, Daniel Schwingshackl, Jakob Rehrl, and Martin Horn. Order Determination and Input Selection with Local Model Networks. In *Proceedings of the 20th IFAC World Congress*, page [accepted], Toulouse, France, July 2017.

[4] Axel Dürrbaum and Andreas Kroll. On robust experiment design for identifying locally affine takagi-sugeno models. In *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*, pages 001844–001849. IEEE, 2016.

[5] Tobias Ebert, Torsten Fischer, Julian Belz, Tim Oliver Heinz, Geritt Kampmann, and Oliver Nelles. Extended deterministic local search algorithm for maximin latin hypercube designs. In *Computational Intelligence, 2015 IEEE Symposium Series on*, pages 375–382. IEEE, 2015.

[6] Valerii V Fedorov and Peter Hackl. *Model-oriented design of experiments*, volume 125. Springer Science & Business Media, 2012.

[7] Valerii Vadimovich Fedorov. *Theory of optimal experiments*. Elsevier, 1972.

[8] Christoph Hametner, Markus Stadlbauer, Maxime Deregnaucourt, Stefan Jakubek, and Thomas Winsel. Optimal experiment design based on local model networks and multilayer perceptron networks. *Engineering Applications of Artificial Intelligence*, 26(1):251–261, 2013.

[9] Benjamin Hartmann and Oliver Nelles. Adaptive test planning for the calibration of combustion engines-methodology. *Design of experiments (DoE) in engine development*, pages 1–16, 2013.

[10] Ronald L Iman and WJ Conover. Small sample sensitivity analysis techniques for computer models. with an application to risk asses-

sment. *Communications in statistics-theory and methods*, 9(17):1749–1842, 1980.

[11] Mark E Johnson, Leslie M Moore, and Donald Ylvisaker. Minimax and maximin distance designs. *Journal of statistical planning and inference*, 26(2):131–148, 1990.

[12] A Kroll and A Dürrbaum. On joint optimal experiment design for identifying partition and local model parameters of takagi-sugeno models. *IFAC-PapersOnLine*, 48(28):1427–1432, 2015.

[13] A Kroll and A Dürrbaum. On optimal experiment design for identifying premise and conclusion parameters of takagi-sugeno models: Nonlinear regression case. *Applied Soft Computing*, 60:407–422, 2017.

[14] Andreas Kroll and Axel Dürrbaum. On joint optimal experiment design for identifying partition and local model parameters of takagi-sugeno models. In *Proceedings of the 17th IFAC Symposium on System Identification (SysID)*, pages 1427 – 1432, Beijing, China, 2015.

[15] Michael D McKay, Richard J Beckman, and William J Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[16] Ruth K Meyer and Christopher J Nachtsheim. The coordinate-exchange algorithm for constructing exact optimal experimental designs. *Technometrics*, 37(1):60–69, 1995.

[17] Roderick Murray-Smith. Local model networks and local learning. *Fuzzy Duisburg*, 94:404–409, 1994.

[18] Oliver Nelles. *Nonlinear system identification*. Springer, 2001.

[19] Oliver Nelles. Axes-oblique partitioning strategies for local model networks. In *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, pages 2378–2383. IEEE, 2006.

[20] Jeong-Soo Park. Optimal latin-hypercube designs for computer experiments. *Journal of statistical planning and inference*, 39(1):95–111, 1994.

[21] Thomas J Santner, Brian J Williams, and William I Notz. *The design and analysis of computer experiments*. Springer Science & Business Media, 2013.

[22] Ilya M Sobol. On quasi-monte carlo integrations. *Mathematics and Computers in Simulation*, 47(2):103–112, 1998.

Augmentations of the Bag of Visual Words Approach for Real-Time Fuzzy and Partial Image Classification

Andreas Bartschat[1], Johannes Stegmaier[1], Stephan Allgeier[1], Klaus-Martin Reichert[1], Sebastian Bohn[2], Oliver Stachs[2], Bernd Köhler[1], Ralf Mikut[1]

[1]Institute for Applied Computer Science, Karlsruhe Institute of Technology
[2] Department of Ophthalmology, University of Rostock E-Mail: andreas.bartschat@kit.edu

1 Introduction

Challenges in computer vision led to the development of automatic classification methods for images and objects. Advances of these methods deal with semantic segmentation and object detection for applications like image search, quality assurance or life sciences [7]. For objects with a clear shape and border, the expected result of a segmentation can be precisely defined. Thus, an even more challenging task is the segmentation of multiple classes with homogenous areas and smooth transitions. The result of these methods must be probabilistic or fuzzy and the creation of ground truth with fuzzy regions and transitions is time consuming. In the presence of large datasets and fast hardware, deep learning strategies show a superior performance in such tasks [6, 10]. The development of semantic scene classification with convolutional neural nets e.g. for autonomous driving applications outperforms other machine learning methods [9]. However, for online applications with real-time constraints or small datasets, approaches with hand-crafted feature extraction like the Bag of Visual Words (BoVW) method are still reasonable choices [13]. Especially, for tasks in controlled environments such as image-based quality management with a small number of working pieces or medical and biological imaging techniques with a small number of different tissues, these methods might still be preferable. In particular, if a huge amount of data such as 3D scans in life sciences must be processed, a compact and fast processing pipeline is crucial [14].

The traditional pipeline of the BoVW approach consists of a feature extraction to create the visual words and a quantization of these words into a histogram. Based on this histogram, the image is described and can be classified. The method proved to work well and even inspired the design of recurrent neural networks [12].

The aim of this work is to present augmentations of the BoVW method with probabilistic outputs and a sliding window approach to apply the classification as a segmentation method for homogeneous areas in images. Starting with the popular SURF descriptor, the influence of two clustering approaches to the classification accuracy are shown [4]. Additionally, two classification methods are investigated, including kernel based Support Vector Machines (SVM) and possibilities to augment the binary classification in order to get class probabilities and fuzzy classification results. Based on a medical image dataset of tissue layers in the human cornea, we show the possibilities of the proposed augmentations and their benefits.

2 Methods

2.1 Bag of Visual Words Approach

Given an image $\mathbf{I}$ the first step of the BoVW approach is the feature extraction. This can be done by an image patch description method like SURF, which extracts a feature vector $\mathbf{f}_{surf} \in \mathbb{R}^D$ from every sampling point. Especially, for use cases like object tracking or natural image registration either for panorama stitching or 3D scene reconstruction many patch description methods are available [3]. The selection of sampling points on the image can be done either by a keypoint detection method or by sampling from a uniform grid, which is usually referred to as dense sampling. Since the proposed methods focus on homogenous regions in an image with a regular structure and not on recognizable objects, the dense sampling method is used throughout this work.

Given a grid with S sampling points for each dimension, the resulting feature tensor for each image can be described as $\mathbf{F}_{samp}^{S \times S \times D}$. The next step is a dimension reduction of the feature space, usually performed by a clustering with K_c clusters. By representing each sampling point with the ID of its cluster, the feature tensor is turned into a matrix $\mathbf{F}_{cl}^{S \times S}$.

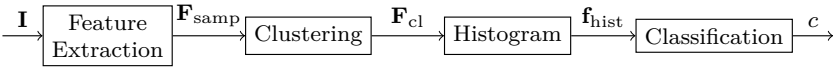


Figure 1: Processing steps of the BoVW approach

The final description of the image is then computed by a histogram over the present clusters in the image $\mathbf{f}_{hist} \in \mathbb{N}^{K_c}$. Using this vector, a classification method can be trained to predict the final class c of the image. A visualization of the whole pipeline is shown in Figure 1.

2.2 Clustering

A common clustering approach is the K-Means clustering, where K_c cluster centers are positioned such that the sum of distances of each feature point to its nearest center is minimized. This leads to spherical and uniformly sized clusters that are unable to model correlations in the feature space. Since the Fuzzy-C-Means approach has the same kind of drawback, if applied with Euclidean distance measures, it is not investigated. Instead, a mixture of multivariate normal distributions can be trained with the expectation maximization (EM) approach [5]. Based on the cluster-specific covariance matrices, it is able to model different cluster sizes as well as correlations in the feature dimensions.

2.3 Classification

Especially for normally distributed multi-class problems, the Naïve Bayes classifier is a solution, which can handle such problems out of the box and provides not only a classification result but also a probability. This is useful if a measure of certainty is required. Unfortunately, it does not perform well on the high dimensional feature space of the image description (e.g. $K_c = 100$).

A better choice in terms of high dimensional features is a Support Vector Machine (SVM) with radial basis functions [5]. Since this method is only capable of solving binary problems, multiple SVMs M_c are used and turned into a multi-class classifier with the one-against-all approach. Here, a single SVM is trained for each class with the goal to distinguish between the class and all other classes. The result m_c of a SVM classification is the distance of the data point to the decision boundary. For the binary decision, only the sign of the distance is interpreted, indicating

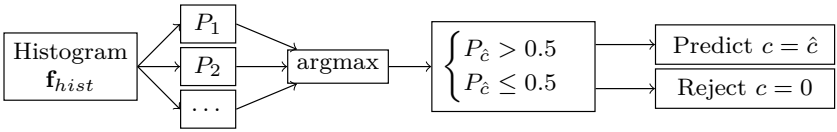


Figure 2: Fuzzy processing of the one-against-all classification with probabilistic SVMs.

the most likely class of the data point. Especially in the multi-class case an interpretation of this distance as a probability is useful, since it allows a better comparison of all SVM results. To deal with this problem, Platt proposed the post-processing of the distances with a logit function, where the function parameter A and B are trained with a maximum likelihood method on the results of the training set [11]:

$$P_c(\hat{y} = c) = \frac{1}{1 + e^{A_c \cdot m_c + B_c}} \tag{1}$$

Using this post-processing strategy, the result for each class $c = 1, 2, \ldots, C$ is a probability P_c. Given the vector $\mathbf{f}_{prob} = [P_1, P_2, \ldots, P_C]$, the argmax can be used to predict the most likely class:

$$\widehat{c} = \text{argmax}(\mathbf{f}_{prob}) \tag{2}$$

However, the results in $\mathbf{f}_{prob}$ do not necessarily sum up to one and are treatable as a fuzzy membership. If the result is required to be a probability distribution, a normalization like the softmax function can be used:

$$\widehat{P}_{norm,i} = \frac{e^{P_i}}{\sum_{j=1}^{C} e^{P_j}} \tag{3}$$

Using the probabilities without the softmax allows the definition of a rejection class $c = 0$. Since it is possible that all SVMs result in a probability lower than 0.5, the feature vector is not likely to belong to any class and can be rejected. Otherwise, the most likely class is predicted as shown in Figure 2.

2.4 Sliding Window

In order to get a pixel-wise classification of an image, it is possible to apply the BoVW method for the classification of regions in a sliding window approach. Here, the description is not computed over the whole

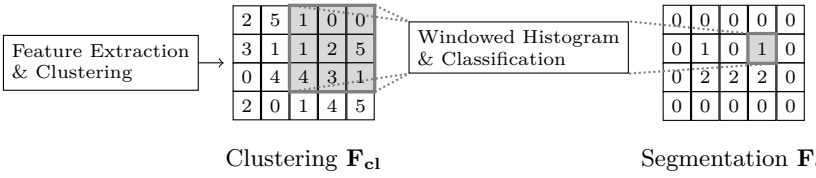


Figure 3: Sliding window approach of pixels with the classes $c = 0, 1, 2$ and a window size of $W = 3$.

image but only for a single pixel using a window of its neighborhood as shown in Figure 3. The matrix of the clustering result $\mathbf{F}_{cl}$ is used to create a histogram for each patch, which can be described by the matrix $\mathbf{F}_{hist}$. If each of these histograms is now processed with the classification method, a segmentation of the image $\mathbf{F}_c$ can be computed. However, if the BoVW method is trained in the sliding window approach, the amount of training data points for the SVM are increased by the number of patches extracted from the images. This is likely a factor of thousand to millions, making the training for the SVM tedious or even impossible. Thus, a batch learning approach is proposed. A further drawback of this method is the demand of pixel-wise labeled images.

A parameter introduced by this approach is the size of the window W. If this parameter defines a quadratic window, it can be described by its width, i.e. the number of covered sampling points. Therefore, the total number of covered sampling points of a window can be described by $S = W^2$. The window can then be applied between the clustering and the histogram computation shown in Figure 1 as visualized in Figure 3. Thus, the feature extraction and clustering can be used like before. Furthermore, by introducing a normalization step after the histogram creation, the final image description becomes independent of the size of the window. This makes it possible to train the BoVW method on one window size and apply it to another. The success of this approach depends on the application, but it can be used to reduce the labeling effort and amount of data points for the SVM training.

2.5 Batch Learning

The batch learning of the SVM is initialized with I_{init} randomly sampled data points from each class. Given the window size W every data point

is described by its neighborhood. With these image patches, a classifier is trained and applied to the whole training set. After the removal of the data points used for training, the data points of each class are sorted in ascending order, according to the result of the SVM classifier. Using an exponential distribution I_{add} data points for each class are sampled and added to the train set:

$$p(x) = \lambda \cdot e^{(-\lambda \cdot x)} \tag{4}$$

This sampling of the exponential distribution together with the learning can then be repeated until the cross-validation accuracy no longer improves or the maximum number of iterations is reached. Since the amount of available training data points reaches several thousands or millions, a small $\lambda < 0.01$ must be chosen. Otherwise, only the first few data points get a high probability, since the exponential function converges to fast to zero.

2.6 Cluster Interpretation

The clustering introduces an unsupervised learning step, insights on the meaning of these clusters is of interest. Since a valid number of clusters might exceed several hundreds, the visualization of each cluster in the original images does not provide a reasonable insight. Thus, a matching of each cluster to its preferred class might be more useful. To be able to distinguish between clusters equally occupied from all classes and clusters preferring a single or a few classes the entropy can be used:

$$H = -\sum_{c=1}^{C} p_c \cdot \log_2(p_c) \tag{5}$$

The threshold to find interesting clusters depends on the problem and the number of classes and must be tuned manually. If only clusters representing a single class are of interest, a simpler approach is the analysis of clusters where one class represents more than 50% of the data points.

3 Application

3.1 Corneal Confocal Microscopy

The practical evaluation of the presented methods is shown on images from the human cornea. The investigation of the cornea is a promising diagnostic method for peripheral neuropathy [15]. In order to acquire enough information of the innervation, multiple images must be recorded of the sub-basal nerve plexus (SNP), a thin (10 µm) layer with a high nerve density.

Since the manual recording is time consuming, Allgeier et al. proposed an automatic scanning process combined with a mosaicking of the resulting images [1]. Here, the focal plane follows a triangular wave to ensure the coverage of the SNP. In combination with an eye movement control, this leads to an automated recording process and the creation of increased field of view images usable for diagnostics. However, in the process also the surrounding tissues are covered, namely the epithelium and stroma. Since these tissues are not of interest and affect the mosaicking process, an automatic classification of the whole images is proposed to remove these images [3]. The combination of these approaches can also be used to improve the automated recording process itself. By using the classification or the segmentation for the control of the microscope, it might be possible to create an online control, where the focal plane follows the depth of the SNP [8]. This will not only increase the quality of the resulting images but also accelerate the recording process.

Confocal laser scanning microscopy of the cornea (CCM) allows non-invasive in vivo imaging of the nerve fibers constituting the corneal sub-basal nerve plexus (SNP). The thin SNP layer (~10 µm) is arranged parallel to the surface of the eye between the epithelium and the stroma tissue (Figure 4). Several studies found significant correlations between morphometric parameters of the SNP and the progression of diabetic peripheral neuropathy [15]. Due to local variance of the nerve fiber density, evaluation of a single image with a typical field of view of less than $0.2\,\mathrm{mm}^2$ is insufficient for reliable morphometric characterization.

To enlarge the field of view, Allgeier et al. proposed an automatic scanning process combined with a mosaicking of recorded CCM image sequences [1]. During the imaging process the focal plane oscillates around the SNP between the epithelium and the anterior stroma to ensure the capture of the thin SNP layer. In combination with an eye movement control, the

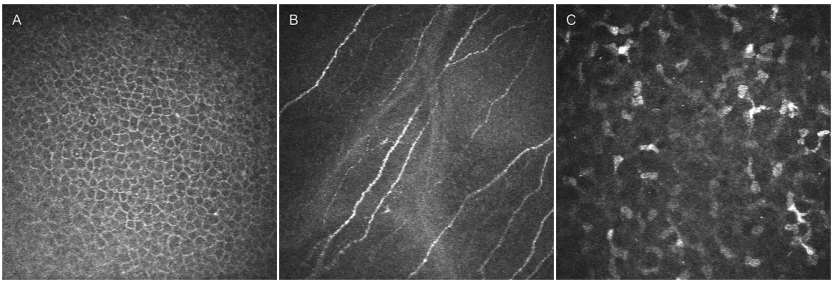


Figure 4: Example images of the tissues in the human cornea. (A) shows a typical epithelium tissue, (B) a typical SNP image with visible nerve fibers and (C) shows the stroma tissue.

recording process is fully automated and the increased field of view of the assembled mosaic image is suitable for a reliable morphometric SNP characterization. However, the SNP layer is superimposed by the images of the epithelium and the stromatissue in the mosaic image representation. Since these superimpositions affect the SNP representation, an automatic classification of the whole images is proposed to remove these images [3]. The combination of these approaches can also be used to improve the automated recording process itself. By using the tissue classification for the control of the CCM focus drive, it might be possible to create an online control, where the focal plane follows the depth of the SNP [8]. This will not only increase the quality of the resulting images but also accelerate the recording process.

3.2 Data

The image data used for the evaluation are *in vivo* CCM image sequences from eleven human volunteers. The recordings are made with the Heidelberg Retina Tomograph with the Rostock Corneal Module[1]. With a resolution of 384×384 pixels, the images cover an area of $0.16\,\mathrm{mm}^2$ of corneal tissue. Due to the anatomical structure of the cornea and the imaging technique, several challenges complicate the automatic analysis of these images. The intensity of the imaging laser must be low enough to avoid damages in the living tissue. Hence, the signal to noise ratio is suboptimal and the contrast is reduced. In combination with a decreasing intensity towards the edges of the images, automatic image processing is

[1]HRT II/RCM, Heidelberg Engineering, GmbH, Dossenheim, Germany

Table 1: Data distribution in S_{Train} and S_{Eval}. The images belong to eleven subjects and are labeled according to the tissue with the most visible area.

S_{Train}				S_{Eval}			
Sub.	Epithel	SNP	Stroma	Sub.	Epithel	SNP	Stroma
2	36	36	36	1	49	47	7
7	133	133	133	3	60	0	6
8	130	130	130	4	0	29	60
				5	0	47	70
				6	4	0	0
				9	23	43	0
				10	56	58	0
				11	23	23	58
Total	299	299	299	Total	215	247	201

challenging. Furthermore, irregularities and ridge-like deformations as well as the inclination of the imaging plane can result in the appearance of multiple tissue types in a single image (see Figure 7A).

For the evaluation, 1560 images were labeled image-wise as well as pixel-wise. The distribution of the images in terms of the training and evaluation set as well as the subjects are shown in Table 1. Given the presented images, a grid of $S = 37$ sampling points per dimension was used, resulting in a sampling point every ten pixels. The SURF descriptor was applied with a window size of $D = 25$ pixels per dimension as proposed in [3]. This led to a feature tensor $\mathbf{F}^{13\times13\times64}$. With a clustering of $K_c = 50$, these features were quantized with their cluster ID and the histogram over all clusters was created as the final image descriptor.

3.3 Evaluation

To provide a baseline for the classification results, the BoVW method was trained on S_{Train} and applied to S_{Eval} for the whole images. With the proposed parametrization, the accuracies shown in Table 2 were achieved. With an accuracy over 99% the method performed well, for the classification of entire images.

Table 2: Accuracy for the BoVW approach using whole images.

	Naïve Bayes	SVM
K-Means	$0,9185$	$0,9938$
EM	$0,9834$	$0,9984$

Table 3: Accuracy for the BoVW approach with a sliding window of $W = 11$ sampling points using $K_c = 50$ clusters. Trained and applied on image patches.

	Naïve Bayes	SVM
K-Means	$0,4771$	$0,8026$
EM	$0,5509$	$0,7928$

If the same parameters were applied for the batch learning with a window size of $W = 11$, the results shown in Table 3 were achieved. Even though the window was large, covering a quadratic patch with more than hundred pixels in each dimension, the accuracy of the example with full images could not be reached. However, the SVM clearly outperforms the Naïve Bayes classifier.

Due to the normalization of the histogram, the size of the window for training does not need to be the same as in the application. Hence, the training can be done over the full image and then applied with the sliding window approach.

As shown by the results in Table 4, the SVM outperformed the Naïve Bayes classifier and was able to achieve good results if the training took place on entire images rather than on patches. To further investigate the impact of the window size, Figure 5 shows the accuracy for different window sizes. To be able to investigate also smaller windows, the number of sampling points was doubled using every fifth pixel, resulting in $S = 75$ sampling points per dimension. The results of Table 3 are comparable to a window size of $W = 22$ in Figure 5.

Table 4: Accuracy for the BoVW approach trained on whole images and applied with the sliding window of 11 sampling points.

	Naïve Bayes	SVM
K-Means	$0,8592$	$0,9720$
EM	$0,8794$	$0,9240$

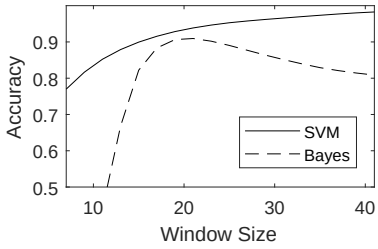


Figure 5: Accuracy of the sliding window approach for different window sizes.

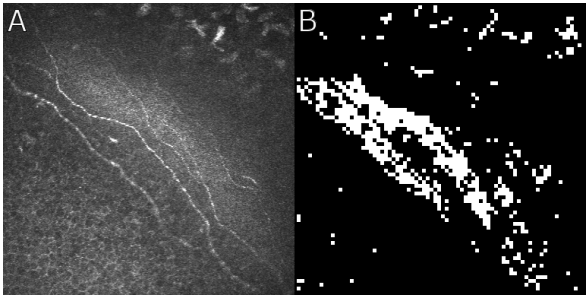


Figure 6: Image of the cornea showing all three tissue types (A) and sampling points with a low entropy ($H < 1$) for the class SNP are highlighted in (B).

3.4 Interpretation of Clusters

With the entropy evaluation of each cluster, the sampling points of the image can be investigated in order to find characteristic examples for each class. As shown in Figure 6, the method recognized nerve fibers in the image as a characteristic of SNP images. The same observation was made when investigating stroma cells. The most problematic tissue seemed to be the epithelium, as no cluster seemed to focus on the typical membrane structure visible in this class. The method seemed not to be able to distinguish effectively between those structures and the background, which resulted in clusters with background from all classes. If the subsequent classification method was trained, it created a bias towards the classification of the epithelium since the background of the other classes was also classified as epithelium.

This observation was also recognizable in the confusion matrix shown in Table 5. With a window size $W = 11$ the number of processed patches

Table 5: Confusion matrix for the sliding window of $W = 23$ sampling points using $K_c = 50$ cluster. If the rejected data points are not considered, the accuracy was 0.9422.

		Prediction			
		Epithelium	SNP	Stroma	Reject
Label	Epithelium	486275	6515	5	10688
	SNP	52448	514824	5934	54065
	Stroma	20209	4353	457381	16455

reached 1.6 million. The highest misclassification rate was observed for patches from SNP and stroma, which were classified as epithelium resulting in a positive prediction value PPV_{ep} (precision) for the epithelium of only 0.87:

$$\mathrm{PPV}_{ep} = \frac{\sum \text{True Positive}_{ep}}{\sum \text{True Positive}_{ep} + \sum \text{False Positive}_{ep}} \tag{6}$$

3.5 Probability Maps

Since the results of the SVM can be treated as a probability for each class, the sliding window approach can be used to create probability maps for each class in an image as shown in Figure 7. Since no treatment of the border was implemented, the results of pixels at the border could not be computed and were set to zero. Other than that, the bright pixels indicate a high probability for epithelium (B), SNP (C) and stroma (D) on the original image (A). These results cannot only be used to create a segmentation of the image but also provide additional insights to the classification, complementing the confusion matrix.

In the probability map for the epithelium (B), the area with actual epithelium had a high probability. However, also the area with the stroma showed an increased probability. The same was true for (D) where the area with stroma had a high probability but also the epithelium showed an increase in the probability. The area with SNP only showed a high probability in (C) and almost zero in the other maps. With the proposed classification method, the areas with a low probability for any class were rejected. The increase in information gained by this evaluation can potentially be used to improve the online control of the microscope as

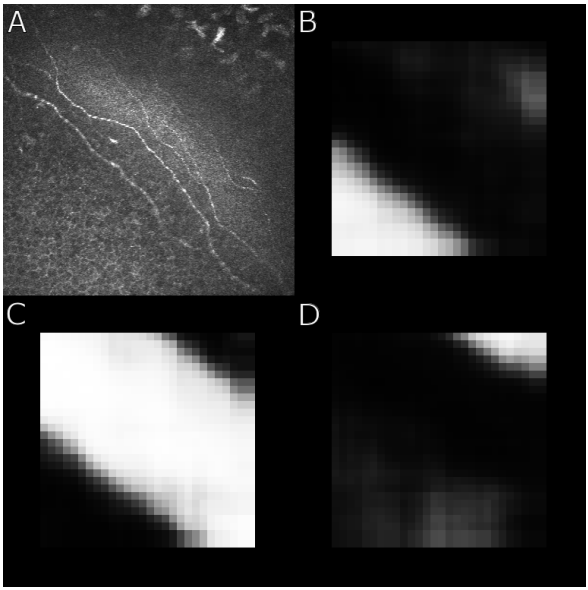


Figure 7: Image of the cornea showing all three tissue types (A). (B-D) are the probability maps for the classes epithelium (B), SNP (C) and stroma (D). The bright pixels indicate a high probability.

proposed in [8]. In addition, the segmentation and uncertainty can be used to realize precise depth adaptations.

4 Conclusion

The analysis of images with homogeneous regions and smooth transitions like medical images or the visual inspection of fabrics for quality control are difficult applications for machine learning methods. Not only needs the method to deal with a segmentation task, it needs also be able to deal with uncertainties and superpositions of textures. In this paper, we present augmentations of the BoVW pipeline with probabilistic SVMs. Trained in a one-against-all approach, the fuzzy result of the classification was able to model uncertainties and to reject images, which did not belong to any class. Furthermore, the application of the BoVW approach using a sliding window allows the classification of image patches, resulting in a segmentation of the image. This extends the global classification of

an image by a regional classification, able to characterize different tissue types.

Based on a use case with cornea tissues, we show that the method is a powerful tool to classify tissues in images. Given the fuzzy outputs of the multi-class classification, the transition of tissue types can be modeled and investigated. Furthermore, we showed that it is possible to train the method on whole images and apply it in a sliding window approach to small patches. This reduced the amount of required labeled data and even resulted in a higher accuracy for the shown use case.

The speed and the small amount of required training data makes the proposed method an alternative to deep learning based approaches, especially, for use cases with real-time constraints and homogeneous regions.

In addition, an entropy-based examination of the clusters was done, revealing typical structures for classes and indicating problems where no such structures are found. Since the classification method is also able to find typical combinations of clusters for certain structures, this might not be a sufficient analysis of the whole pipeline. However, the discriminative ability of the feature extractor can be investigated for every class.

Although the method has been applied to a specific problem, the evaluation on a benchmark data has yet to be done. However, since the method follows a specific use case, large benchmarks of computer vision challenges, e.g. for object detection are not suitable. We plan to integrate the proposed method with our image processing tool XPIWIT and will extend its functionality by machine learning methods [2].

Acknowledgment

The presented work was supported by the Helmholtz Association and the DFG (German Research Foundation, grant number MI 1315/5-1).

References

[1] Allgeier, S.; Maier, S.; Mikut, R.; Peschel, S.; Reichert, K.-M.; Stachs, O.; Köhler, B.: Mosaicking the Subbasal Nerve Plexus by Guided Eye Movements. *Investigative Ophthalmology & Visual Science* 55(9) (2014), p. 6082–6089.

[2] Bartschat, A.; Hübner, E.; Reischl, M.; Mikut, R.; Stegmaier, J.: XPIWIT - An XML Pipeline Wrapper for the Insight Toolkit. *Bioinformatics* 32(2) (2016), p. 315–317.

[3] Bartschat, A.; Toso, L.; Stegmaier, J.; Kuijper, A.; Mikut, R.; Köhler, B.; Allgeier, S.: Automatic Corneal Tissue Classification using Bag-of-visual-word Approaches. In: *Proc., Forum Bildverarbeitung, Karlsruhe*, p. 245–256, KIT Scientific Publishing, 2016.

[4] Bay, H.; Tuytelaars, T.; Van Gool, L.: SURF: Speeded Up Robust Features. In: *9th European Conference on Computer Vision (ECCV)* (Leonardis, A.; Bischof, H.; Pinz, A., Ed.), p. 404–417, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[5] Bishop, C. M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer, 2006.

[6] He, K.; Zhang, X.; Ren, S.; Sun, J.: Deep Residual Learning for Image Recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 770–778, 2016.

[7] Isensee, F.; Jaeger, P.; Full, P. M.; Wolf, I.; Engelhardt, S.; Maier-Hein, K. H.: Automatic Cardiac Disease Assessment on cine-MRI via Time-Series Segmentation and Domain Specific Features. *arXiv preprint arXiv:1707.00587* (2017).

[8] Köhler, B.; Allgeier, S.; Bartschat, A.; Guthoff, R. F.; Bohn, S.; Reichert, K.-M.; Stachs, O.; Winter, K.; Mikut, R.: In-vivo-Bildgebung des kornealen Nervenplexus. *Der Ophthalmologe* 114(7) (2017), p. 601–607.

[9] Oeljeklaus, M.; Hoffmann, F.; Bertram, T.: Convolutional Neural Networks für die semantische Segmentierung von Szenen. In: *Proc., 25. Workshop Computational Intelligence, Dortmund*, p. 241–254, KIT Scientific Publishing, 2015.

[10] Okafor, E.; Pawara, P.; Karaaba, F.; Surinta, O.; Codreanu, V.; Schomaker, L.; Wiering, M.: Comparative Study Between Deep Learning and Bag of Visual Words for Wild-Animal Recognition. In: *IEEE Symposium Series on Computational Intelligence (SSCI)*, p. 1–8, 2016.

[11] Platt, J.: Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Advances in Large Margin Classifiers* 10(3) (1999), p. 61–74.

[12] Richard, A.; Gall, J.: A Bag-of-Words Equivalent Recurrent Neural Network for Action Recognition. *Computer Vision and Image Understanding* 156 (2017), p. 79–91.

[13] Sivic, J.; Zisserman, A.: Video Google: a Text Retrieval Approach to Object Matching in Videos. In: *Proc. 9th IEEE International Conference on Computer Vision (ICCV)*, 2, p. 1470–1477, 2003.

[14] Stegmaier, J.; Otte, J. C.; Kobitski, A.; Bartschat, A.; Garcia, A.; Nienhaus, G. U.; Strähle, U.; Mikut, R.: Fast Segmentation of Stained Nuclei in Terabyte-Scale, Time Resolved 3D Microscopy Image Stacks. *PLoS ONE* 9(2) (2014), p. e90036.

[15] Tavakoli, M.; Petropoulos, I.; Malik, R. A.: Corneal Confocal Microscopy to Assess Diabetic Neuropathy: An Eye on the Foot. *Journal of Diabetes Science and Technology* 7(5) (2013), p. 1179–1189.

Comparing Kriging and Radial Basis Function Surrogates

Samineh Bagheri[1], Wolfgang Konen[1], Thomas Bäck[2]

[1]Department of Computer Science
TH Köln (University of Applied Sciences)
Steinmüllerallee 1
51643 Gummersbach
E-Mail: {wolfgang.konen, samineh.bagheri}@th-koeln.de,

[2]Department of Computer Science
Leiden University, LIACS,
2333 CA Leiden, The Netherlands
Email: T.H.W.Baeck@liacs.leidenuniv.nl

Abstract

Radial basis function interpolation (RBF) and Kriging based on the Gaussian processes models (GPs) are great tools for multidimensional surrogate modeling. They can often deliver accurate models for complicated nonlinear functions. Although RBFs and GPs have very different origins, they share many fundamentals in practice. Gaussian processes not only provide a model but also an error estimate associated with the model which indicates the uncertainty of the model at different points. The uncertainty property determined by GPs is dependent on the distribution of the sample points in the input space and as expected the model error estimate is low where the distribution is dense and large where the distribution is sparse. Therefore, the uncertainty property can be determined regardless of the modeling technique.

In this work, we compare RBFs and GPs from the theoretical point of view. We show how to calculate the model uncertainty for any arbitrary kernel, e. g. cubic RBF, augmented cubic RBF and other types. Furthermore, we replace the Kriging model from the DiceKriging R package used in the SOCU framework [1] with a vectorized RBF model and report some preliminary results. We show that the new implementation of SOCU with RBF is faster than the older one. It delivers in many cases equal or even better optimization accuracy.

1 Introduction

Real-world optimization problems are often black-box and function evaluation can be very expensive.In order to handle such problems, different surrogate-assisted constrained optimization algorithms have been developed [1, 10, 8]. A common approach in surrogate-assisted optimization is the so-called *Expected Improvement* (EI) method based on Kriging models. EI was originally developed for unconstrained optimization, but approaches for constrained optimization have been proposed as well [1, 10], one of them being the SOCU algorithm [1].

Kriging has the big advantage of providing uncertainty information in surrogates, which is a necessary prerequisite for EI. But Kriging – at least in most currently available implementations – has also some disadvantages: We experienced in SOCU often crashes of the Kriging package if we did not introduce some form of regularization by setting a nonzero value to the noise variance parameter. This, however, leads in turn to less accurate models. Secondly, Kriging model calculations are often time-consuming, if either the dimensionality, the number of design points or the number of constraints becomes higher.

RBF surrogate models, which we used in other optimizers [4, 8], can be computed fast, in vectorized form, and robustly. They lack however the uncertainty information. The motivation for this paper is driven by the following research question: *Can we advise - by exploiting the analogies between RBF and GP - some form of RBF modeling which provides an uncertainty measure?* And if so, can we apply it to an EI-based optimization scheme like SOCU and in such arrive at a method having one or several of these desirable properties:

- Avoiding crashes without regularization
- Providing more accurate models
- Better computation time (e. g. through vectorization)
- More variety in radial basis kernels (parameter-free, augmented, ...)

The rest of this paper is organized as follows: In Section. 2, we describe Gaussian processes, radial basis function interpolation and their connection with each other. A brief description of the SOCU algorithm also appears in the same section. Section 3 describes the experimental setup and the test functions. We report our results and compare different versions of SOCU in Section 4. Section 5 concludes the paper.

Table 1: Commonly used kernel functions for GP and radial basis functions for RBF interpolation. $r = ||x_i - x_j||$

Name	GP	RBF
cubic	–	$\varphi(r) = r^3$
Gaussian	$\sigma e^{-\frac{r^2}{2\alpha^2}}$	$\varphi(r, \alpha) = e^{-\frac{r^2}{2\alpha^2}}$
multiquadric	–	$\varphi(r, \alpha) = \sqrt{1 - (\frac{r}{\alpha})^2}$
matern(3-2)	$\sigma(1 + \frac{\sqrt{3}r}{\alpha}) \exp(-\frac{\sqrt{3}r}{\alpha})$	–

2 Methods

2.1 Gaussian Processes

Gaussian processes (**GP**) – also known as Kriging – is a probabilistic modeling technique which applies Bayesian inferences over functions. Let us assume that an unknown function f is evaluated on a finite set of n arbitrary points $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ and $f_i = f(\mathbf{x}_i) = y_i$. The Gaussian processes method assumes that $p(f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots, f(\mathbf{x}_n))$ belongs to a multivariate (jointly) Gaussian with a mean μ and covariance matrix $\mathbf{\Sigma}$:

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \sim N \left(\begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1n} \\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{n1} & \Sigma_{n2} & \cdots & \Sigma_{nn} \end{bmatrix} \right) \sim N(\mu, \mathbf{\Sigma}) \qquad (1)$$

where $\Sigma_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. The covariance matrix contains the dependencies and similarities of random variables, in this case the $f(x_i)$. Suppose the unknown function f is smooth, then it is very likely that two points which are located very close to each other in the input space have very similar values in the output space too. This explains why the κ is also known as the similarity function and is often symmetric and a function of distances. Table 1 shows several commonly used kernel functions.

Suppose that we want to predict f_* the value of function f at a new point $\mathbf{x}_*$. The joint Gaussian distribution including the new point is

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \\ f_* \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \\ \mu_* \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1n} & \Sigma_{1*} \\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2n} & \Sigma_{1*} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Sigma_{n1} & \Sigma_{n2} & \cdots & \Sigma_{nn} & \Sigma_{n*} \\ \Sigma_{*1} & \Sigma_{*2} & \cdots & \Sigma_{*n} & \Sigma_{**} \end{bmatrix} \right) \tag{2}$$

which can be summarized as follows

$$\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} \sim N\left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} \mathbf{K} & K_* \\ K_*^T & K_{**} \end{bmatrix} \right), \tag{3}$$

where $\mathbf{K} = \Sigma$ is the $n \times n$ matrix of Eq. (1), $K_* = \kappa(\mathbf{X}, x_*)$ is an $n \times 1$ vector and $K_{**} = \kappa(\mathbf{x}_*, \mathbf{x}_*)$ is a scalar and $\mathbf{f} = \{f_1, f_2, \cdots, f_n\}$ is an $n \times 1$ vector. $\mathbf{X} = (\mathbf{x}_1 \ldots \mathbf{x}_n)$ is the matrix of the data points. We look for the probability of f_* when the data $\mathbf{X}$, their corresponding values $\mathbf{f}$ and the new point $\mathbf{x}_*$ are given. Based on the conditional probability theorem [6] we can determine a distribution at every new point as follows:

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{f}) = N(K_*^T \mathbf{K}^{-1} \mathbf{f}, K_{**} - K_*^T \mathbf{K}^{-1} K_*) = N(\mu_*, \Sigma_*), \tag{4}$$

where μ_* and Σ_* can be interpreted as the mean and the uncertainty of the Gaussian processes model, respectively. Fig. 1 shows an example of Gaussian process with a Gauss kernel function. Fig. 1-left illustrates several samples from the prior $p(\mathbf{f}|\mathbf{X})$ and Fig. 1-right shows samples from the posterior function $p(f_*|\mathbf{X}_*, \mathbf{X}, \mathbf{f})$. The dark black curve is the mean μ_* and the shaded area is showing the 90% confidence interval.

We can rewrite the mean as follows:

$$\begin{aligned} f_* &= K_*^T \mathbf{K}^{-1} \mathbf{f} \\ f_* &= K_*^T \underline{\theta} \\ f_* &= \sum_{i=1}^{n} \theta_i \kappa(\mathbf{x}_i, \mathbf{x}_*), \end{aligned} \tag{5}$$

Eq. (5) shows that the mean of the GP model can be determined as a linear summation of weighted kernel functions.

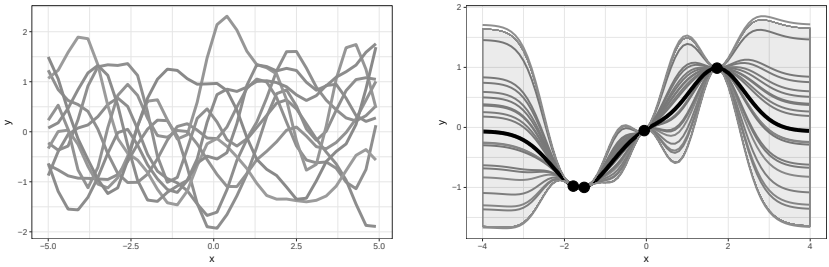


Figure 1: Left: prior. Right: posterior.

The uncertainty term in Eq. (4) can be rewritten as:

$$\Sigma_* \quad = \quad -K_*^T K^{-1} K_* + K_{**} \tag{6}$$

It is important to mention that the uncertainty of the model estimated by Eq. (6) is only a function of the distribution of points in the input space. Fig. 1 illustrates that the model uncertainty goes to zero at the given points and it becomes larger as the distance from the evaluated points increases.

2.2 Radial Basis Function Interpolation

Radial basis function (RBF) interpolation developed by Hardy in 1971 for cartography purposes was meant to be a suitable interpolation technique to model hills and valleys with a reasonably high local and global accuracy. RBF interpolation approximates a function by fitting a linear weighted combination of radial basis functions. A radial basis function is by definition any function $\varphi(||\cdot||)$ which is dependent on the distance of the points $\mathbf{x}$ from some fixed centers $\mathbf{c}$. It is common that all the evaluated points $\mathbf{x}_i$ are considered as centers.

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \theta_i \phi(||\mathbf{x} - \mathbf{x}_i||) \tag{7}$$

In order to compute the weights θ_i we need to address the following linear system:

$$[\mathbf{\Phi}]\,[\theta] = [\mathbf{f}], \tag{8}$$

where $\mathbf{\Phi} \in \mathbb{R}^{n \times n}$: $\Phi_{ij} = \varphi(||\mathbf{x}_i - \mathbf{x}_j||), i, j = 1, \ldots, n$ and $\mathbf{f} = f_1, f_2, \cdots, f_n$. Therefore, the weights will be determined as following:

$$\theta = \mathbf{\Phi}^{-1}\mathbf{f} \tag{9}$$

Now that we have the weights θ, we can compute f_* at any point $\mathbf{x}_*$:

$$f_* = \sum_{i=1}^{n} \theta_i \phi(||\mathbf{x}_* - \mathbf{x}_i||) = \Phi_*^T \theta = \Phi_*^T \mathbf{\Phi}^{-1}\mathbf{f} \tag{10}$$

where $\Phi_*^T = [\varphi(||\mathbf{x}_* - \mathbf{x}_1||), \varphi(||\mathbf{x}_* - \mathbf{x}_2||), \cdots, \varphi(||\mathbf{x}_* - \mathbf{x}_n||)]$.

2.3 Augmented RBF

It is proven that $\mathbf{\Phi}$ in Eq. 8 is not guaranteed to be positive definite for all radial basis functions [5]. In order to assure that the Eq. 8 has a unique solution with all radial basis functions, Micchelli introduced augmented RBFs [5]. Augmented RBFs are actually RBF functions with a polynomial tail.

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \theta_i \varphi(||\underline{x} - \underline{u}^{(i)}||) + p(\underline{x}), \quad \underline{x} \in \mathbb{R}^d, \tag{11}$$

where $p(\underline{x}) = \mu_0 + \underline{\mu_1}\,\underline{x} + \underline{\mu_2}\,\underline{x}^2 \cdots + \underline{\mu_k}\,\underline{x}^k$ is a k-th order polynomial in d variables with $kd + 1$ coefficients.

The augmented RBF model requires the solution of the following linear system of equations:

$$\begin{bmatrix} \mathbf{\Phi} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0}_{(kd+1)\times(kd+1)} \end{bmatrix} \begin{bmatrix} \underline{\theta} \\ \underline{\mu}' \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0_{(kd+1)} \end{bmatrix} \tag{12}$$

Here, $\mathbf{P} \in \mathbb{R}^{n \times (kd+1)}$ is a matrix with $(1, \underline{x}_{(i)}, \underline{x}_{(i)}^2)$ in its ith row, $\mathbf{0}_{(kd+1)\times(kd+1)} \in \mathbb{R}^{(kd+1)\times(kd+1)}$ is a zero matrix, $0_{(kd+1)}$ is a vector of zeros. In this work we use the augmented cubic radial basis function with a second order polynomial tail.

2.4 GP vs. RBF

Although GP and RBF interpolation have two very different origins, the comparison of Eq. (10) and Eq. (5) shows that the mean of GP is identical to the RBF result, if the kernel function κ is identified with the basis function φ. On the other hand, GPs provide beside the prediction of the mean also a prediction of the model uncertainty Σ_* (Eq. (6)). Although radial basis functions by definition do not have any sort of uncertainty measure, we can determine the model uncertainty for any radial basis function in a similar way as GP does.

$$\Sigma_{rbf} = \varphi(||\mathbf{x}_* - \mathbf{x}_*||) - \Phi_*^T \boldsymbol{\Phi}^{-1} \Phi_*, \tag{13}$$

where $\varphi(||\mathbf{x}_* - \mathbf{x}_*||) = \varphi(0)$ is a scalar value.

Fig. 2 illustrates that RBF and Kriging with the same kernel type and parameters give almost the same results. The minimal differences in the first three columns are due to different matrix inversion techniques which the two implementations use. As it is shown in Fig. 2 the choice of kernel parameter has a large impact on the quality of the models. In this example, small values of α resulted in a very non-informative spiky model. The correct choice of parameter(s) is often a problem dependent challenging task. Kriging [9] tunes the kernel parameter(s) based on the maximum likelihood estimation (MLE) approach which its computational complexity is approximately $\mathcal{O}(\frac{1}{3}n^3 + \frac{1}{2}dn^2)$.

The kernel parameters for the RBF interpolation are often set manually. A recent work [2] suggests an online selection algorithm for choosing the best kernel type and parameters during an optimization process. In this work we use a parameter free radial basis function.

The plots in the last column of Fig. 2 are generated by the default configuration of RBF and Kriging. RBF's default configuration uses an augmented cubic kernel function with no need for parameter tuning. Kriging uses a Gauss kernel and the kernel parameters are assigned by MLE. We can see that the augmented cubic RBF model produces smaller errors than the default Kriging model (with MLE). Furthermore, we can observe that the Kriging model with MLE is not as good as Kriging with fixed parameters $\sigma = 1, \alpha = 10$ for the example shown in Fig. 2. Rasmussen and Williams [7] show that MLE cannot guarantee optimal estimation of kernel parameters, since the likelihood function can have multiple local optima. MLE can suffer from getting stuck in such a local optimum.

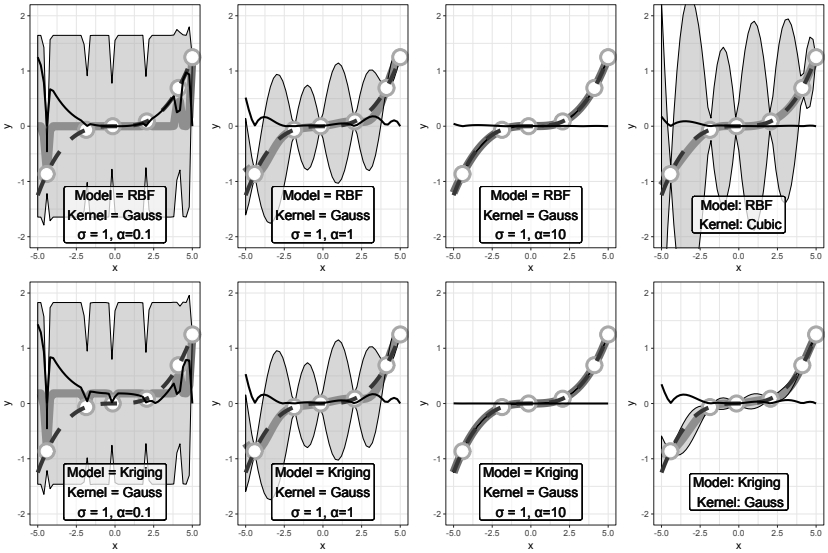


Figure 2: Comparing RBF and GP from the DiceKriging package in R. The examples in the first row are all generated by RBF and in the second row with GP. The plots in the last column are generated by the default configuration of RBF and GP. The dashed curve $f = x^3$ is the target curve to be modeled. The circles are the evaluated points. The solid thick curve is the delivered model. The solid thin curve is the model's absolute error. The gray areas indicate the 90% model uncertainty. For the same kernel function and same parameters, RBF and GP produce almost the same results. The minimal differences are due to different matrix inversion techniques used by the two implementations.

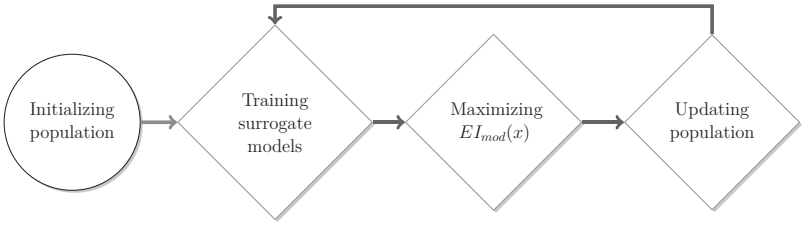


Figure 3: Main steps of the SOCU algorithm.

2.5 SOCU: Surrogate-Assisted Optimization encompassing Constraints and Uncertainties

The SOCU algorithm developed in [1] is a surrogate assisted constrained optimization method which makes use of Gaussian processes to model objective function and feasibility probability.

SOCU is a sequential EGO-based [3] optimization algorithm with four main steps as illustrated in Fig. 3. After generating an initial population of solutions, the objective and constraint function(s) are modeled. The next step is to generate a new infill point which maximizes the modified expected improvement:

$$EI_{mod}(x) = EI(x) \cdot F(x) = EI(x) \cdot \prod_{i=1}^{m} \min\Big(2P(g_i(x) < 0), 1\Big), \qquad (14)$$

where $EI(x)$ is the expected improvement of the objective function and $P(g_i(x) < 0)$ is the probability of the i-th constraint not being violated. The single new infill point is evaluated on the real functions and added to the population. The last three steps will be repeated as long as the budget is exhausted. More details about the computation of $EI_{mod}(x)$ can be found in [1].

3 Experimental Setup

In this work we compare two versions of the SOCU optimization framework, namely SOCU-Kriging and SOCU-RBF. SOCU-Kriging uses the Kriging model from R packages DiceKriging and DiceOptim, while SOCU-RBF uses our own implementation of RBF interpolation. The first

Table 2: Characteristics of the G-functions: d: dimension, ρ^*: feasibility rate (%), LI/NI: number of linear / nonlinear inequalities, a: number of constraints active at the optimum. We selected here only those G-functions without equality constraints.

Fct.	d	ρ^*	LI / NI	a
G01	13	0.0003%	9 / 0	6
G04	5	26.9217%	0 / 6	2
G06	2	0.0072%	0 / 2	2
G07	10	0.0000%	3 / 5	6
G08	2	0.8751%	0 / 2	0
G09	7	0.5207%	0 / 4	2
G10	8	0.0008%	3 / 3	6
G12	3	0.04819%	0 / 1	0
G24	2	0.44250%	0 / 2	2

major difference between the two versions of SOCU is the choice of kernel functions. SOCU-Kriging uses matern3-2 which is a relatively stable kernel function according to our initial experiments. SOCU-RBF makes use of cubic basis function which is a parameter-free kernel function.

DiceKriging uses a maximum likelihood estimation (MLE) algorithm to tune the two parameters of the matern3-2 kernel. The second difference is the numerical technique used in both versions for the required matrix inversion. The DiceKriging package [9] uses Cholesky decomposition but we found singular value decomposition used for RBF to be a more stable approach. In our experiments described in [1] we experienced frequent crashes of Kriging models. It was possible to cure this problem by using a non-zero regularization factor that can be assigned as the noise variance parameter. In this work we represent SOCU-Kriging results with two regularization factors of $\delta = \{10^{-3}, 10^{-4}\}$. The third major difference is an implementation detail which is the underlying reason for SOCU-RBF being much more time-efficient than SOCU-Kriging. The DiceKriging package does not support modeling several functions simultaneously which means that for a problem with m constraints, we have to run through a loop $m + 1$ times in each iteration, while SOCU-Kriging uses vectorization and performs training and prediction of all models within one pass. The main differences between SOCU-RBF and SOCU-Kriging are summarized in Table. 3.

Table 3: Differences between SOCU-Kriging and SOCU-RBF

	SOCU-Kriging	SOCU-RBF
kernel	matern3-2	cubic
parameter assignment	MLE	parameter free
matrix inversion	cholesky decomposition	svd
noise variance	0.001	0.0
vectorization	no	yes

We apply SOCU-Kriging and SOCU-RBF to the subset of G-problems having only inequality constraints (see Table 2). G02 is a scalable problem in its dimension d. We use here $d = 2$. For each algorithm we run 10 independent trials with different $n = 3d$ initial points. In order to optimize EI_{mod} we use Generalized Simulated Annealing (R package GenSA).

4 Results and Discussion

4.1 Performance on G-problems

Fig. 4 shows the optimization results over iterations for SOCU-Kriging and SOCU-RBF. For most of the problems, SOCU-RBF performs better than or comparable to SOCU-Kriging except for G12 and G24. G12 and G24 are the only problems where SOCU-RBF has a larger median error. However, several optimization runs for G12 conducted by SOCU-RBF perform better than the best runs of SOCU-Kriging.

4.2 Computational Time

Fig. 5 clearly shows that SOCU-Kriging is computationally more expensive than SOCU-RBF for all G-problems. SOCU-Kriging's computational time varies strongly in a range of (0.5-2.5) minutes per iteration for different G-problems, while SOCU-RBF's computational time per iteration is under 0.75 minutes regardless of the problem. The difference between computational time of SOCU-Kriging and SOCU-RBF is dependent on

the number of constraint functions. For example, the largest gap between SOCU-Kriging and SOCU-RBF appears to be for G01 and G07 which have 9 and 8 constraint functions, respectively. We can observe that solving G12 with one constraint has almost the same computational cost for SOCU-Kriging and for SOCU-RBF.

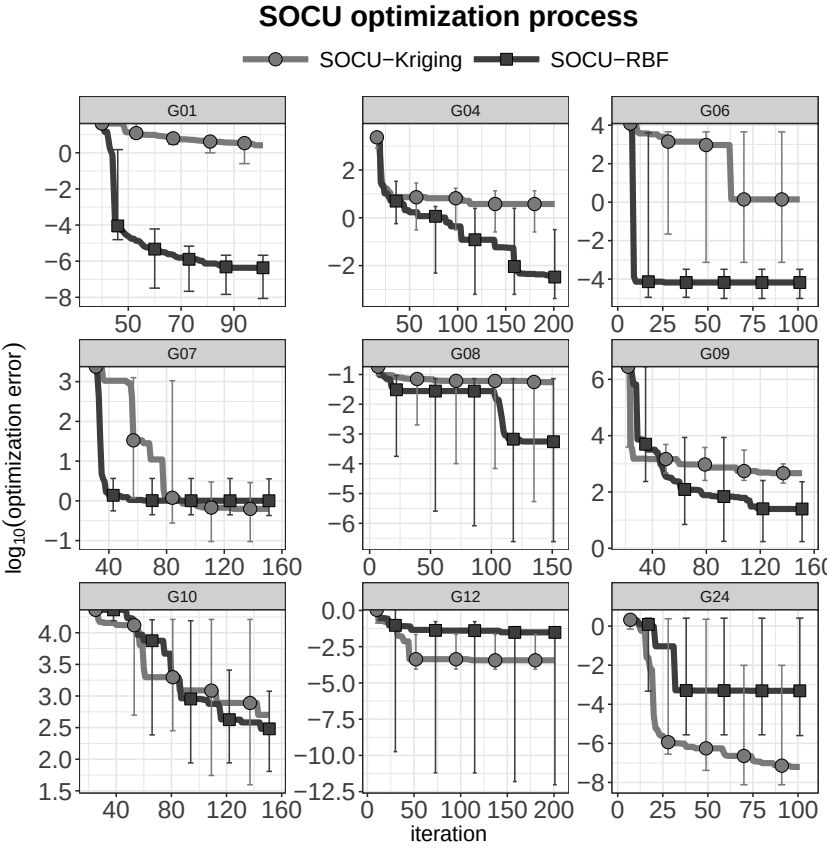


Figure 4: Comparing optimization performance of SOCU-Kriging and SOCU-RBF on G-problems. The curves are showing the median error out of 10 trials. The error bars indicate the best and the worst results out of 10 trials.

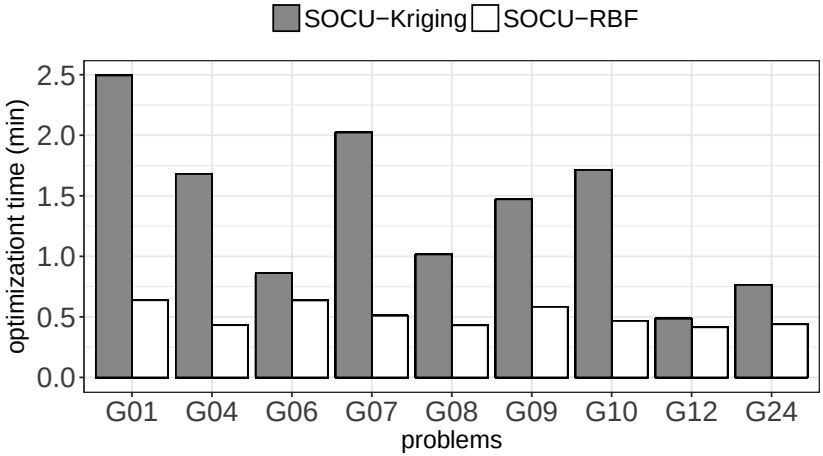


Figure 5: Average computational time required by SOCU-Kriging and SOCU-RBF to run one iteration of each G-problem.

4.3 Noise Variance

We have already shown that SOCU-RBF outperforms SOCU-Kriging in Fig. 4. The SOCU-Kriging algorithm used for generating Fig. 4 has a non-zero noise variance $\delta = 10^{-3}$. One possible reason behind the weaker performance of SOCU-Kriging in comparison to SOCU-RBF can be that SOCU-Kriging generates less accurate models due to the non-zero noise variance value. In order to investigate the impact of noise variance we applied the SOCU-Kriging framework to all G-problems in Table. 2 with two different noise variance values $\delta_1 = 10^{-3}$ and $\delta_2 = 10^{-4}$. SOCU-Kriging with the smaller noise variance $\delta_2 = 10^{-4}$ crashed on the G06 problem. Fig. 6 compares the SOCU-Kriging optimization results with two different noise variance values for all problems excluding G06.

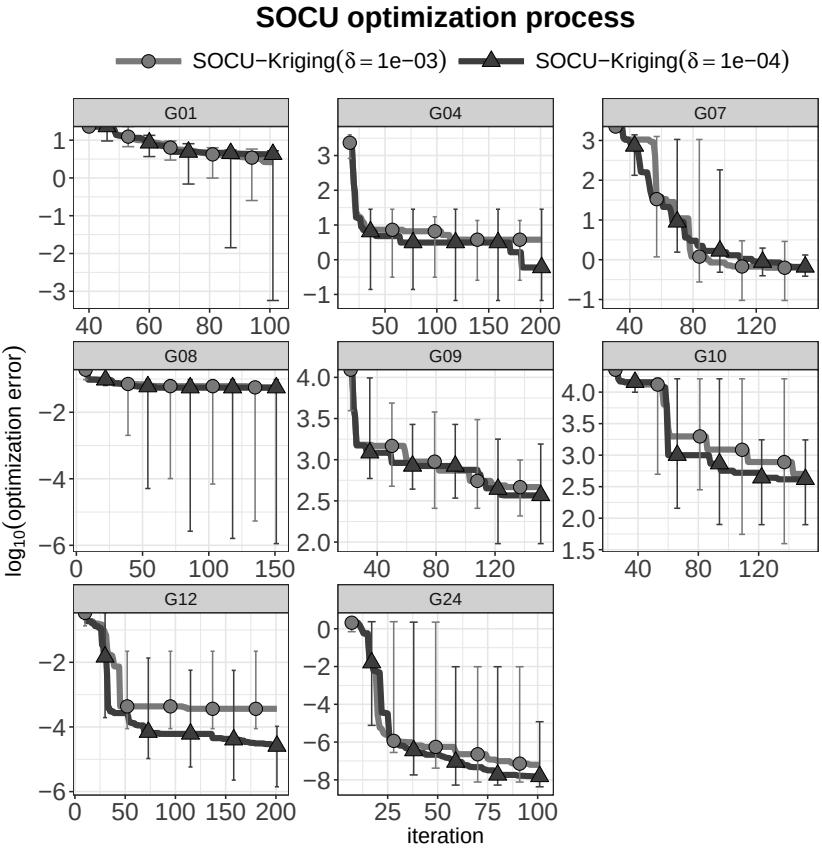


Figure 6: Comparing results of SOCU-Kriging with two different noise variance values δ. The curves are showing the median optimization error of the 10 independent trials for each algorithm. The error bars are indicating the best and worst case results.

Fig. 6 indicates that a smaller noise variance value can lead to a slightly better optimization results. For all the problems illustrated in Fig. 6 except G07, SOCU-Kriging $\delta = 10^{-4}$) has a smaller median or min. error. It is not possible to set the noise variance to zero because this would produce frequent crashes for SOCU-Kriging.

4.4 Model Accuracy

SOCU-Kriging and SOCU-RBF are only distinct in the modeling approach. Therefore, we hypothesize that the different optimization results observed in Fig. 4 are due to the different model quality. The performance of SOCU-Kriging and SOCU-RBF is significantly different especially on the G01 problem. In order to validate our hypothesis, we show the approximation error determined during the optimization process with various SOCU configurations in Fig. 7. As it is shown in Fig. 7, the approximation error of SOCU-RBF is significantly smaller than both SOCU-Kriging versions for objective and constraint functions. This shows that Kriging with matern3-2 kernel and optimized parameters through MLE cannot compete with our implementation of RBF with the parameter free augmented cubic kernel for G01. A large part of SOCU-RBF's better performance can probably be attributed to the augmented part. This advantage may depend on the type of the function to be modeled.

Comparing different versions of SOCU-Kriging in Fig. 7, we can also observe that SOCU-Kriging with a smaller noise variance $\delta = 10^{-4}$ has slightly smaller approximation error in the last iterations.

5 Conclusion

We explored in this work the similarities and differences between Kriging- and RBF-based surrogate models. As a new point from this comparison we could implement an uncertainty measure for RBFs which is needed for EI-optimization. RBFs allow a greater variety of kernel functions, notably in the form of augmented RBF variants introduced in Sec. 2.3. This helps to avoid crashes in the model-building process, which are otherwise encountered from time to time in Kriging modeling. RBF models have shown to provide a higher modeling accuracy and higher robustness (they do not produce crashes in all our experiments).

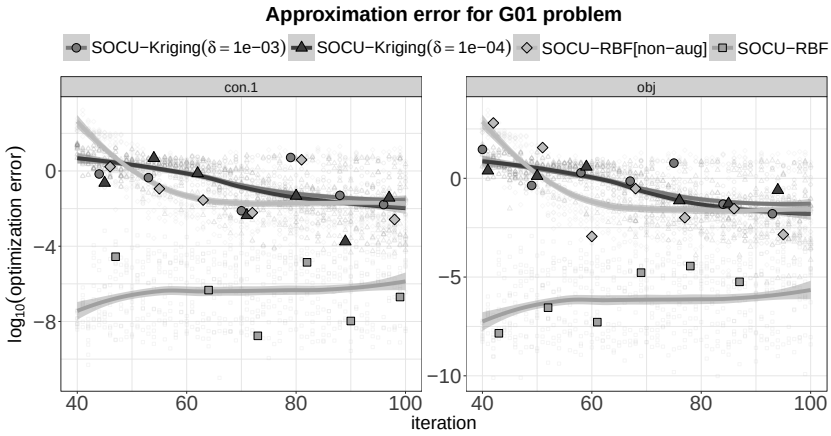


Figure 7: Approximation error for the objective and constraint functions of the G01 problem. G01 has 9 constraints but because of lack of space we just show the approximation error of the objective (a quadratic function) and one of its constraint functions (a linear function). Since all 9 constraints are of the same type, their approximation error curves looks similar. The approximation error is the error in predicting at the new infill point before this point is added to the population.

The new RBF surrogate models including uncertainties were tested on certain optimization benchmarks (a subset of the G-problems). The overall results were better, both in terms of solution quality and computational time. Probably a large part of the quality improvement may be attributed to the ability of augmented RBF models to include a polynomial tail. This may be a large or small advantage, depending on the type of functions to be modeled.

References

[1] Bagheri, S., Konen, W., Allmendinger, R., Branke, J., Deb, K., Fieldsend, J., Quagliarella, D., Sindhya, K.: Constraint handling in efficient global optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 673–680. GECCO '17, ACM, New York, NY, USA (2017)

[2] Bagheri, S., Konen, W., Bäck, T.: Online selection of surrogate models for constrained black-box optimization. In: 2016 IEEE Sym-

posium Series on Computational Intelligence (SSCI). pp. 1–8 (Dec 2016)

[3] Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization 13(4), 455–492 (Dec 1998)

[4] Koch, P., Bagheri, S., Foussette, C., Krause, P., Bäck, T., Konen, W.: Constrained optimization with a limited number of function evaluations. In: Hoffmann, F., Hüllermeier, E. (eds.) Proceedings of the 24th Workshop on Computational Intelligence. pp. 119–134. Universitätsverlag Karlsruhe (2014), young Author Award GMA-CI

[5] Micchelli, C.A.: Interpolation of scattered data: Distance matrices and conditionally positive definite functions. Constructive Approximation 2(1), 11–22 (Dec 1986)

[6] Murphy, K.P.: Machine Learning: A Probabilistic Perspective. The MIT Press (2012), pp. 118-121

[7] Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. the MIT Press (2006), pp. 114-117

[8] Regis, R.G.: Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. Engineering Optimization 46(2), 218–243 (2013)

[9] Roustant, O., Ginsbourger, D., Deville, Y.: DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by Kriging-based metamodeling and optimization. Journal of Statistical Software 21, 1–55 (2012)

[10] Schonlau, M., Welch, W.J., Jones, D.R.: Global versus local search in constrained optimization of computer models. In: Flournoy, N., et al. (eds.) New developments and applications in experimental design, Lecture Notes–Monograph Series, vol. 34, pp. 11–25. Institute of Mathematical Statistics, Hayward, CA (1998)

Bias-Correction of Satellite Rainfall Estimates Through the Use of Metamodels using Gaussian Process and Bayesian Regression. A Case Study for the Imperial Basin (Chile)

Margarita Alejandra Rebolledo Coy[1], Oscar Manuel Baez Villanueva[2], Thomas Bartz-Beielstein[1], Lars Ribbe[2]

[1]SPOTseven Lab, TH Köln
Steinmüllerallee 1, 51643, Gummersbach
E-Mail: margarita.rebolledo@th-koeln.de,
thomas.bartz-beielstein@th-koeln.de

[2] Institute for Technology and resources management in the Tropics and Subtropics (ITT), TH Köln
Betzdorfer Straße 2, 50679 Kön
E-Mail: oscarmbaez89@hotmail.com, lars.ribbe@th-koeln.de

1 Problem Description

Precipitation is a key parameter in the water cycle and plays an important role in both weather and climate. Nevertheless, an accurate assessment of the Spatio-temporal variability of rainfall is severely limited [1]. The traditional approach to determine the spatial distribution of rainfall is to use a dense network of rain gauge stations over the study area [2, 3]. However, in developing countries, hard to access areas and even over the ocean this rain gauge network can be sparse or non-existent. The use of rainfall estimation algorithms on satellite imagery to generate Satellite Rainfall Estimates (SRE) has helped to partially solve this problem, as satellites provide imagery at high temporal and spatial resolution. The SREs are algorithm-based precipitation datasets. These algorithms use information collected by infrared (IR) sensors aboard geostationary (GEO) satellites and passive or active microwave sensors aboard low-earth orbit (LEO) satellites. It is worth to mention that the SREs are estimates of the rainfall constructed with indirect measurements. Therefore, SREs may be used incorrectly if a validation process is not carried out [4].

Several SRE have been recently developed such as the Multi-source Weighted Ensemble Precipitation (MSWEP) [5], the Climate Prediction Center Morphing technique product (CMORPH) [6], the Precipitation Estimation from Remotely Sensed Information using Artificial Neural Networks-Climate Data Record (PERSIANN-CDR) [8], the Tropical Rainfall Measuring Mission (TRMM) Multi-satellite Precipitation Analysis [9], and the Climate Hazards group Infrared Precipitation with Stations dataset (CHIRPS) [10] among others.

Zambrano-Bigiarini et al. [1], evaluated the temporal and spatial performance of the above mentioned SREs over Chile in a monthly scale. The authors concluded that despite continuous improvement on most SRE products, different types of discrepancies between SREs and ground observations might be reduced by using local observations to calibrate the satellite estimates. Following these results we propose a bias correction for the SREs using regression methods to correct the satellite estimate error for the Imperial river basin in Chile. The study area has a rain gauge network of 13 station which record the real rainfall value. We model the SRE against the ground station measurements using Gaussian process and Bayesian regression methods. The regression will be carried out using yearly and seasonal time frames to find out which applies better for the error correction. The obtained models will be analysed to discover which method better fits the area of study and the error characteristics will be evaluate to find possible patterns. Additionally it is of our interest to investigate how the two different regression methods perform in this task.

1.1 Study Area

The Imperial river basin is a chilean catchment located in the ninth region named Araucanía, one of the fifteen regions of Chile. The Imperial basin has an area of 12.763 Km^2 with 540599 inhabitants and a river length of about 230 kilometres. Fig. 1 shows the location of the basin together with the location of the 13 rainfall gauge stations it contains.

According to the General Water Direction of Chile (DGA), this basin presents two climate types. A tempered warm rainy climate with a mediterranean influence in the center and low sectors. On the other hand, the highest areas present a tempered cold rainy climate with mediterranean influence. Both climates record a mean on annual precipitation of about 1245 mm and 1850 mm respectively.

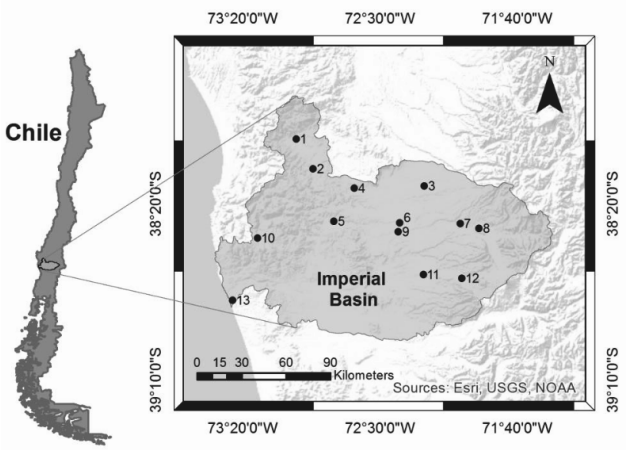


Figure 1: Imperial river basin located in Chile. The image show the area of study and the location of the 13 gauge stations used for this work.

The Imperial basin presents a pluvial hydrological regime. In other words, the water resource comes mainly for the rainfall. This characteristic is related to the low snow accumulation associated with the relative small altitude in the Andean mountain range on its latitude [11].

1.2 Data Description

The selected rainfall dataset covers the period from January 2003 up to December 2015 on a daily basis. The data is observed across 13 rain gauge stations dispersed across the area of study as seen is Fig. 1. This result in a total of 4748 data points for each station. The collected information is organised in 13 data frames, one for each station. All data frames contain the following data:

- The date on which the measurement was taken.
- The precipitation value in millimetres (mm) measured by the rain gauge (observed values).
- The SRE precipitation value in mm yearly recorded for the specific station area (SRE_annual).
- The SRE precipitation value in mm seasonally recorded for the specific station area (SRE_seasonal).

Table 1: Summary of the available rain gauge and SRE information for station 7.

	observed	SRE_annual	SRE_seasonal
1	Min. : 0.000	Min. : 0.000	Min. : 0.000
2	1st Qu.: 0.000	1st Qu.: 0.000	1st Qu.: 0.000
3	Median : 0.000	Median : 0.000	Median : 0.000
4	Mean : 2.489	Mean : 3.081	Mean : 2.933
5	3rd Qu.: 0.000	3rd Qu.: 3.082	3rd Qu.: 2.034
6	Max. :99.000	Max. :77.818	Max. :108.237
		NA's :4	NA's :3

The rain gauge measurements are used as the real precipitation value and the models will use it as its dependent variable. The SRE information is available in two formats annual and seasonal. The seasons are a quarterly subdivision of the year corresponding to December - February, March - May, June - August, and September - November. For each station the SRE which showed more closeness to the observed rainfall values on that season was selected. This means that a year of seasonal data consists of a mixture of several SREs. On the other hand, the SRE_annual data covers the whole year using only one satellite measurement, in this case all the annual data comes from the SRE PERSIANN-CDR. It is good to mention again that these satellites not necessarily represent an accurate reading of the studied area and the error in the estimates is still considerable.
Table 1 shows the summary of the data for station 7 as enumerated by the map in Fig 1. It can be seen that the majority of the data points are close to zero.

The SREs data is obtained from the public data set of the center for hydrometeorology and remote sensing (CHRS). The data from the ground stations are taken from the Chile center of climate science and resilience (CR). Included in the data there were four missing values all of which happened on the same days across all stations. We assumed this was due to a non-recurrent error on the SRE and omitted the missing data points from the analysis.

2 Gaussian Process

Gaussian process regression, also know as kriging, is an interpolation method that works by placing a prior distribution on an unknown function. This function is evaluated at the input data points $X = \{x_i\}_{i=1}^n$ and response observations $Y = \{y_i\}_{i=1}^n$ are collected. These observations are used to update the prior distribution into a posterior distribution by means of bayesian inference. In this sense, kriging assumes the observed points come from some multivariate Gaussian distribution and are related between them by the kernel function. The kernel function used in our study is given by

$$K(x, x') = exp(-\sum(\gamma_i |x_i - \hat{x}_i|^{p_i})) \tag{1}$$

The obtained posterior distribution distribution of y is given by a Gaussian process:

$$y|X, y \sim N(m(x), s^2(x)) \tag{2}$$

Where $m(x)$ is the maximum a posteriori probability at x and $s^2(x)$ is the estimations mean squared error.

2.1 Cluster Kriging

One of the major problems of Kriging is its high time and space complexity for large datasets. To help with this issue we divided the data set into smaller not overlapping clusters. For each cluster a Kriging model is built. A global posterior model is obtained by a weighted linear combination of the models as suggested by [13]. The global posterior is defined then as:

$$y|X, y \sim N(\sum_{i=1}^{q} w_i m_i(x), \sum_{i=1}^{q} w_i s_i^2(x)) \tag{3}$$

Where m_i and s_i represent the mean and variance of the model in the $i-th$ cluster. The weights w_i in the equation were calculated by minimizing the variance of the process [13].

$$w_i = s_i^{-2}(x) / \sum_{i=1}^{q} s_i^{-2}(x) \tag{4}$$

3 Bayesian Regression

The essential of the Bayesian methods is their use of probability for quantifying uncertainty in inferences based on statistical analysis [15]. Bayesian regression follow the Bayes' rule to make inferences from the data. Let y represent the observed precipitation values, x the SRE and θ a sequence of unknown parameters. We can infer the posterior distribution in terms of the likelihood and prior knowledge as shown in Eq. 5.

$$p(\theta|y) \propto p(y|\theta)p(\theta) \tag{5}$$

Given the nature of the data, we choose to use a logarithmic transformation on the observed values to allow the posterior to be sampled from a normal distribution. In these terms the likelihood $p(y|\theta)$ is defined as:

$$log(y+1) \sim N(\mu, \sigma) \tag{6}$$

The prior distributions for the parameters μ and σ are assumed to be normally distributed since not much previous information is known.

4 Experiments

The experiments were executed using SPOT [12] for the Gaussian process regression and rstan [14] for the Bayesian regression.

Two regression models were generated for each method, one using the SRE_annual data and other using the SRE_seasonal data. For each case the first twelve years of data were used as the training set and the last year was used as the validation set. The goodness-of-fit measures used to evaluate the models performance are the root mean square error (RMSE) and the Kling-Gupta efficiency (KGE). KGE is a goodness-of-fit measure obtained through the decomposition of the squared mean error [16]. This decomposition facilitates the analysis of the different components of the model: The correlation (r), the bias (β), and the relative variability in the simulated and observed values (α). All three components have their optima at unity. The KGE range from ∞ to 1 and its value is defined as shown in Eq. 7. The closer to 1 the more accurate the model is

$$KGE = 1 - \sqrt{(r-1)^2 + (\alpha-1)^2 + (\beta-1)^2} \tag{7}$$

For the Gaussian process regression the kernel terms γ and p from Eq. 1 were determined using the maximum likelihood method. The terms values are updated in every SPOT iteration. After some tests this approach was deemed to deliver better results than with static γ or p values.
To implement the cluster kriging the SRE_annual data set was divided into 20 clusters using the K-means method. The regression was run 10 times for each station. Each run had a different random seed to proof the stability of the system.
Since the SRE_seasonal data was already divided into subgroups, i.e. the seasons, the cluster kriging was not implemented here. In this case the analysis was also run 10 times for each station with different random seeds each.

The Bayesian regression was implemented with a logarithmic transformation on the response variable with the form:

$$ln(y+1)|x,\theta \sim N(\alpha+\beta*x,\sigma) \tag{8}$$

Uninformative proper priors are used for the parameters α, β and σ since no additional information is previously known.

$$\alpha \sim N(0,5), \qquad \beta \sim N(0,5), \qquad \sigma \sim cauchy(0.001,10) \tag{9}$$

The sampling of the posterior distribution of y is done using Hamiltonian Monte Carlo sampling. A total of 4 chains were generated for the posterior. After checking that all the chains converged and were accurate representations of the posterior distribution the number of iterations for each chain was set at 10000 including 2000 warmup iterations.

Before comparing the models we need to define a baseline against which to compare the models with. Since our goal is to approximate the SRE to the observed rain gauge values as much as possible, a model will be good when its RMSE and KGE values are better than those on the raw data. For example, for the annual model the baseline RMSE will be the RMSE between the station observed and SRE_annual values without any model.

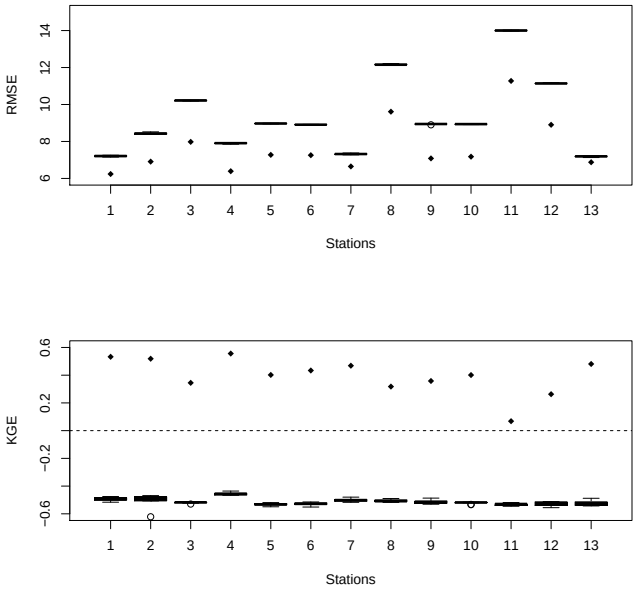


Figure 2: Boxplot of the RMSE and KGE values between the fitted annual Gaussian model response and the real rain gauge values respectively. The values are taken for all runs across all stations. The baseline RMSE and KGE values are depicted as diamonds. The results are stable between runs, yet the model had no improvement with respect to the baseline values.

4.1 Results Gaussian Process

The annual Gaussian process regression was done using the cluster kriging method. The RMSE and KGE fitness measures were computed for all stations in all 10 runs. The results indicated that the model was stable since there is no considerable difference between the different runs results. However, the model did not bring any improvement to the baseline SRE values. Fig. 2 illustrates this. The RMSE between the model fitted values and the real observed values for all runs across all stations is plotted against the baseline RMSE values, depicted in the image as diamonds. Every station shows a worse RMSE as that of the Baseline. The same is the case with the KGE, in all stations the baseline value is better than the model fitted values.

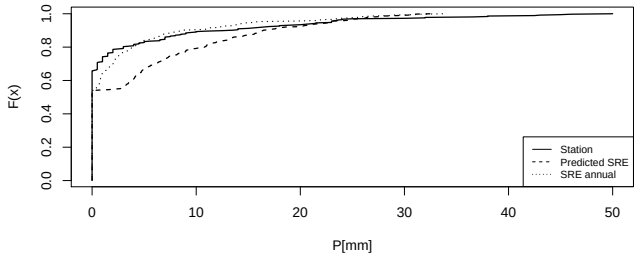


Figure 3: CDF of the validation set observed values of station 6. The Solid line indicate the CDF for the validation set observed values. The dashed line shows the annual Gaussian model predicted values. The dotted line are the validation set SRE_annual values. It can be seen that the model tends to underestimate the precipitation. The model predicted values and the SRE_annual values also fail to approximate the rain gauge maximum value.

The model doesn't seem to fit the data well but for completeness and to understand better the annual data behaviour we also analyse the model response on the validation set. As was expected based on the results on the training set ,the predicted model response across stations was steadily performing worse than the baseline SRE_annual. To illustrate the divergence between (1) the predicted values, (2) the observed values and (3) the SRE_annual values, we compare the three measurements cumulative distribution functions (CFD). Fig. 3 shows the CDFs of the three measurements for station 6 as an example. It was observed that the predicted values tend to underestimate the real precipitation before being truncated at a lower maximum value than the observed values.

Apart from the model fitted and predicted values, the model uncertainty was also explored. The standard deviation of the fitted values has a maximum value of $6.326e^{-01}$ while for the predicted values it has a maximum of $4.404e^{-02}$. This uncertainty values are well inside the tolerance limits for rainfall and thus the model uncertainty is not an issue.

The Gaussian process seasonal regression showed a considerable improvement in terms of RMSE and KGE fitness. The stability of the model runs was also improved with almost no identifiable difference between the different runs. For the training set Fig. 4 shows the fitted model

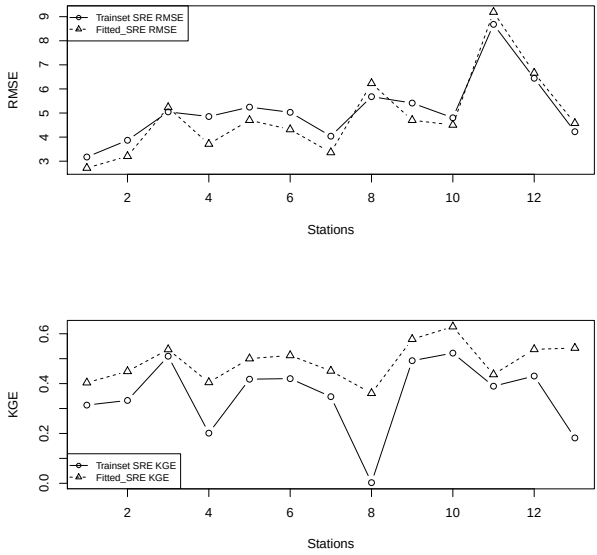


Figure 4: RMSE and KGE for the seasonal Gaussian model fitted values for one season across all stations. The baseline RMSE and KGE are depicted for comparison. For the most cases the model fitted values RMSE improved the baseline measurements and approximated the train set observed values better than the SRE_season data.

values RMSE and KGE for all stations for one season. Again the baseline RMSE and KGE values are illustrated as triangles in the figure. It can be seen that in almost all cases the model fitted values approximated better the station observed values than the SRE_season values did. It muss be mentioned that this behaviour was not stable for all seasons in terms of RMSE. There were some seasons that performed poorly against the baseline RMSE values. However, the KGE values was for all stations and all seasons always better than the baseline.

Since our main use for the fitted model values is on retrospective applications where the stations real values are always available, we will analyse more closely the model results on the training set. The training set observed values CDF for one season of station 2 can be seen on Fig. 5. The behaviour observed in the figure is replicated in the other stations. Here it can be seen that the model fitted values CDF follows the observed values closely despite the difference in RMSE previously noted.

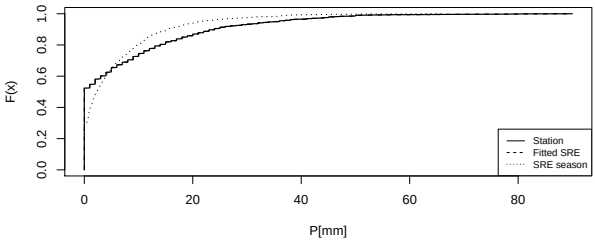


Figure 5: CDF of the training set observed values for station 2 in one season. The Solid line indicates the CDF for the observed values. The dotted line are the values for the validation set satellite measurements. Here the dashed line indicating the seasonal Gaussian model fitted values is not visible due to it overlapping the observed values in spite of the worse RMSE performance.

This performance improvement was also observed on the validation set where the model predicted values were able to predict the whole range of precipitation, contrary to the case observed in Fig. 3.

Evaluated on the training set the standard deviation of the fitted values had a maximum value of $8.444e^{-2}$. However, on the validation set the predicted values had a maximum deviation of up to 1.840. This value is worse than what we expected but still inside the tolerance limits for precipitation values.

4.2 Bayesian regression

The converge of all the chains in the Bayesian regression was controlled using three different criteria: The effective sample size (ESS), Rhat, and the Monte Carlo standard error (MCSE).
ESS gives insight to the autocorrelation within chains. A large value indicates there are enough independent samples in a chain. The Rhat criteria checks that the chains have converged to a common distribution by measuring the variance within and between chains. If the chains are not converging then its value will be greater than one. MCSE measures the uncertainty associated with the Monte Carlo approximation and has zero as its ideal value.

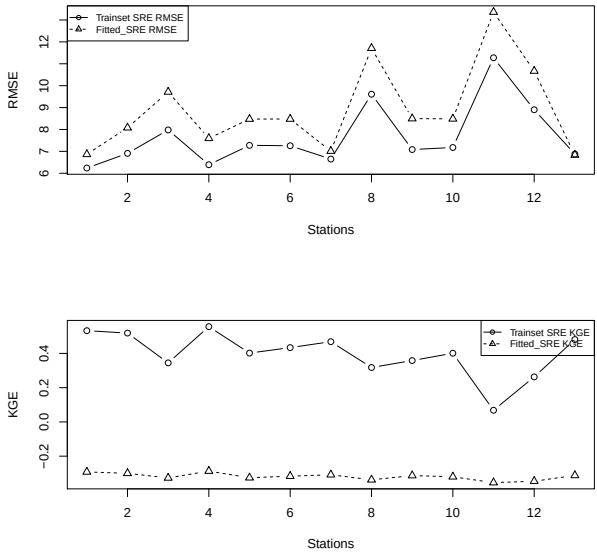


Figure 6: RMSE and KGE measurements of the Bayesian annual regression fitted values. The baseline fitness measurements are also included for comparison. The model performed worse for both fitness measurements. However, it improved in a small scale the KGE Gaussian process annual regression performance.

During the execution of the Hamiltonian Monte Carlo on all models there was no evidence to assume that the chains were not converging or that the Monte Carlo samples were not representative of the posterior distribution.

As with the Gaussian process regression we start by generating and analysing the model using the SRE_annual data. The model follows close in behaviour and improves in a small scale the Gaussian process annual results KGE measures. Fig. 6 shows the RMSE and KGE behaviour of the model against the baseline values for all stations. Here the model RMSE and KGE where computed using the mean of the 4 chains converged results.

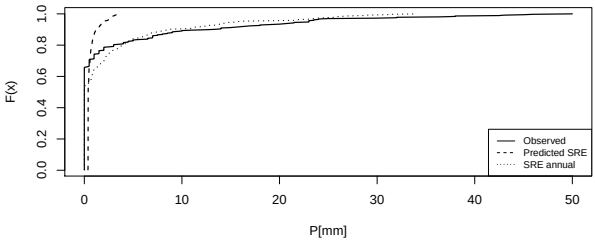


Figure 7: CDF of the validation set observed values for station 6. The solid line indicates the observed values CDF. The dotted line are the SRE_annual values. The dashed line are the annual Bayesian model predicted values. The model fails to predict non-zero values and is truncated on a lower precipitation value. This is due to the predicted posterior having its density centred at zero.

Again the CDF of the model predicted values was examined to find differences between the Bayes and Gaussian process regression on data forecasting. Fig. 7 shows the CDF of the observed, predicted and SRE_annual values on the validation set for station 6. The divergence of the model from the real data is clear. Since the posterior distribution of the model predicted values has most of its density around zero, it fails to predict non-zero values and has a maximum predicted precipitation value of only 4 mm.

Following the same criteria as exposed in the annual analysis, the convergence and representativeness of the chains for the seasonal regression was confirmed.
As was already seen on the Gaussian process regression analysis, the model behaviour across seasons changes. This implies that at some seasons the model fitted RMSE outperforms the baseline RMSE values. However, for all seasons the model fitted KGE was worse than the baseline in all stations, as seen in Fig. 8. This means that overall the model was not doing better that the SRE_season data.

Fig. 9 shows the divergence of the fitted values of the seasonal Bayes regression against the observed values on the training set. In the figure it can again be seen how the model fails to fit precipitation values not close to zero.

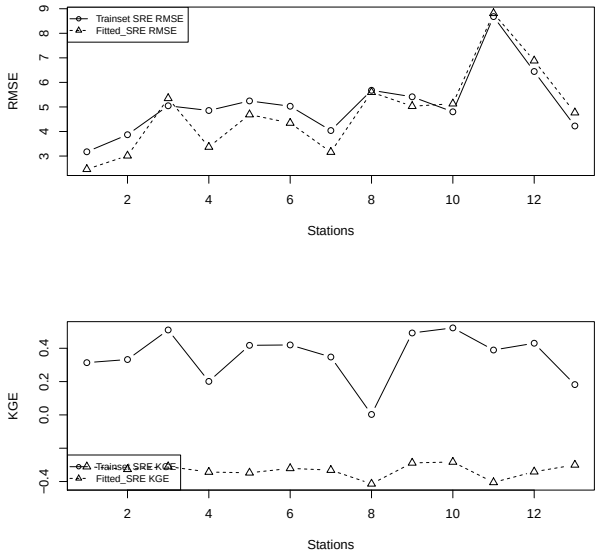


Figure 8: RMSE and KGE of the fitted seasonal bayesian model values for one season across all stations. The baseline RMSE and KGE are also showed for comparison. Even though some stations showed an improved performance on the fitted RMSE values, the KGE performance was consistently worse.

5 Conclusions

The aim of this work was to implement and test a bias correction method for satellite rainfall estimates on the Imperial basin in Chile. This was encouraged by the low accuracy SREs can have under some space and time conditions. We tested two regression methods on two SRE data . The first SRE data class was annual data, this data was collected using only one SRE. The second SRE data class was seasonal data, this data consisted on a combination of SREs. These two data classes were used to test the data aggregation procedures implemented to fit the SRE time scales (annual and seasonal). The two tested regression models were Gaussian process and Bayesian regression. From our analysis it was clear that any bias correction needed to be carried out using seasonal data. Since the combination of two or more SRE improved the rainfall estimates in the varying conditions of the seasons.

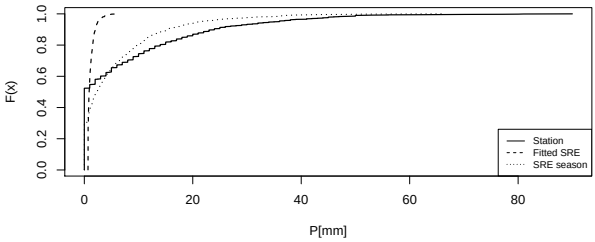


Figure 9: CDF of the validation set observed values of station 6. The Solid line indicates the CDF of the validation set observed values. The dashed line shows the seasonal Bayes model predicted values. The dotted line are the validation set SRE_season values. It is apparent that the model fails to predict precipitation values ranges away from zero.

Overall Gaussian process regression delivered better and more accurate results when working with seasonal data while the Gaussian process annual model showed no improvement at all. There are some factors that could have contributed to the lack of improvement in the annual Gaussian process regression. First, the use of cluster kriging may have introduced an unknown amount of error into the annual model. This error could also have been aggravated by the varying size of the clusters. It is possible that the use of a cluster method that generates fix sized non-overlapping clusters could bring some improvement to the annual analysis. This however needs to come in hand with a better SRE.

Even though the Bayesian regression did not show satisfying results it was interesting to discover that it was considerably faster that the Gaussian regression. Given that the intention of the bias correction model is for it to be used in bigger areas with denser rain gauge networks, the speed in which the model can be generated is important. Taking this into account, it is in our interests to keep exploring the possibility of implementing the bias correction using the Bayesian regression. For this we propose future experiments using hurdle or zero-inflated models that allow to have separate distributions for zero and non-zero values.

The Bayesian regression results also highlighted the differences between RMSE and KGE measures of fitness. Where the RMSE for the Bayesian annual model showed no improvement, the KGE showed an small improvement. The same was the case with the seasonal Bayesian model where the RMSE showed an improvement but KGE steadily showed a

worse performance. It is clear that in the latter case the RMSE was overestimating the model skill. This can be due to the choice of baseline with which the models are compared. A more detailed inspection of the KGE components showed an improvement in the bias (β) values for the Bayes regression models, while the correlation and variability did not show any improvement.

References

[1] M. Zambrano-Bigiarini, A. Nauditt, C. Birkel, K. Verbist, L. Ribbe. "Temporal and spatial evaluation of satellite-based rainfall estimates across the complex topographical and climatic gradients of Chile". In: *Hydrology and Earth System Sciences*. 21.2. 2017.

[2] C. Berndt, E. Rabiei, U. Haberlandt. " Geostatistical merging of rain gauge and radar data for high temporal resolutions and various station density scenarios". In: *Journal of Hydrology* 508. 2014.

[3] T.K. Chang, A. Talei, S. Alaghmand, M.P. Ooi. "Choice of rainfall inputs for event-based rainfall-runoff modeling in a catchment with multiple rainfall stations using data-driven techniques". In: *Journal of Hydrology* 545. 2017.

[4] M. Gebremichael, E.N. Anagnostou, M.M. Bitew. "Critical Steps for Continuing Advancement of Satellite Rainfall Applications for Surface Hydrology in the Nile River Basin". In: *jJAWRA Journal of The American Water Resources Assosiation* 46.2. 2010.

[5] H.E. Beck, A.I.J.M. van Dijk, V. Levizzani, J. Schellekens, D.G. Miralles, B. Martens, A. de Roo. "MSWEP: 3-hourly 0.25° global gridded precipitation (1979–2015) by merging gauge, satellite, and reanalysis data". In: *Hydrology and Earth System Sciences* 21. 2017.

[6] R. Joyce, J.E. Janowiak, P.A. Arkin, P. Xie. "CMORPH: A Method that Produces Global Precipitation Estimates fron Passive Microwave and Infrared Data at High Spatial and Temporal Resolution". In: *Journal of Hydrometeorology* 5.3. 2004.

[7] K.L. Hsu, X. Gao, S. Sorooshian, H.V. Gupta. "Precipitation estimation from remotely sensed information using artificial neural networks". In: *Journal of Applied Meteorology* 36.9. 1997.

[8] H. Ashouri, K.L. Hsu, S. Sorooshian, D.K. Braithwaite, K.R. Knapp, L.D. Cecil, B.R. Nelson, O.P. Prat. "PERSIANN-CDR: Daily precipitation climate data record from multisatellite observations for hydrological and climate studies". In: *Bulletin of the American Meteorological Society* 96.1. 2015.

[9] G.J Huffman, D.T. Bolvin, E.J. Nelkin, D.B. Wolff, R.F. Adler, G. Gu, Y. Hong, K.P. Bowman, E.F. Stocker. "The TRMM multisatellite precipitation analysis (TMPA): Quasi-global, multiyear, combined-sensor precipitation estimates at fine scales". In: *Journal of Hydrometeorology* 8.1. 2007.

[10] C. Funk, P. Peterson, M. Landsfeld, D. Pedreros, J. Verdi, S. Shukla, G. Husak, J. Rowland, L. Harrison, A. Hoell, J. Michaelsen. "The climate hazards infrared precipitation with stations?a new environmental record for monitoring extremes". In: *Scientific Data 2*. 2015.

[11] N.R. Rivera, F. Encima, A. Muñoz-Pedreros, P. Mejias. "Water Quality in the Cautín and Imperial Rivers, IX Region-Chile". In: *Información Technológica* 15.5. 2004.

[12] T. Bartz-Beielstein, C. Lasarczyk, M. Preuss "Sequential Parameter Optimization". In: *IEEE Congress on evolutionary computation.* 2005.

[13] H. Wang, B. van Stein, M. Emmerich, T. Bäck "Time complexity reduction in efficient global optimization using cluster kriging". In: *Proceedings of the Genetic and Evolutionary Computation Conference. ACM.* 2017

[14] Stan Development Team. "RStan: the interface for Stan in R" Package version 2.16.2 *http://mc-stan.org* 2017

[15] A. Gelman, J.B. Carlin, H.S. Stern, D.B. Dunson, A. Vehtari, D.B Rubin "Bayesian Data Analysis". *Chapman &Hall/CTC Press, Third edition* 2013

[16] H.V. Gupta, H. Kling, K.K. Yilmaz, G.F. Martinez. "Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling". In: *Journal of Hydrology* 377. 2009.

Conflict-based Feature Selection for Information Fusion Systems

Christoph-Alexander Holst[1], Uwe Mönks[2], and Volker Lohweg[1]

[1]inIT - Institute Industrial IT, Ostwestfalen-Lippe University of Applied Sciences,
Langenbruch 6, 32657 Lemgo, Germany
E-Mail: {christoph-alexander.holst, volker.lohweg}@hs-owl.de
[2]coverno GmbH, Langenbruch 6, 32657 Lemgo, Germany
E-Mail: uwe.moenks@coverno.de

Abstract

Applying information fusion systems aims at gaining information of higher quality and simultaneously decreasing computational and communicational efforts. An increased availability of sensors in industrial machines, but also in everyday life, results in large amounts of potential features. Each feature entails computational and communicational costs. An information fusion system may not require all features, supported by the available sensors, to fulfil its purpose. Feature selection methods reduce the amount of features with the aim to maintain or even increase performance. This contribution proposes a feature selection approach exploiting the inherent conflict between features and utilising a state-of-the-art information fusion operator. The performance of the proposed method is evaluated in the scope of a publicly available data set and benchmarked against an established feature selection method. It is shown that the proposed approach is faster and produces more accurate feature subsets containing very few features, although the established method produces slightly better performing subsets for large feature subsets.

1 Introduction

Information fusion gathers and combines information from different sources resulting in information of higher quality, less complexity, and less uncertainty. It is applied in industrial systems, e. g., for condition monitoring, but also in end-user devices, such as smartphones or navigation

systems, to decrease computational and communication efforts, and to represent information in a human-comprehensible form. Information fusion involves four steps: namely (i) acquisition of signals from different sources, (ii) extraction of an appropriate feature set, (iii) classification of the features, and (iv) decision if the obtained results represent the expectations [1]. Designing such information fusion systems including selection of appropriate features is an extensively manual task. This task is becoming increasingly complex as industrial machines as well as end-user devices are equipped with increasing amounts of sensors, which is directly related to an increase in the number of potential features. Each feature entails a cost for information fusion systems; it needs to be extracted from the sensor signal, needs to be communicated between processing units in the case of distributed systems, and increases the complexity of the fusion process itself. The set of all features may include irrelevant or redundant features, contributing little to the classification result of an information fusion system. Thus, an information fusion system may perform similarly or even better relying only on a subset of the available features.

Feature selection aims at reducing the amount of features utilised for a classification process, without losing quality or at best even improving the quality of classification results [2]. Potential advantages are an improvement of the classification performance, a decrease in computation time and effort, a reduction in memory and communication requirements, and a better comprehension of the data-generating process [3]. A drawback is that feature selection algorithms themselves are often computationally complex, i. e. they grow often exponentially, and, thus, are time-consuming if many features exist. In contrast, modern industrial machines need to adapt frequently which requires an information fusion system to quickly reorganise itself and relearn its classification algorithms. Thus, demands on the performance of industrial machines may not allow for time-consuming feature selections.

This contribution proposes a fast feature selection algorithm intended for the use in information fusion systems. The focus is on condition monitoring scenarios, applicable both to industrial systems as well as end-user devices, i. e., two-class classification problems. The selection algorithm evaluates and ranks features according to their *individual conflicting coefficient*, which is introduced in this paper, and the distance of their fuzzy memberships to the trained classes. It creates feature subsets by gradually excluding features with high conflict coefficients

and high distances. All found feature subsets serve as search space for an optimisation algorithm, which is wrapped around the *balanced two-layer conflict solving* (BalTLCS) fusion operator [4]. The optimisation algorithm searches for the subset with maximum classification accuracy of the BalTLCS fusion output. Exploring only the subsets selected by the ranking algorithm is significantly less complex than searching all possible combinations of features. This results in a fast selection algorithm suitable for applications which require adaptable systems.

This paper is structured as follows. In Sec. 2 related work is discussed. Section 3 presents the proposed feature selection approach. Following, the proposed approach is evaluated in comparison to an existing popular feature selection algorithm w. r. t. to a publicly available data set in Sec. 4. Finally, this contribution closes with a conclusion in Sec. 5.

2 Related Work

Feature selection is an active research field for more than 60 years. An increasing availability of features and data instances in modern machines makes feature selection even more important and ensures that it stays actively researched.

Two main approaches for feature selection are identified in [5]: *filter* and *wrapper* methods. Filters rank and select features individually. Features are filtered out which are unlikely to improve the classification output. Filtering is carried out independently from the classifier in a preprocessing step. Filters are computationally fast, but often do not consider inter-correlations between features and, thus, tend to select redundant features. Furthermore, it is not guaranteed that filters select features that perform well in conjunction with the intended classifier [6]. Filters assign each feature a relevance index. Features are then ranked by this index and low-ranked features are filtered out. Popular individual relevance indices are the *pearson correlation coefficient*, *mutual information* [7, 2], and the *Fisher coefficient* [8]. The pearson correlation coefficient determines the statistical correlation between two variables (feature and class), whereas mutual information describes the amount of information two variables share. Therefore, they are only applicable to two-class problems. The Fisher coefficient takes inter-class variances into account and, thus, is suitable for multi-class problems. All three ranking criteria do not incorporate the inter-correlation of features [8, 2].

One of the most well-known filter algorithms is *Relief* proposed in [9, 10]. Relief estimates the relevance of a feature by its ability to distinguish between nearby instances of classes. The feature's value of an instance $f(\theta)$ is compared to the nearest instance of the same class $f(\theta)_\mathrm{s}$ (nearest hit) and the nearest instance of a different class $f(\theta)_\mathrm{d}$ (nearest miss). The relevance of a feature is increased proportionally to $|f(\theta) - f(\theta)_\mathrm{d}|$ and decreased proportionally to $|f(\theta) - f(\theta)_\mathrm{s}|$. The Relief algorithm also does not consider inter-correlations. A further downside of Relief is that it requires a manually defined threshold to select features [2]. Relief is limited to two-class problems. The algorithm *ReliefF* proposed in [11] extends Relief to multi-class problems.

Wrapper algorithms incorporate the classifier to select complete feature subsets, in contrast to filter approaches. Hence, they inherently consider correlations between features in a subset. Wrappers are often computationally expensive and time-consuming because they explore a search space consisting of all possible subsets of available features.

The success of wrapper approaches depends mostly on the applied search strategy. An exhaustive search guarantees to find the optimal feature subset, but is in most real-world scenarios infeasible [7]. Another search strategy, the *branch-and-bound* algorithm originally proposed in [12], reduces the search space gradually and, hence, has the potential to run a lot faster than exhaustive searches. Drawbacks to the branch-and-bound search are that it searches only for a subset of specific size, that it still has an exponential complexity in worst case, and that it requires the evaluation function of the subsets to be monotonic, i. e. adding a feature does not worsen the performance of the subset. Especially because of the requirement of monotonicity is the branch-and-bound search ill-suited for many real-world applications. *Greedy* search strategies, such as *forward selection* and *backward selection*, show a quadratic computational complexity but risk to get stuck in local optima [6]. These selection algorithms add or remove, respectively, the best or worse feature one at a time. A more recent promising field of search strategies are evolutionary approaches. Though, Xue et al. [13] conclude in their survey that evolutionary approaches are to date too computationally expensive for high numbers of features or instances. However, they also note that further research may improve evolutionary approaches.

Wrapper approaches are, e. g., applied in [14] and [15]. Guyon et al. [14] select the most relevant genes for cancer classification. They apply a backward selection algorithm as search strategy and evaluate feature

subsets with a *support vector machine*. Bator et al. [15] propose a heuristic wrapper approach based on the *linear discriminant analysis* (LDA) to evaluate subsets of features. They search for a feature subset of minimum size which is able to solve a multi-class classification problem. They apply a bottom-up exhaustive search, i. e. feature subsets of size 2 are considered first, but stop as soon as they found a subset yielding an error smaller than a predetermined threshold. Thus, the complete power set of features is only searched in the worst case scenario; though, depending on the application, this method is computational demanding.

Exploiting conflict as a measure of inter-correlation for feature selection is a novel idea proposed in this contribution. Conflict occurs when at least one information source, i. e. feature, contradicts the remaining information sources [17]. Conflict is a form of conscious ignorance, because it is not apparent which source represents the actual situation correctly. If conflict is not recognised and properly addressed during information processing, it leads to inconsistent and incorrect fusion results [18]. Thus, conflict is usually a disturbance which needs to be handled. Works which particularly exploit conflict are rare. For example, Ehlenbröker et al. [19] utilise conflict as a consistency measure to detect sensor defects in a group of sensors.

The *balanced two-layer conflict solving* [4] is an information fusion operator, which is based on *Dempster-Shafer Theory* (DST) [16], and is optimised to handle high-conflicting situations. It assigns every feature F for each data instance a *basic belief assignment* (BBA), modelling the belief of F into a proposition. Fusion is carried out by aggregating the BBAs of every pairwise combination of features. This results in two outputs; the fused information result itself plus an importance measure, both in $[0, 1]$. The importance is the complement of the conflict between all input features. The BalTLCS operator can be utilised as a classifier for two-class problems by setting a discriminating threshold. In fact, BalTLCS has been successfully applied to condition monitoring applications [17].

In conclusion, feature selection approaches are often either computationally expensive or do not take inter-correlations of features into consideration. In the current state-of-the-art, no concept, approach or implementation is available relying on conflict between features as a measurement of inter-correlation. Therefore, this contribution proposes a feature selection method utilising conflict as relevance index and

the BalTLCS operator as evaluating classifier. This feature selection approach is presented in the next section.

3 Approach

In this section a backward feature selection method is proposed, which is wrapped around the *balanced two-layer conflict solving* (BalTLCS) information fusion operator. In this approach features are rated and ranked according to their average fuzzy memberships (correlation to class) and their conflict (inter-correlation between features) over a labelled training data set. A feature subset is selected based on these rankings by eliminating the worst ranked features one after another. The feature subsets are rated regarding their accuracy to classify a test data set in conjunction with the BalTLCS operator.

The proposed approach is developed w. r. t. to a condition monitoring scenario. Thus, a two-class classification problem is assumed, of which the two classes represent the *normal condition* N and the *abnormal condition* $\overline{\mathrm{N}}$ of the monitored system. A system in its normal condition N functions as intended. Deviating from its intended operating conditions, a system either is unable to fulfil its task or gets damaged in the process. Therefore, it operates in its abnormal condition $\overline{\mathrm{N}}$.

With that in mind, in the following Sec. 3.1 the method of ranking features and grouping them to subsets is detailed. Section 3.2 then specifies the method of selecting the optimal subset out of the created subsets in the ranking process.

3.1 Creating Feature Subsets by Ranking Features

The first of two ranking criteria is the distance of a data instance to the current condition of the monitored system. It is based on the *Modified Fuzzy Pattern Classifier* (MFPC), introduced in [20], and is therefore referred to as *fuzzy membership distance*. The MFPC utilises a unimodal potential function as fuzzy membership function $\mu : \mathbb{R} \to [0, 1]$. Regarding a feature F_f, the fuzzy set representing the normal condition of a system is modelled by the membership function ${}^{\mathrm{N}}\mu_f : \mathbb{R} \to [0, 1]$. The fuzzy membership ${}^{\mathrm{N}}\mu_f(\theta)$ of a specific data instance θ then determines the degree to which θ represents the normal condition. The membership

function of the abnormal condition is complementary to ${}^{\mathrm{N}}\mu_f$, i. e. ${}^{\overline{\mathrm{N}}}\mu_f = 1 - {}^{\mathrm{N}}\mu_f$. The fuzzy membership distance is then defined as follows:

Definition 1 (Fuzzy membership distance)**.** Let $A \subset \{\mathrm{N}, \overline{\mathrm{N}}\}$ be a proposition from the universal set, $\theta \in A$ be a datum measurement of a feature F_f, and ${}^{\mathrm{N}}\mu_f$ the membership to the fuzzy set modelling the normal condition N, then the fuzzy membership distance is defined by

$$d_f(\theta) = \begin{cases} 1 - {}^{\mathrm{N}}\mu_f(\theta) & \text{if } \theta \in \mathrm{N} \ , \\ {}^{\mathrm{N}}\mu_f(\theta) & \text{if } \theta \in \overline{\mathrm{N}} \ . \end{cases}$$

The overall distance of a feature, which is utilised as ranking criterion, is determined by

$$\delta_f = \frac{1}{m} \sum_{j=1}^{m} d_f(\theta_j) \ , \tag{1}$$

where m is the number of data instances in the training data set.

The distance measurement, on its own, does not consider potential redundancy of features. But, as stated in [21], a feature is relevant to a classification task if it is not redundant, i. e. a feature is not highly correlated to other selected features. Selecting redundant features may lead to a situation, in which features are neglected although they indicate a specific abnormal condition uniquely. Thus, if the system is in an abnormal condition, features are favourable which show unique characteristics. This is not the case if the observed system is in its normal condition. Since the membership function of a feature is trained on the normal condition, a relevant feature is expected to have high memberships during the normal condition in test or validation data as well.

In this contribution the *individual conflicting coefficient* is proposed to determine the uniqueness of a feature. It is applied as the second of the two ranking criteria. The coefficient expands on the *normalised conflicting coefficient*, which is introduced in [17] and utilised in the BalTLCS fusion operator. It is determined as

$$c(\theta) = \frac{1}{\binom{n}{2}} \sum_{f=1}^{n-1} \sum_{g=f+1}^{n} \sum_{i=1}^{o} {}^{A_i}\mu_f(\theta) \cdot (1 - {}^{A_i}\mu_g(\theta)) \ , \tag{2}$$

where f and g are indices of features, n the number of features, and o the number of possible propositions. In the case of a condition monitoring scenario $o = 2$, $A_1 = \mathrm{N}$, and $A_2 = \overline{\mathrm{N}}$ (or vice versa). The normalised

conflicting coefficient assesses the overall conflict present in a group of features by considering all features pairwise.

This approach proposes to alter Eq. (2) in such a way that the conflict of an individual feature rather than the overall conflict is determined by introducing the *individual conflicting coefficient* as follows:

Definition 2 (Individual conflicting coefficient)**.** Let $A_i \subset \{\mathrm{N}, \overline{\mathrm{N}}\}$ be a proposition for the normal or abnormal condition of a system, and $\theta \in A$ be a datum measurement of a feature F_f, then the degree of conflict between the fuzzy membership of a single feature $F_f \in \mathbf{F}$ compared to all other features $F_g \in \mathbf{F} \backslash F_f$ is determined by

$$c_f^{\mathrm{i}}(\theta) = \frac{1}{n-1} \sum_{\substack{g=1| \\ g \neq f}}^{n} \sum_{i=1}^{2} {}^{A_i}\mu_f(\theta) \left(1 - {}^{A_i}\mu_g(\theta)\right) \ . \tag{3}$$

The conflicting coefficient is a real number inside the unit interval ($c_f^{\mathrm{i}} \in [0,1]$) with $c_f^{\mathrm{i}} = 0$ stating no conflict and $c_f^{\mathrm{i}} = 1$ stating maximum conflict.

In the following, Eq. (3) is simplified utilising that $\sum_{i=1}^{2} {}^{A_i}\mu_f(\theta) = 1$. This leads to

$$c_f^{\mathrm{i}}(\theta) = 1 - \frac{1}{n-1} \sum_{\substack{g=1| \\ g \neq f}}^{n} \sum_{i=1}^{2} {}^{A_i}\mu_f(\theta) \cdot {}^{A_i}\mu_g(\theta) \ . \tag{4}$$

In Eq. (4) A_1 is replaced with N and A_2 with $\overline{\mathrm{N}}$. Further simplifications lead to

$$c_f^{\mathrm{i}}(\theta) = {}^{\mathrm{N}}\mu_f(\theta) - \frac{2 \cdot {}^{\mathrm{N}}\mu_f(\theta) - 1}{n-1} \sum_{\substack{g=1| \\ g \neq f}}^{n} {}^{\mathrm{N}}\mu_g(\theta) \ .$$

Similar to the distance criterion in Eq. (1), the individual conflicting coefficient is averaged over the training data set applying the arithmetic mean. As previously mentioned, features are desirable which are uncorrelated (showing a high conflict) during the abnormal condition of a system and which are highly correlated (showing a low conflict) during the normal condition. Thus, the individual conflict is averaged as follows:

$$\gamma_f = \frac{1}{m} \sum_{j=1}^{m} \begin{cases} c_f^{\mathrm{i}}(\theta_j) & \text{if } \theta_j \in \mathrm{N} \ , \\ 1 - c_f^{\mathrm{i}}(\theta_j) & \text{if } \theta_j \in \overline{\mathrm{N}} \ . \end{cases} \tag{5}$$

Algorithm 1.: Creating set of feature subsets by backward selection

Input: $\mathbf{F}$: set of features

```
1: function createFeatureSubsets(F, 𝔽)
2:     F′ ← F
3:     𝔽.append(F′)
4:     for number of features in F do
5:         for every F_f ∈ F′ do
6:             compute relevance index w_f^r
7:         remove feature with minimal relevance index from F′
8:         𝔽.append(F′)
```

Output: $\mathbb{F}$: set of feature subsets

Distance (see Eq. (1)) and conflict (see Eq. (5)) are then combined into a relevance index

$$w_f^{\mathrm{r}} = \delta_f \cdot \gamma_f \ . \tag{6}$$

All features are ranked according to their relevance index w_f^{r}. An index w_f^{r} close to 0 indicates a favourable feature, whereas an index close to 1 implies an unfavourable feature.

Based on the relevance indices of all features, a backward selection algorithm creates feature subsets iteratively. First, the lowest ranking feature is removed from the original set of feature $\mathbf{F}$. In the process the first feature subset $\mathbf{F}'$ is created. The features in this new subset are then re-evaluated regarding their relevance index utilising Eq. (6) again. This is necessary because, by removing a feature, conflict changes. Ranking features and removing the lowest ranked feature repeats until only two features are left. The output of the backward selection algorithm is a set of feature subsets $\mathbb{F}$. The algorithm is shown in Listing 1.

3.2 Selecting a Feature Subset

In this section a method is proposed to assess the quality of the found subsets $\mathbf{F} \in \mathbb{F}$ and to select the optimal one. The quality of a feature subset is determined by its classification accuracy over a validation data set. The classification is carried out utilising the BalTLCS information fusion operator. The BalTLCS fusion outputs a value in the interval $[0, 1]$ denoted as system health. If the system health for a data instance is

above a manually defined threshold, then the observed system is regarded as working in normal condition.

The accuracy of a feature set relies on the number of *true positive* (*tp*), *true negative* (*tn*), *false positive* (*fp*), and *false negative* (*fn*) classifications. It represents the average performance of the classification and is determined as follows:

$$accuracy = \frac{tp + tn}{tp + fp + tn + fn} \ .$$

An exhaustive search over $\mathbb{F}$ results in finding the optimal $\mathbf{F} \subseteq \mathbb{F}$, i. e. the feature set with maximum accuracy. In case of large feature sets, an exhaustive search may be unfavourable or even infeasible. Search algorithms which identify promising areas inside the search space and focus on these specific areas reduce the search time. For this approach, a heuristic direct random search algorithm based on the *Luus-Jaakola-algorithm* [22] is proposed, which is shown in Listing 2. Direct search algorithms are favourable for problems in which no gradient of the objective function is available [23], though they may get stuck in a local optimum or may not converge to the optimal solution [24].

The Luus-Jaakola-algorithm selects candidate solutions (feature subsets) randomly in a region around the currently known best solution. If a candidate proves to be a better solution (higher accuracy), it selects this candidate as current best solution. If the candidate is a worse solution, the search region is reduced. The algorithm terminates as soon as the search region is smaller than a stopping criterion. This stopping criterion, the rate at which the search region is reduced, and the starting size of the search region need to be specified manually.

The Luus-Jaakola-algorithm is modified by implementing a *tabu*-list comprising all already visited candidate solutions. Candidates listed as *tabu* are not considered as solution again. The algorithm accepts only better candidates as a new solution as it moves through the search space. It does not accept worse solutions, so it is not necessary to re-evaluate already visited candidates. The tabu-list reduces computation time of the algorithm.

The proposed approach for a conflict-based feature selection wrapped around the BalTLCS fusion operator is evaluated in the next section.

Algorithm 2.: Random heuristic search for best feature subset in $\mathbb{F}$ based on [22]

Input: $\mathbb{F}$: Set of feature subsets

function RANDOMSEARCH($\mathbb{F}$)
  $n = size(\mathbb{F})$
  $range = 1$
  $red \in]0, 1[$ ▷ Constant with which *range* decreases.
  $stopcriterion \in]0, 1]$
  Select a feature subset $\mathbf{F}_b \in \mathbb{F}$ randomly
  while $range > stopcriterion$ **do**
    Select a non-tabu new feature subset randomly:
    $\mathbf{F}_i \in \mathbb{F} \mid i \in [b - n \cdot range, b + n \cdot range]$
    Put $\mathbf{F}_i$ in tabu-list
    if $accuracy(\mathbf{F}_i) > accuracy(\mathbf{F}_b)$ **then**
      $\mathbf{F}_b = \mathbf{F}_i$ ▷ Current feature set is new best.
    else
      $range = red \cdot range$ ▷ No improvement results in a smaller search range.

Output: $\mathbf{F}_b$: Best found feature subset in $\mathbb{F}$

4 Evaluation

The performance of the proposed conflict-based feature selection is evaluated w. r. t. to the *Human Activity Recognition Using Smartphones* (HAR) data set published in [25]. The HAR data set consists of 561 features extracted from 17 sensor signals of a smartphone (e. g., accelerometers, gyroscopes) worn at the waist of human beings. The data set is classified into dynamic (e. g., walking) and static (e. g., standing, sitting) activity of the observed person. It is partitioned into training and test data sets for dynamic and static activity, respectively.

First, the accuracy of all feature subsets created by the conflict-based selection algorithm are compared to subsets created by the ReliefF algorithm [11] in Sec. 4.1. Afterwards, the performance of the proposed search algorithm is evaluated in Sec. 4.2.

4.1 Benchmark of the Conflict-based Feature Selection

In the following the proposed conflict-based feature subset selection approach (see Listing 1) is benchmarked against the ReliefF algorithm

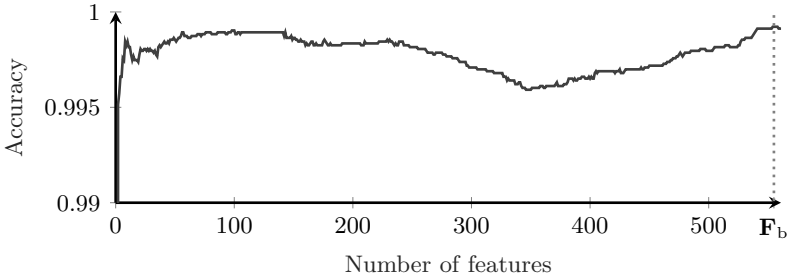


Figure 1: The plot shows the accuracy of the feature subsets created by the proposed conflict-based feature selection. Accuracies are obtained by inputting the feature subsets into the BalTLCS operator. The best accuracy of 0.9992 is achieved for the feature sets $\mathbf{F}_b$ with $b \in \{554, 555, 556, 557, 558\}$ (dotted line).

[11]. In addition, it is compared to a random selection strategy. The results obtained by the proposed approach are presented first. Afterwards these results are compared to the ReliefF and the random approach. It is shown how accurate the created subsets are in classifying the condition of the monitored person in the HAR data set. The accuracy of a subset is determined with BalTLCS.

Certain parameters of the MFPC are to be set manually for each membership function. These are the slope steepness D and the percental elementary fuzziness $p_{\mathrm{c_e}}$, which are set identically for each membership function to $D = 4$ and $p_{\mathrm{c_e}} = 0.075$. For the BalTLCS, a threshold needs to be specified, which determines if an instance represents the normal condition (dynamic activity). A membership of an instance above this threshold indicates that the instance represents the normal condition. This threshold is set to 0.95.

The results of the proposed conflict-based subset selection approach are depicted in Fig. 1. The best accuracy of 0.9992 is achieved for subsets comprising 554 to 558 features. Removing features from thereon worsens the accuracy to 0.9959 at 350 to 346 features. The subset with 10 features is a local optimum with an accuracy of 0.9984.

Figure 2 shows additionally the accuracy of subsets created with the ReliefF algorithm and randomly selected subsets. The conflict-based selection as well as the selection by ReliefF outperform the randomly selected subsets below 330 features. The ReliefF algorithm outputs subsets which show a slightly better accuracy than the subsets created

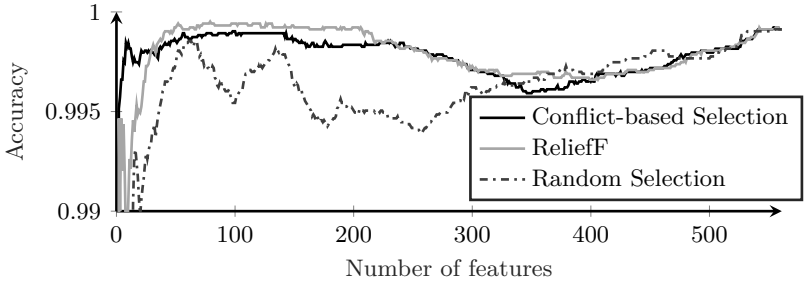


Figure 2: The accuracies of the feature subsets created by the conflict-based selection, the ReliefF algorithm, and randomly created subsets. The accuracy of conflict-based subsets are shown in black, ReliefF subsets in grey, and randomly selected subsets in a dash-dotted line.

with the conflict-based approach except for subsets with very few features. For subsets with less than 30 features the proposed approach performs best. The best performing subset is created by the ReliefF algorithm with an accuracy of 0.9995 at 79 features.

As motivated in Sec. 1, the required computation time of a feature selection algorithm is of importance. The computation time is evaluated in required time in seconds on an Intel Core i7-3820 CPU including four processing cores running at 3.6GHz. Applied to the HAR data set, the conflict-based feature selection requires $t_\mathrm{c} = 32.4$s to create all feature subsets, whereas the ReliefF algorithm takes $t_\mathrm{R} = 368.9$s. This results in a *speed-up* $s = \frac{t_R}{t_s} = 11.39$.

4.2 Performance of the Search Algorithm

In addition to the evaluation w. r. t. the ReliefF algorithm, the search algorithm proposed in Listing 2 is discussed in this section. The search algorithm takes the set of feature subsets created by the conflict-based selection as input, as discussed in the previous section, and searches for the subset with maximum accuracy. The parameters *range*, *red*, and *stopcriterion* are set manually as follows: $range = 1$, $red = 0.95$, $stopcriterion = 0.005$.

With these parameters the search algorithm outputs the subset with 554 features and an accuracy of 0.9992. It takes 47 iterations to achieve the

result which requires $t_o = 23.32$s. The algorithm started with the feature subset containing 104 features.

An exhaustive search over the complete $\mathbb{F}$ takes $t_{ex} = 223.14$s. Thus, the proposed search algorithm achieves a speed-up $s = \frac{t_{ex}}{t_o} = 9.57$. Drawbacks of the proposed algorithm are (i) that the result and computation time depend on the randomly chosen starting feature set, (ii) that it may only find a local optimum, and (iii) that it requires parameters to be specified manually.

5 Conclusion

Modern industrial systems are becoming equipped with an increasing number of sensors supporting large amounts of features. This may include features of little relevance, which impede especially distributed information fusion systems and systems which need to be highly adaptable. This contribution proposes a fast feature selection algorithm combining ideas from the filter and wrapper approaches. The filter part of the approach evaluates and ranks features according to their average fuzzy membership distance and their conflict to the remaining features. Features with high distance and high conflict during the abnormal condition of a system are rated as less relevant. The filter subsets $\mathbb{F}$, created by the filter part, are then input into the BalTLCS information fusion operator to assess their accuracy on a given data set. A search algorithm is proposed to find the best feature subset in $\mathbb{F}$. The proposed approach is evaluated w. r. t. to the ReliefF algorithm [11] in the scope of the publicly available HAR data set [25].

It is shown that the feature subsets created by the conflict-based feature selection perform better for very low numbers of features and slightly worse for high numbers of features than subsets created by ReliefF. In this evaluation, the Relieff algorithm creates the subset with the highest overall accuracy (0.9995 by Relieff to 0.9992 by the conflict-based ranking). The accuracy of feature subsets with very few features created by the proposed approach is higher than Relieff's subsets with the same amount of features, e. g., 0.9984 (conflict-based) to 0.9905 (Relieff) at 10 features. Major advantage of the conflict-based ranking is that it requires less computation time than the ReliefF algorithm by a speed-up factor of 11.39. This makes the proposed feature selection favourable for frequently adapting systems.

Furthermore, it is shown that the proposed search algorithm terminates faster than an exhaustive search over the set of feature subsets $\mathbb{F}$. In the presented evaluation, the search algorithm is able to find the subset with optimal accuracy. Although, this is not guaranteed to be the case for every application scenario. The algorithm may find only a local optimum, which depends on the randomly selected starting feature subset and the chosen parameters.

Thus, future works need to improve the proposed search algorithm. Moreover, this contribution does not consider that information fusion systems benefit from reducing the overall amount of features. It simply searches for the subset showing best performance, although there may be a feature subset performing slightly worse, which consists of significantly less features. A promising idea is to bias the search algorithm in favour of small feature subsets. This also needs to be addressed in future work.

Acknowledgment

This work was partly funded by the German Federal Ministry of Education and Research (BMBF) within the Leading-Edge Cluster "Intelligent Technical Systems OstWestfalenLippe" (it's OWL) (Grant No. 02PQ1020).

References

[1] D. Hall and J. Llinas, "Multisensor Data Fusion," in *Handbook of Multisensor Data Fusion*, ser. Electrical Engineering & Applied Signal Processing Series, D. Hall and J. Llinas, Eds. CRC Press, 2001, vol. 3.

[2] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[3] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[4] V. Lohweg and U. Mönks, "Sensor Fusion by Two-Layer Conflict Solving," in *2nd International Workshop on Cognitive Information Processing (CIP 2010)*. IEEE, 2010, pp. 370–375.

[5] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant Features and the Subset Selection Problem," in *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 121–129.

[6] E. Alpaydın, *Introduction to Machine Learning*, 2nd ed. Cambridge: The MIT Press, 2010.

[7] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: Foundations and applications*, ser. Studies in Fuzziness and Soft Computing. Berlin Heidelberg: Springer, 2006, vol. 207.

[8] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley-Interscience, 2012.

[9] K. Kira and L. A. Rendell, "A Practical Approach to Feature Selection," in *Machine Learning Proceedings 1992*. Elsevier, 1992, pp. 249–256.

[10] K. Kira and L. A. Rendell, "The Feature Selection Problem: Traditional Methods and a New Algorithm," in *Proceedings of the Tenth National Conference on Artificial Intelligence*, ser. AAAI'92. AAAI Press, 1992, pp. 129–134.

[11] I. Kononenko, E. Simec, and M. Robnik-Sikonja, "Overcoming the myopia of inductive learning algorithms with RELIEFF," *Applied Intelligence*, vol. 7, pp. 39–55, 1997.

[12] A. H. Land and A. G. Doig, "An Automatic Method of Solving Discrete Programming Problems," *Econometrica*, vol. 28, no. 3, p. 497, 1960.

[13] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A Survey on Evolutionary Computation Approaches to Feature Selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016.

[14] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene Selection for Cancer Classification using Support Vector Machines," *Machine Learning*, vol. 46, no. 1, pp. 389–422, 2002.

[15] M. Bator, A. Dicks, U. Mönks, and V. Lohweg, "Feature Extraction and Reduction Applied to Sensorless Drive Diagnosis," in *22. Workshop Computational Intelligence*, F. Hoffmann and E. Hüllermeier, Eds. KIT Scientific Publishing, 2012, pp. 163–178.

[16] G. Shafer, *A Mathematical Theory of Evidence*. Princeton NJ: Princeton University Press, 1976.

[17] U. Mönks, *Information Fusion Under Consideration of Conflicting Input Signals*, ser. Technologies for Intelligent Automation. Berlin, Heidelberg: Springer, 2017.

[18] B. M. Ayyub and G. J. Klir, *Uncertainty Modeling and Analysis in Engineering and the Sciences*. Boca Raton, FL: Chapman & Hall/CRC, 2006.

[19] J.-F. Ehlenbröker, U. Mönks, and V. Lohweg, "Sensor Defect Detection in Multisensor Information Fusion," *Journal of Sensors and Sensor Systems*, vol. 5, no. 2, pp. 337–353, 2016.

[20] V. Lohweg, C. Diederichs, and D. Müller, "Algorithms for Hardware-Based Pattern Recognition," *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 12, pp. 1912–1920, 2004.

[21] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1, pp. 273–324, 1997.

[22] R. Luus and T. H. I. Jaakola, *AIChE Journal*, vol. 19, no. 4, pp. 760–766, 1973.

[23] T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods," *SIAM Review*, vol. 45, no. 3, pp. 385–482, 2003.

[24] G. Gopalakrishnan Nair, "On the convergence of the LJ search method," *Journal of Optimization Theory and Applications*, vol. 28, no. 3, pp. 429–434, 1979.

[25] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine," in *Ambient Assisted Living and Home Care*, ser. Lecture Notes in Computer Science, D. Hutchison et. al., Eds. Springer, 2012, vol. 7657, pp. 216–223.

Dieser Tagungsband enthält die Beiträge des 27. Workshops „Computational Intelligence“ des Fachausschusses 5.14 der VDI/VDE-Gesellschaft für Mess- und Automatisierungstechnik (GMA) und der Fachgruppe „Fuzzy-Systeme und Soft-Computing“ der Gesellschaft für Informatik (GI), der vom 23. – 24.11.2017 in Dortmund stattfindet.

Der GMA-Fachausschuss 5.14 „Computational Intelligence“ entstand 2005 aus den bisherigen Fachausschüssen „Neuronale Netze und Evolutionäre Algorithmen“ (FA 5.21) sowie „Fuzzy Control“ (FA 5.22). Der Workshop steht in der Tradition der bisherigen Fuzzy-Workshops, hat aber seinen Fokus in den letzten Jahren schrittweise erweitert.

Die Schwerpunkte sind Methoden, Anwendungen und Tools für
- Fuzzy-Systeme,
- Künstliche Neuronale Netze,
- Evolutionäre Algorithmen und
- Data-Mining-Verfahren

sowie der Methodenvergleich anhand von industriellen und Benchmark-Problemen.

Die Ergebnisse werden von Teilnehmern aus Hochschulen, Forschungseinrichtungen und der Industrie in einer offenen Atmosphäre intensiv diskutiert. Dabei ist es gute Tradition, auch neue Ansätze und Ideen bereits in einem frühen Entwicklungsstadium vorzustellen, in dem sie noch nicht vollständig ausgereift sind.

ISBN 978-3-7315-0726-0
9 783731 507260 >

Gedruckt auf FSC-zertifiziertem Papier