Data-Driven Optimization Of Hot Rolling Processes

Dissertation

zur Erlangung des Grades eines

Doktors der Ingenieurwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Christian Jung

Dortmund

2019

Tag der mündlichen Prüfung:
12. Dezember 2019

Dekan:
Prof. Dr.-Ing. Gernot A. Fink

Gutachter:
Prof. Dr. Günter Rudolph, Technische Universität Dortmund
Prof. Dr. Thomas Bartz-Beielstein, Technische Hochschule Köln

Acknowledgments

This work would not have been possible without the help and support from many friends, colleagues and family. Being a part-time researcher is always a balancing act which is only possible with continuous support from many people.
I would like to express my gratitude to my supervisor Prof. Dr. Thomas Bartz-Beielstein for his strong support over all these years. The great atmosphere at his institute and the fruitful discussions were an important factor for the successful completion of this thesis.
My supervisor at TU Dortmund, Prof. Dr. Günter Rudolph, made it possible to carry out my doctoral research with a part-time position. He provided valuable guidance and gave generous support. The ideas and suggestions improved the quality of my research tremendously.
I would like to thank Prof. Dr. Thomas Schwentick for being my mentor and guiding me through the last phase of my PhD.
A very special thanks to my colleagues at the Institut for Data Science, Engineering, and Analytics (IDE+A), especially Martin Zaefferer, Martina Friese, Andreas Fischbach and Jörg Stork for great discussions about connected research topics, further research ideas and valuable feedback.
A very special and warm thanks goes to the following persons for their advice and discussions during the doctoral seminars: Beate Breiderhoff, Oliver Flasch, Sebastian Krey, Olaf Mersmann, Steffen Moritz, Boris Naujoks, Quoc Cuong Pham, Stefanie Raupach, Margarita Rebolledo, Frederik Rehbach, Viktoria Schaale, Katya Vladislavleva.
I am grateful to Gregor Schneider for his continuous support and for making this work possible.
I cannot express how thankful I am to my family for their endless support and love they gave me through all of my live and especially during my academic education.

Contents

1. Introduction

Complex real-world problems require adequate handling and analysis. There is no best method for all kind of problem available and experts are needed to understand and adapt algorithms on a specific problem. This is also known as the No-Free Lunch theorem described by Wolpert and Macready [WM97].
In contrast to artificial problems, many real-world problems have to deal with uncertainties and inaccuracies. Hot rolling is an important example of a real-world problem, where multiple of these aspects have to be considered.
Finite element methods (FEM) are available to calculate exact solutions for certain conditions. Due to the long calculation time required for FEM methods and the necessity to dynamically adapt to certain conditions, FEM methods cannot be used for prediction. Instead of FEM models, simple analytical models have been developed. They are validated and optimized for all occurring situations. One example of such an analytical model is a material model which describes the mechanical properties [Zhi10] and the temperature behavior [KW99, YS16] of the material during the rolling process. For regular material grades the description and behavior during rolling is well known and the accuracy is very high.
guessed based on similar materials. One way to improve the description is to take expensive laboratory measurements. Another way is to use data from rolling trials and to build a data-driven meta model for the description of the material. The meta-model will be embedded in a simulation environment and will be validated against the analytical model. This can save costs and time and can easily be extended also to optimize existing material descriptions. The optimization is done offline, when enough data for a certain material is available. Offline hereby refers to the fact that there will be no restriction to time or any other limitation to the optimization. Since the description of a material is highly interconnected with other analytical process models, the question is, if this data-driven optimization procedure can significantly improve the accuracy of the material description for the whole range of products. The roll force required during rolling is one important parameter which is strongly correlated to the material and the geometries. Therefore, the prediction errors of the roll force will be used to measure the quality of the description.
Residual prediction inaccuracies will still be observable for all kind of analytical models, even after optimization. To achieve the highest possible prediction

accuracy, online algorithms are used. Online refers to a property of the algorithm to update its coefficients on arrival of new samples. Basically, the update mechanism is executed incrementally instead of calculating the coefficients from the whole dataset.
Online algorithms are especially of interest, if the distribution of a dependent variable or the target value changes. This behavior is called drift [SG86, WK96] and will cause the prediction accuracy to decrease. It cannot be avoided and has many reasons, e.g. sensor drift or mechanical clearance.
Online Support Vector Regression (SVR) is a promising algorithm which was developed at the early years of the century by Martin [Mar02] and Ma and Theiler [MTP03] as an extension to Support Vector Machines (SVM). Despite the high popularity of SVM in general, which have been applied in various fields like on text recognition [Kat17] or protein domain classification [HNK14], only a few publications (see e.g. [OMBH07, OJB11]) devoted for online SVR on real-world problems are available. The published algorithm allows the SVR algorithm to be incrementally updated instead of a complete recalculation of all parameters from the whole dataset. Also, a procedure for the removal of a sample is available but still some questions are open which have to be solved for real-world applications:

- How many samples should be stored?
- How to select samples which should be removed?
- Which kernel should be used?
- How to select constraint and insensitive loss for the online SVR?
- How to handle categorical parameters?

To answer these questions, the available online SVR algorithm based on Ma and Theiler was extended. To validate the results, the performance on publicly available datasets is shown. The impact on the kernel choice, storage size and SVR parameters are shown on data collected from various hot rolling mills.
A general problem for many online algorithms is the handling of categorical variables. In classical learning methods, specific coding schemes are used and additional variables are added to the dataset. This concept is only feasible if the number of different occurrences of the categorical variable is known prior to modeling. This will in general not be valid for online algorithms, where no information about the categorical is known in advance. Four concepts for online learning with categorical variables are addressed in this theses. Their influence on the performance for three different online learning algorithms is discussed on data generated during hot rolling.

1.1. Innovation

Parts of this thesis are based on the following publications:

- Metamodel-based optimization of hot rolling processes in the metal industry [JZBBR17].
- Extending Support Vector Regression for On-line Learning Methods on Real-world Data [JBBR18].

The main innovation was achieved on the following topics:

- Kriging application on real-world industrial data.
- First data-driven optimization procedure for determination of flow curve parameter.
- Data-driven optimization across multiple rolling mills.
- Comparison of online algorithms on real-world data from rolling mills.
- Extension of Online SVR with strategies for storage management.
- Analysis of Online SVR for different storage sizes.
- Creation of strategies to handle categorical variables in online algorithms.
- Online parameter optimization, especially for SVR.

1.2. Outline

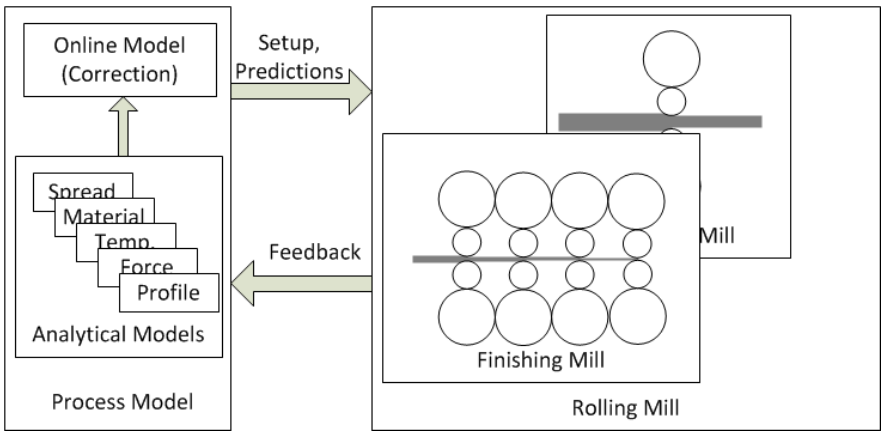


Figure 1.1.: Hot rolling optimization overview. Analytical models are used to predict and describe the rolling process. These models can be used for various types of rolling mills. The analytical models are interconnected and will be corrected with online models which are using feedback from the rolling process to minimize the prediction error.

Figure 1.1 shows an overview of the optimization procedure for hot rolling. Several analytical models are used to describe the rolling process. All those models are closely interconnected with each other. The analytical models are mostly well established physically-based models but also empirical models are used. Online models are optimizing the prediction accuracy with feedback received from the rolling mills. The main reason for this correction is the high dynamic behavior of the process and an inaccurate description for special situations.
This thesis starts in Part I with an introduction to the rolling process and a description of possible sources of errors in Section 2. The focus will be on hot rolling of steel and aluminum but most of the proposed methods, models and algorithms can directly be transferred to cold rolling. Further, the main analytical models and their influence on the prediction are introduced. After introducing the objectives in Section 3, a data driven approach for meta-model based optimization is presented in Section 4. This will be used to optimize the analytical models for various scenarios. Following the offline optimization, this thesis continues with the main contribution on online algorithms and their optimization in Part II. After an introduction to online algorithms in Section 5, the dataset used for performance evaluations are described in Section 6. A discussion and description of the online algorithms is given in Section 7. A simple first order algorithm, i.e., Passive Aggressive, will be compared to the more advanced Recursive Least Squares algorithm. Afterwards the online SVR algorithm is presented and discussed. Extension made for the online SVR are presented and their impact on the performance will be shown. The performance for all online algorithm will be compared on several datasets generated during hot rolling but also the performance to well established state-of-the-art algorithms is shown. The thesis ends with a summary on the achievements and gives some recommendations for other real-world processes.

Part I.

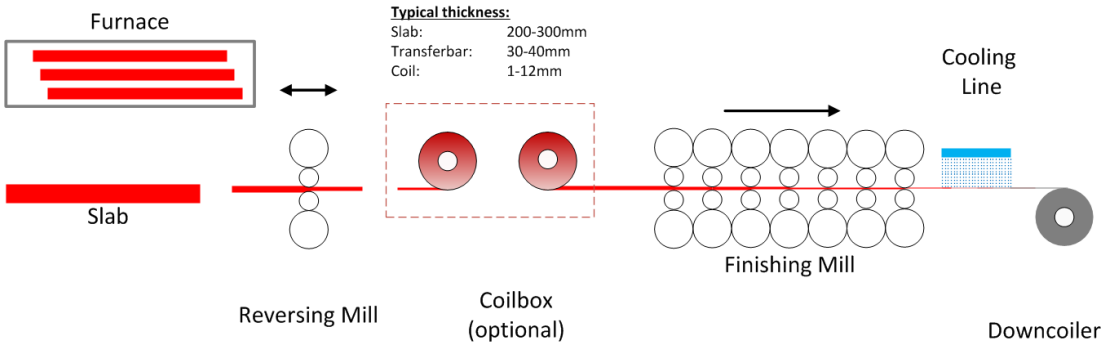
Data-Driven Surrogate Optimization of Hot Rolling Processes

2. Hot Rolling

Hot rolling refers to a process where heated material is plastically deformed between two or more steel rolls in order to produce a thinner product. A typical layout for a steel and aluminum mill is shown in Figure 2.1. Today's rolling mills require a huge variety of different materials and geometries to be processed within close tolerances. To achieve this, the prediction of the process behavior and controlling of process parameter should be as accurate as possible. Various interconnected physical-based software models are used for the process prediction. The most relevant models and their tasks are introduced in Section 2.3.
Various mill types have been established in the market for different requirements. The major difference is the target product type which can be a plate or a strip. The mills for the production of a plate are consequently called plate mills. The most common mill types for steel or aluminum production are hot strip mills where the material is coiled at then end of the production. Plate mills usually produce much thicker products than hot strip mills.
Another difference is the maximum width which can be rolled. Plate mills are able to produce much wider products than hot strip mills. This is achieved with turning tables which allow the product to be rotated by 90 degrees. Table 2.1 shows common geometries for the different aluminum and steel mills. Although aluminum and steel as well as plate mills and hot strip mills differ in detail, the main process flow and utilized components are common to all mill types. Therefore, these components are introduced in the following section.

Table 2.1.: Typically used product geometries for plate mills and hot strips mills for aluminum and steel. The ranges are defining the common values for the input and target geometries in mm.

			Steel Mills		Aluminum Mills	
			Plate	Strip	Plate	Strip
Input	Thickness	[mm]	200-300		400-600	
	Width	[mm]	600-2000		600-2000	
Target	Thickness	[mm]	5 - 50	1 - 25	4 - 100	2 - 15
	Width	[mm]	600 - 5500	600 - 2500	600 - 5500	600 - 2500



(a) Hot Strip Mill for Steel

(b) Hot Strip Mill for Aluminum

Figure 2.1.: Hot Rolling mills for steel (top) with coil box and aluminum (bottom). The concept is similar. For steel mills, additional a cooling line is installed after the last finishing mill stand. Due to higher initial thickness the total length of aluminum mills is usually much longer than for steel mills. The coilbox can be used for steel mills to further decrease the distance between reversing mill and finishing mill and to improve the temperature behavior.

2.1. Hot Strip Mill Process

The layout of a common hot strip mill for steel and aluminium is depicted in Figure 2.1. The figure is reduced to represent only the main components. The rolling process for steel and aluminum is very similar. The work flow of the process is from the left to the right side.
The process starts with the charging of the furnace. Here, slabs which are usually at room temperature are charged and reheated to temperatures around 1200°C for steel mills and 500°C for aluminum mills. The exact temperature depends on the specific material and target properties. If multiple material grades are charged within the same furnace a balancing for all different properties has to be made. In order to reduce energy costs, the heating time is tried to be kept at a minimum. The initial products are ingots, which are casted into a crystallizer shape, or traditional slabs which are produced through a continuous casting process. For aluminum hot strip mills, ingots are usually used. They are additionally milled and sawed to compensate material flow effects during rolling. Due to the wide range of aluminum materials, different shapes have been established for hard and soft alloys.
After the target temperature in the furnace has been reached, the product will be discharged from furnace and transported to the first mill stand where the first deformations take place. The mill stand is usually a so-called quarto reversing mill, because the stand consists of four rolls. Reversing refers to the change of rolling direction of two consecutive reductions. Here, the thickness of the product is reduced to an intermediate thickness for hot strip mills or to the final thickness for plate mills. The reduction is done in a multiple steps, so-called passes. For plate mills the next process step is the hot plate leveler, which is not depicted in Figure 2.1. Here, the plate is straightened before it will be transported to a cooling bed.
For hot strip mills, the rolling will continue in the finishing mill, where the product is rolled in several stands to the final thickness. The additional cooling line after the last finishing mill stand is only used for steel mills because of the higher temperatures for steel. Furthermore, the cooling strategy will have an huge impact on the strength of the product[TLWM13]. Finally, the product is coiled in the downcoiler, respectively belt wrapper.

2.2. Main Components of a Hot Strip Mill

After the hot strip mill process has been described, this sections will briefly introduce the different tasks of the main components:

- A reheating furnace
- Reversing mill
- Finishing mill

Afterwards, a short summary with the major tasks and the problems is given in Table 2.2. For a more detailed description, the specific literature is recommended. A good overview for hot rolling is given, e.g., in [GB00, Hin03, HS78, Web73].

Gas Furnace

The furnace will usually be charged with cold material and has the task to reheat the product until a certain temperature is reached. Moreover, the furnace is divided into several heating zones with the task to homogeneously heat the material. The temperature distribution within the material is of major importance especially for steel rolling. The reasons for this is the different heat conductivity. Aluminum has a much higher conductivity than usual steel grades and therefore the temperature distribution for aluminum grades is much more homogeneous than for steel grades. If the surface of the slab is much hotter than the core, the deformation will not be equally among the thickness cross-section and undesirable effects might occur. Sometimes the reason for a high temperature gradient among the thickness is the desire to save energy. To minimize gas consumption and to optimize the temperature distribution, a predictive control is used to continuously predict the temperature of the slab at distinct locations over the thickness, width and length. The reheating procedure will cause an inhomogeneous temperature distribution within the product. Temperature deviations over the thickness are critical for the rolling process and have to be limited. The length and width directions can usually be ignored for the prediction. The furnace model will predict 5-10 discrete points over the thickness and according to this prediction and some other constraints, the gas consumption is controlled.
The reheating process lacks of precise monitoring since only the surface temperature of the product and the gas temperatures within the furnace can be measured online. A common way to improve the temperature predictions is to use prepared slabs with thermocouples positioned at multiple locations within the slab [SK96]. They monitor the heating process and allow further tuning of the predictive models.

Hot Reversing Mill

A reversing mill is usually the first mill after the product has been discharged. Reversing is referring to a situation, where the rolling direction will change after

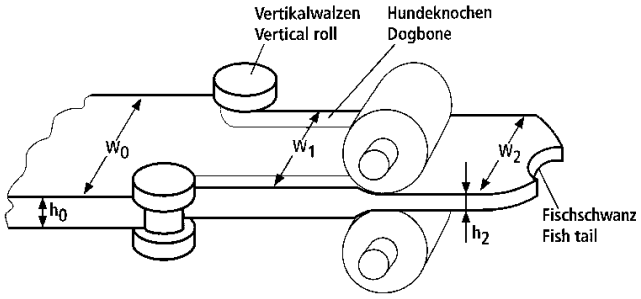


Figure 2.2.: Illustration of a width reduction followed by a thickness reduction in a roughing mill according to [WSDQS98].

each deformation. For plate mills there are usually one or two reversing mills. A conventional hot strip mill usually also consists of one or two reversing mills, called roughing mills. In contrast to reversing mills used for hot strip mills, plate mills can handle much higher forces and torques, because they are build and used for much wider products. Modern plate mills are able to produce plates of approximately 5m width while the usual production width of hot strip mills is up to 2m.
In a roughing mill, the reheated product is rolled down to a thickness of 30mm - 50mm, dependent on the final thickness of the product and capability of the proceeding mill. The reduction from the initial thickness down to this final thickness is conducted in multiple steps, so-called passes. A pass is a specific deformation of the product and may consist of a width reduction followed by a thickness reduction. The width reduction is done in a vertical mill, called edger, which is an optional part of the roughing mill. The edger is used to allow a certain variation of slab width and to guarantee a specific target width of the product. This will be achieved by controlling the position of the edger rolls dynamically during each pass. After each width reduction in the edger a thickness reduction will be made. Therefore, the width can only be influenced indirectly. The combination of a width and a thickness reduction is depicted in Figure 2.2.
In the following figures, index 0 will always denote the initial geometry, index 1, the intermediate geometry, i.e. the geometrical description between edger and rougher. Index 2 will denote the final geometry after deformation. The initial width w_0 is reduced in the edger to w_1 and the thickness will increase to h_1. The width reduction creates a dog-bone shaped product which is depicted in Figure 2.3. The dog-bone will cause a higher material spreading as a result of the thickness reduction from h_1 to h_2. The efficiency of the edger is usually between 40% and 60%. A width reduction of 100mm ($w_0 - w_1$) in the edger will cause a higher spreading during the thickness reduction. The remaining width reduction, i.e.

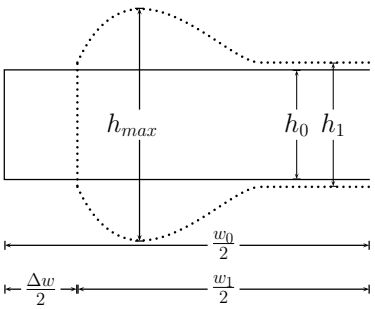


Figure 2.3.: Example of a dog-bone shape which is created after the width reduction and before the slab enters the horizontal mil. w_0 is the entry width into the edger, w_1 is the exit width out of the edger, Δw is the width reduction. h_0 defines the entry thickness, h_1 the exit thickness out of the edger. This might be higher than h_0. h_{max} is the maximum height of the dog bone area.

$w_0 - w_2$ will then only be 40mm - 60mm. This efficiency is highly correlated with geometry, materials and temperature distributions within the material. The spreading behavior of the material is not equally throughout the length of the slab. To compete this effect, the width reduction over the length is dynamically adjusted by the edger. This is done especially at the beginning and the end of the pass. During the beginning and the end of a pass the material has different flow properties causing an unequally spread among the length. A typical behavior is the creation of a so-called fish tail which is depicted in Figure 2.2. A colder core part of the product will increase the material flow to the edges of the product. This will result in a higher spreading of the material. Unfortunately, the core temperature is only a predicted value and cannot be measured directly on-line.
The thickness reductions in reversing mills are optimized in terms of productivity and may consists of five to seven passes for steel roughing mills or up to 30-40 passes for plate mills or aluminum roughing mills. Here the prediction of the roll force is essential to achieve a proper quality of the product. The following measurements may be retrieved from a reversing mill:

- Rolling force.
- Drive torque.
- Roll gap - distance between the rolls.
- Rotational speed of work rolls.
- Bending force (for profile and flatness control).

- Axial shifting (for profile and flatness control).

Usually, additional measurements are installed in front or behind the mill. This may include width, profile, thickness and speed measurements. For plate mills usually also the profile will be measured.

Finishing Mill

Standard finishing mills consists of multiple stands, i.e., six or seven for steel and three to five for aluminum. In contrast to reversing mills, the rolling direction is continuous and therefore this part of the hot strip mill is sometimes also called hot continuous mill. The finishing mill will constantly reduce the thickness of the product in each stand and will finally wind the hot strip. The final product is then called a coil.

For steel mills, an additional cooling line is installed between the last mill stand and the coiling process (see Figure 2.1). The cooling line will apply a decent amount of water on the product to reach the target coiling temperatures. The additional cooling is of major importance for the production of advanced steel grades.

In aluminum rolling, no cooling line is used. During rolling, all stands are active at the same time. Therefore the speed of all finishing stands should be exactly controlled during rolling. Otherwise the mass flow is not constant and more material leaves one stand than enters the subsequent stand. As a consequence, the material would accumulate in front of one stand resulting into a stop of the rolling process. To stabilize the rolling process, a certain amount of tension between the stands is used and controlled. For the start of rolling in the finishing mill, i.e. before any thickness controller is used, roll force prediction errors will cause a thickness deviation at the beginning of the coil. Later on, the thickness is controlled by measurement feedback. In contrast to the hot reversing mills, the prediction for a certain material is only updated on a product-to-product basis instead of a pass-to-pass basis. Fewer corrections during rolling are required, if the predictions are improved. Therefore, good predictions will cause a stable rolling process. The measurements which are taken for finishing mills are basically the same measurements which are also taken for reversing mills. Additionally, the interstand tension will be used. Table 2.2 summarizes this section.

Table 2.2.: Tasks of the main components and possible problems.

Component	Task	Problem
Furnace	Homogeneous heating of slabs / ingots to a specified temperature	Lack of measurements and validation. High impact on process.
Hot Reversing Mill	Rolling in multiple steps to target thickness and target width	Temperature distribution not measurable. No direct width control.
Finishing Mill	Rolling from intermediate thickness to final thickness.	Very accurate speed and force prediction necessary.

2.3. Process Models

Various models are used in hot and cold rolling which have the task of predicting important process parameters. The term model hereby refers to computer programs which are describing the physical process as exactly as possible. These programs are normally developed by implementing the physics behind the process. Some assumptions made in the software are simplifications, the result of some finite element (FE) studies or some data driven approaches. All of those simplifications are made to accelerate the computations. Because this will be a general description the term product will address plate and coil production throughout this section. Figure 2.4 shows the interaction of process models for the description of a single pass. Various analytical process models are participating in the state description S_{t_i} for time instance t_i. Furthermore, they depend on each other and share information. At time instance t_0 the product starts accelerating towards the mill. During this time, i.e., from t_0 to t_1, the product will cool down. This will be taken into account by the temperature model M_T, which will calculate the expected entry temperature distribution $\vartheta(t_1)$ at t_1. This temperature will be used by the force model M_F to calculate an expected roll force which will then again be used by other models to update the predictions. The principle will be the same for all state descriptions until the pass has finished and the product reached its end position at t_2.

The accuracy of the physical models which are describing the rolling process are of major importance. According to their predictions the mill will be operated. Some guidelines may be set from process experts but the calculations and predictions are executed with those process models. One of the major aspects of the rolling process for a single product are e.g. the number of passes in a hot reversing mill. The value can be set from outside but it is not guaranteed that it will be possible

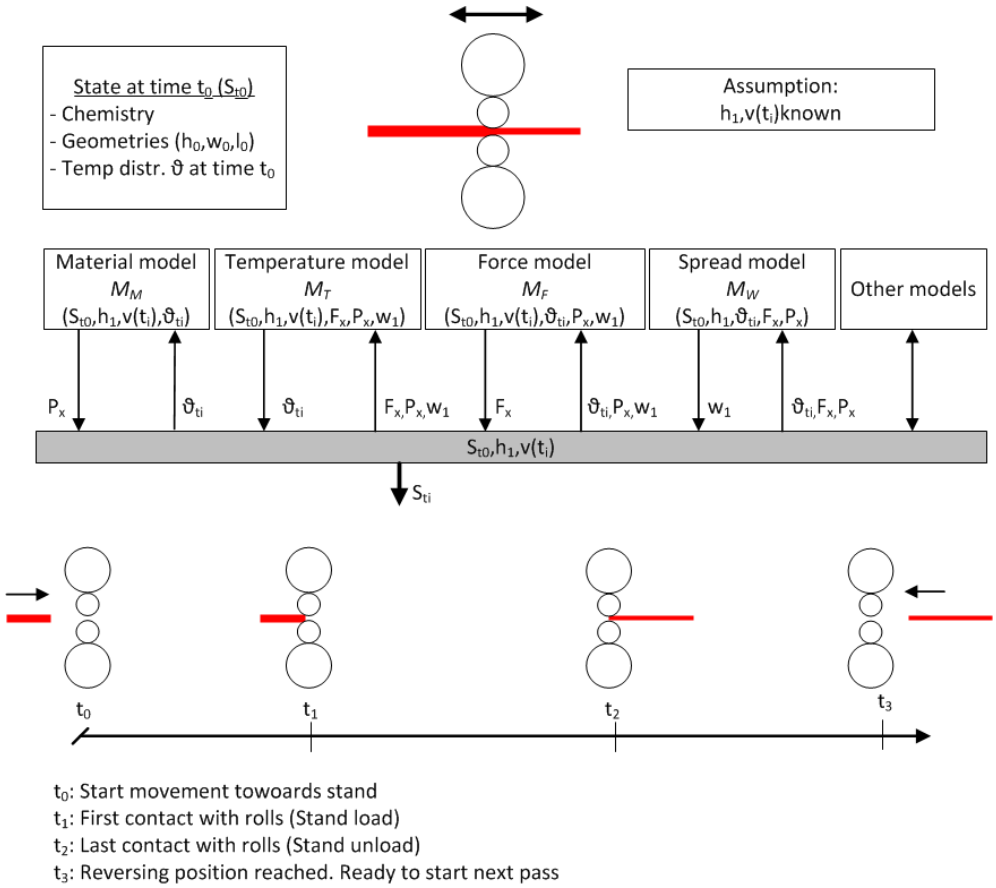


Figure 2.4.: Interactions of various process models for the product description of a single pass. It is assumed that all relevant information at time instance t_0 are known and summarized in the state description S_{t_0}. For simplicity, the target thickness h_1, the transport and rolling speed $v(t_i)$ are also assumed to be known. Four relevant times are denoted in the time line. t_1 denotes the time where the product will have the initial contact with the rolls and the deformation starts. From t_0 to t_1 the most relevant model is the temperature and material model. They will calculate the temperature loss of the product and will calculate a new temperature distribution ϑ_{t_i} within the product. Because the temperature will change also temperature dependent data P_x from the material model M_M are changed. The force model and spread model will also update their calculations F_x and w_1 since both are dependent on the temperature model and the material model. From t_1 to t_2 the deformation takes place and almost all models participate in the product description. After the pass has finished at time instance t_2 the further temperature loss has to be calculated until the product reaches the reversing point at time instance t_3.

to reach a certain target thickness with the specified number of passes. There are limitations like maximum reduction within each pass, i.e. geometric constraints, and also mill capacities which have to be considered. These are e.g. maximum roll force and maximum drive torque. Those limits are considered within the physical models to calculate the number of passes which are required to reach the target thickness. As a result an instruction is given how many passes and how much reduction each pass should have. Additionally, temperatures, width, speed and all other necessary parameter are calculated. The total instructions for each pass are combined in the pass schedule for each product.
For plate rolling the width can be achieved with the help of turning tables. They are turning the product so the passes can be rolled in cross-direction. After some so-called spreading passes the product is turned again but is now much broader than before. The calculation of the lengthening and broadening effect has to be very precise in order to achieve also a specific target width. Limits, which are considered here are e.g. the maximum length of the product where turning is still possible. If the real length is larger than the calculation it will be impossible to turn the product again and thus the final geometries cannot be reached anymore.
The following section describes the main physical models and their tasks. For models which are considered in some of the experiments, a more detailed description is available in the corresponding section.

Material Model M_M

The description of the material properties is one of the most critical aspects for hot rolling. The material will determine the required roll force which is used for deformation, the temperature behavior and many more aspects relevant in terms of quality. According to the Worldsteel Association[1] more than 3.500 different steel grades are currently known. Each material has an unique physical and chemical property which depends on the chemical composition. Some of them are very similar and might not show significant differences during rolling. These materials can then be treated together as one material group. Nevertheless, different groups behave completely different and it is essential to predict their behavior as accurate as possible.
The main material characteristic which are used for the hot rolling process are:

- Deformation resistance,
- heat capacity,
- conductivity and
- linear expansion.

The deformation resistance is the most relevant characteristic for the prediction of

[1] https://www.worldsteel.org

the roll force. The heat capacity and conductivity are influencing the temperature behavior of the material. The linear expansion determines the relationship between hot and cold geometries. This is an important property since the material is rolled at high temperatures and they are sold when they are cold.
Those properties are calculated online for the actual process state since most of those properties are described by polynomial functions which are dependent on the actual temperature.
Unfortunately, there is no general description of those properties in dependence of the chemical content available. Thus, different approximations for each material group will be used. Beside the parameter for each material, also the specific approximation formula might be different for each material. As a consequence, the predictive accuracy is also correlated to the material group.

Temperature Model M_T
The temperature model uses information from the material model to calculate the temperature behavior for each product at defined process states and positions within the product. Each cooling device and its influence on the material can be parameterized. Beside the active cooling devices like descaler, cooling lines or transferbar cooler, the temperature losses of the product through radiation and temperature induction caused by the rolling are also taken into account. Although the temperature distribution over the whole material thickness, i.e. surface to core, will be calculated, the validation can only be made by comparison with measured surface temperatures. The measuring will be done by pyrometers which are measuring the intensity of certain wavelength occurrences.
For hot rolling of steel the occurrence of iron oxide impurities might cause abnormal temperature measurements. Also the environment might be full of steam and water which causes measurement inaccuracies. Therefore, temperature measurements should only be used with caution. Pyrometers are also used for the hot rolling of aluminum but here the pyrometers are regularly synchronized by a contact temperature measurement.

Spread Model M_W
Spread is defined as material flow in width direction. This is usually a challenging task for hot strip mills with an edger installed. The material flow is dependent on temperature, material and geometries and these are also the dependencies for additional spread caused by the dog bone shape. There are several descriptions for the dog bone shape available but the validation of those are difficult, since no shape measurements can be installed. The dog-bone shape will be highly correlated with the additional spreading which occur.

Additionally, the usual hot strip mill will have only an exit side measurement gauge installed but material spread will also occur in each backwards pass. The optimization is then called a hidden state optimization, since several states in between cannot be measured and validated.
Aluminum hot strips will usually be trimmed between the last stand of the finishing mill and the belt wrapper. Therefore the prediction does not have to be as accurate as in the case of the steel mill. However, the milled ingots usually have a defined shape to prevent an excessive side alligator effect. Side alligator is referring to an effect, where additional material flow to the edges is created at the top and the bottom of a product. This will cause the edge shape to look like an alligator mouth. The main target for aluminum is a uniform width over the total length. This includes a rectangular shape of the beginning and the end of the product. The edgers are used to maintain such a shape but it can only work if the predicted width is within some close range.

Force Model M_F
Accurate prediction of the force is important to have a stable process. The force is a major contributor to the strategic decision how many passes will be used. Good predictions also prevent damages to the mill because they ensure that no reductions are made which would require roll forces higher than the mill capability. The force is calculated based on the deformation resistance of the material which is unique for each material. The so-called flow curve are dependent on the deformation itself, the deformation rate and the temperature. A more detailed description of the roll force is given in Section 4.3.1.

3. Objectives and Performance Figures

The data-driven optimization can be used to optimize all parameters which are used for predictions in the rolling process. Typical objectives are parameters for the prediction of the roll force and roll torque but also geometrical predictions for width, thickness or profile can be optimized. The following measurements are usually taken during the rolling process and can be optimized:

1. Roll force
2. Roll torque
3. Thickness
4. Width
5. Profile
6. Surface temperature
7. Flatness

The algorithm and concepts described in Section 4 are used to optimize parameters which are influencing the prediction of those values. The prediction of the roll force, e.g., will be made with an analytical model M_F, which was introduced in Section 2.3 and will be explained in more detail in Section 4.3.1. It depends on material parameters which can be optimized. Therefore, the optimization may be conducted by comparing the prediction of the analytical model and the corresponding measurement. If the parameters can be guessed or if initial parameters are available, the optimization will be easier in most cases. Therefore, for data-driven optimization, the following tasks for the optimization can be summarized:

- General data-driven optimization for unknown initial parameter (Section 4.4).
- Optimization for known initial parameter (Section 4.5).
- Optimization across multiple mill types (Section 4.6).

The online optimization described in Section 5 will usually optimize the residual prediction error, i.e., the deviation between an analytical model and the measurement of the corresponding variable. It can therefore be seen as the logical next

step in optimization of this variable.
To measure the quality of an optimization it is important to define some metric which describes how good an approximation is. Therefore, a metric is defined which penalizes prediction errors. A common objective used is the root mean squared error (RMSE). For a given data set $s = \{\{\mathbf{x_1}, y_1\}, \{\mathbf{x_2}, y_2\} \cdots \{\mathbf{x_N}, y_N\}\}$ and prediction values $f(\mathbf{x}) = \{\hat{y_1} \cdots \hat{y_N}\}$ with N being the total number of observations, the RMSE is defined as root of the average squared error.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y} - y)^2} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}e_i^2} \tag{3.1}$$

Here, e_i are the residuals defined as the difference between the measurement and the prediction. Another quite common measure is the mean absolute error (MAE). There is no common rule which one is the better so usually the selection is done in dependence of the dataset and what should be expressed. In [TC14] the authors are discussing about the pros and cons of the RMSE against MAE and in which scenarios it might be beneficial to use MAE instead of RMSE. One good argument to use the MAE is its interpretability. Process experts can directly see from the analysis how well an algorithm is performing.
The MAE is defined as

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|e_i| \tag{3.2}$$

and the MSE is defined as

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}e_i^2 \tag{3.3}$$

where N again is the total number of samples, y the true target value and $\hat{y}$ defines the predicted value. The RMSE will be the main performance indicator in Section 4 and the MAE will be usually used for the datasets specific to rolling in Section 5.

4. Data Driven Meta-Model Based Optimization

Several physically based software models are used to describe the complex rolling process. The models are interconnected with each other and the optimization of a single aspect would not be sufficient to optimize the whole rolling process. Some of these software models are describing the different aggregates of the rolling mill. Besides the introduced process models in Section 2.3 also other models will influence the process prediction. An example is the furnace model which is modeling the temperature heat up over time and tries to predict the temperature distribution in the products. It will influence the initial state description S_{t_0} (see Figure 2.4). Other models are more focusing on the material behavior during rolling and are trying to describe phenomena which occur during rolling. This might be the spread behavior, roll force or any other kind of properties related to the finished product. All those different models are combined to achieve the best process description. The reason why they have to be combined can be seen in the prediction of the roll force. The roll force will induce temperature during rolling and therefore temperature and roll force are not independent from each other. Since many material related parameter are also temperature dependent, the changed temperature will also cause a change in this behavior.
Therefore the best approach is to model the whole rolling process for every single product. Additional information about previous rolling also affects the current behavior and also have to be considered. Those information are, e.g., thermal state of the rolls and wear of the rolls.
Consequently, the only possible solution is to model the whole scenario but focus only on the outcome on some parts. An example would be to focus on the roll force model but to have defined performance criteria for the temperature. If the temperature criteria would be violated, a penalty would be added to the roll force performance.
There are many FEM studies describing single deformation behavior of certain materials, e.g. [RPS17], or FEM studies which are analyzing the general flow behavior but unfortunately this would require too much calculation time. Furthermore, most of these studies are only valid in very special cases and clearly defined working points.

Simplifications have to be made and most often analytical models are used to describe the whole process online.
This chapter is focusing on concepts to optimize these analytical models with the use of so-called surrogate models. It starts with the definition of surrogate models, the description of the created simulation environment and the used optimization algorithms. Examples from real-world are shown in Sections 4.4, 4.5 and 4.6. Some parts of this chapter were taken from [JZBBR17].

4.1. Surrogate Modeling

A surrogate model (or meta-model) is a substitute of a usually more complex model [Søn03]. Complexity refers to the computational budget, e.g., the number of function evaluations. Simulation runs on the surrogate-model are expected to be less time consuming than simulation runs on the original complex model. Therefore, surrogate-model based optimization is used when parameter optimizations have to be performed but the run time or the total number of function evaluations on the original model are too high. The original model might be any kind of model, e.g. a finite element model, and is assumed to approximate the real behavior of the system. For some applications, the real behavior of the system might also be obtained by conduction a physical experiment. The surrogate model is designed to approximate the complex model as good as possible. Well known surrogate models are artificial neural networks, linear models, Kriging, random forest. A detailed overview of surrogate model based numerical optimization is presented by Jin [Jin03] and Jones [Jon01].
During parameter optimization with surrogate-based models usually also some fitness evaluations with the original complex model have to be made. The integration of both kinds of models in the optimization process is known as model management. A simple proposal how to combine the usage of both model types can be found in the paper by And [DT97] where the authors are defining pattern search algorithm incorporating two different kinds of models. When combining two different kinds of models a trade-of between computation time and accuracy usually has to be made.
For the problem of hot rolling, a FEM simulation of the whole process is not feasible since the simulation of a single product would already take too long, i.e. some hours. Furthermore, a FEM study of a single product would not meet our requirement since uncertainties originating from chemistry or the heating process cannot be considered and conclusions from this product to the general material behavior are not legible. The optimization would only be valid for some discrete working areas for temperature, deformation and deformation rate. FEM simula-

tions can be used to optimize special parts with limited ability of generalization of the process. An example of such analysis is shown in [RPS17] where the authors are using the FEM study to describe different behavior of stress in dependence of the rolling direction for a AISI 303 stainless steel. In [RZD12] and [LMZZ15] the authors are analyzing the shape of the dog-bone during vertical rolling for specific cases.
Since FEM simulations are not applicable the process will be described using analytical models. For the process description several analytical models are combined as described in Section 2.3. The idea is now to use a data-driven surrogate model for the process description. Data-driven models benefit from measurements which were taken from the real process. This data is used to build a data-driven model from various analytical models and retrieve the fitness according to the real measurements. For this reason throughout this chapter the complex or expensive model is denoted to be the combination of these analytical models. The principle is illustrated in Figure 4.1. A product description is used by the analytical models (1) to update the description and the settings for the process (2). All measurements collected by the process are then used to update the product description for further process steps. All those information are also used to build and optimize data-driven surrogate model. The requirement to have a variation of chemistry and pass schedules leads to the necessity to build a simulation of the rolling process incorporating multiple products. In order to optimize the analytical models, the products are ideally rolled with as much variation as possible. This way, the analytical models are optimized for a wide range of different process situations. Before the concrete analytical model, i.e. the roll force model M_F, which is object of the optimization scenario, is presented, an introduction to the simulation environment is given.

4.2. Simulation Environment

The simulation environment is driven by the fact to optimize any kind of prediction which is used for the rolling process. In fact, every prediction with corresponding measurement can be an objective of the optimization. This includes force, torque, temperature and also geometric predictions like width, thickness or profile. Also parameters like friction, which is influencing the force prediction, can be optimized. The main objective throughout this chapter will be the roll force, i.e., reducing the roll force prediction error. For some easier models, which are not strongly interconnected, simplified optimization procedures may be chosen. Pacing information like reversing times, which specify the regular delay between passes caused by the basic automation system and some simple geometry information is one

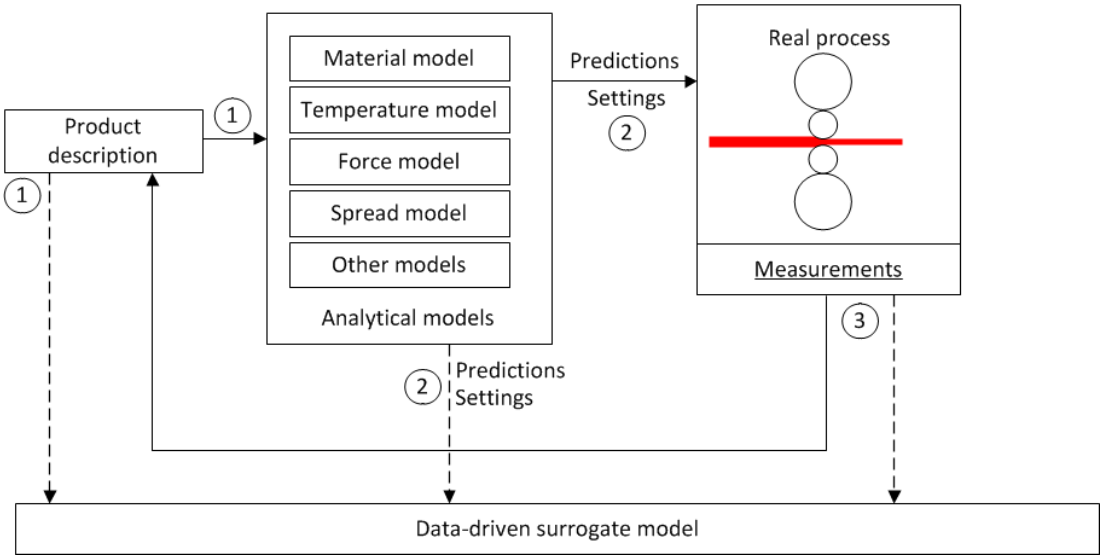


Figure 4.1.: Data-driven surrogate modeling for the description of the rolling process. The process is described by an initial state and product description (1) which is used by various analytical models (2). They will update their predictions and settings for the real process. These settings and predictions will also be available to the surrogate model. Cyclically, or after some deformation has finished, the measurements from the real process (3) will be used to update the product description and to calculate new settings and predictions for the following process. The data-driven surrogate model is used to simplify this description. The solid lines describe the online process and the dashed lines are used to describe the data-driven approach used for the offline environment.

example of these much easier tasks.

4.2.1. From Real-World Process to Simulation

The data which are used for optimization are collected from a real-world process. The analytical model which describes this process was specifically adapted for this rolling mill and is parameterized as accurate as possible. The same parameters which describe the specific rolling mill are used within the simulation scenario. This means, drive configuration, roller table layout, the position and number of all furnaces are configured and adapted within the simulation environment to fit the real mill. This is the reason why there is a different version of the analytical model for each mill available. It may also contain some plant specific configurations and include a list for the materials which are typically rolled on this mill. The data from the real-world process are collected and stored into a database to be able to use them for the simulation. Those data contain:

- Product description: For every product which is rolled, some initial geometry information from product planning are received. Usually those data are coming from the casting process but they may also be updated, e.g. when a manual measurement was taken. The product planning also defines some target geometries (width, thickness), target temperatures and material properties. Along with the geometry information is also a chemical description of the material. Usually the most important 20 or 30 elements are given to characterize the product material.
- Operator interactions: Before or during rolling, the operator may influence the rolling process based on his observations or based on some change within the planning. Some of these changes have to occur prior to rolling and some of them may occur at any time. Possible changes may restrict reduction, speed and cooling and can have huge impact to the process.
- Measurements: There are many measurements installed in rolling mills. The most important ones are geometry measurements (width, thickness), temperature measurements and measurements of roll force, roll torque and speed. The reliability and quality of these measurements are not always suitable to actively use them and therefore a separate handling for those measurements is used for the process. Only the pre-processed measurements are stored.
- Time information: Time information are important because they define the

temperature loss for each product. Especially if there is some delay during production these time information have to be used to update the further process steps.

- State descriptions: The roll state is an important parameter especially for calculation of the flatness and the profile of a product. The roll state has to be considered during calculation. Because the roll state is highly dynamic it will be stored at specific process steps, e.g. before the first pass is rolled.

Especially the state descriptions are important, because the accurate prediction of the process requires a calculation of all previous rolled products since roll change. At roll change a stable description of the rolls is possible because they are mounted at room temperature and no temperature gradient is existing. Because the roll state is stored for every product before rolling it is possible to simulate specific products out of a much larger rolling campaign. An optimization of specific material groups is therefore possible and legible.
The exact same version of the rolling model is used which is also used for the real-world process. The only difference is, that the data is not coming from different sources like product planning department, operator, gauges. Instead, they are all retrieved from the database. Algorithm 1 shows the pseudo-code of the pass schedule calculations for the real-world process which was illustrated in Figure 2.1 and described in Section 2.1. Algorithm 2 shows the adaptions made for the simulation model.
Additionally to the data above also the so-called pass schedule for each individual calculation is stored. The pass schedule contains all relevant information for the production for every single deformation step, called pass. The description of one pass contains all predictions and settings for the basic automation system. The main values are roll gap, speeds, force, torque and geometric information. They are used for the process and may or may not be changed by the operator. Usually the process will follow these changes. The simulation model has to use the same settings like the real-world process. This means the simulation model will retrieve the measured reductions and calculate the passes analogues to the settings of the real process.
This ensures that the calculations from the real-world process can be compared to the calculations of the simulation model.
If continuous mills such as hot strip mills are used, feedback after every single pass and update step in line 7 of Algorithm 1 is not necessary for the real-world process. Similar, for the simulation model step in line 7 up to line 9 will not be required. Instead, measurements after the whole rolling process will be send to the model and the update will occur for the next rolled product. Although the handling is easier, it complicates the online process, because only a certain deviation can be compensated by the basic automation system.

Algorithm 1: Rolling mill process algorithm.

```
collect initial information from product planning and calculate pass schedule;
show setup information to operator and update pass schedule with each change
  or measurement;
let N be the number of total passes in a reversing mill;
for n = 1 to N do
    send setup for pass n to basic automation system and start rolling;
    collect measurements from process and send to online model;
    use measurements to update settings and predictions for pass n;
    if n + 1 < N then
        update predictions and setup for pass n + 1
```

When the simulation scenario is used, it is important to put certain constraints on the parameters. These constraints might be just boxed constraints but can also be highly nonlinear constraints, dependent on the application. If those constraints are violated the simulation will fail or terminate unexpectedly because the usual working areas are violated.
Before any optimization can start the proper selection of products which should be analyzed have to be made. This is basically nothing else than an extraction of given data out of the database which was filled during rolling. After each run of the simulation the prediction of the interesting value is compared with the real measurement and a performance figure is calculated. Here, the RMSE defined in Section 3 is used. The number of measurements N used for the calculation can be equal to the number of products which have been rolled, but usually is much higher. This depends on the parameter which should be optimized. The roll force is calculated on multiple points along the length of a slab. Usually there are at least three discrete points over the length with corresponding measurements and predictions available. In an aluminum roughing mill with 21 passes there would be 63 prediction errors available which can be used for optimization. When the force optimization is conducted in a finishing mill with four stands, only 12 values are then available and used for optimization. Optimization for other parameters, where no difference between the three length coordinates is expected, will result in a lower number of measurements for each product.

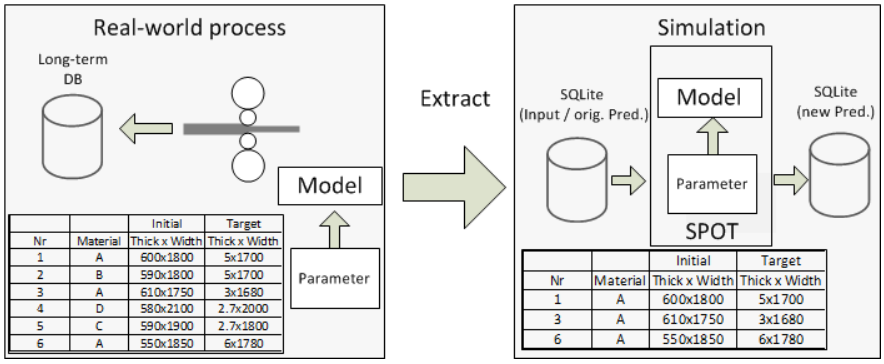
4.2.2. Implementation

The implementation of the simulation environment is done in C++ and R [R C17]. The database used to store calculations and to retrieve the input data is SQLite.

Algorithm 2: Modified algorithm used for the simulation model.

```
initialize model and read optimization parameters from file or database;
foreach product to be simulated do
    use all information available to calculate complete pass schedule;
    use time information to take into acount delay between discharge and start
     of first pass;
    let N be the number of total passes in a reversing mill;
    for n = 1 to N do
        retrieve all information for pass n from database and recalculate pass n
         'as rolled';
        store calculations in seperate database;
        use time information to take into acount delay between pass n and
         n + 1;
compute performance parameter on specified parameter, e.g. RMSE of roll
 force;
```

SQLite was chosen, because it will not require a server architecture and can easily be transferred to any other machine. The extracted data can then also be imported to other databases like Oracle. In fact, in the online process both databases are used. SQLite is used more for daily analysis and the Oracle database is used for long-term analysis.
For the purpose of flow curve parameter optimization on a specific material, the Oracle database is used and recent rolling results for this material are extracted and imported to SQLite. The principle is illustrated in Figure 4.2.
Within each optimization run a batch script is called from R which triggers the simulation. The simulation will cause the process to calculate new prediction results. These results with their corresponding measurements from the real-world process are stored in a separate database. The evaluation of the performance figure is done after the simulation finishes. Therefore, the RMSE is passed to the optimization algorithm which is then modifying one or more parameters. These parameters are used within in the next simulation run. The optimization of parameters can be done in various optimization frameworks. Here, two different optimization strategies are used which are motivated by different use-cases.
If the initial parameters are available and the parameter range is known but the dependencies of the internal models is unclear first a parameter screening with the sequential parameter framework SPO [BBLP05] is made. The framework will be introduced in Section 4.4.1.
If initial parameters are available and parameter ranges are known then the initial screening is skipped and instead a local optimization is used. The local optimization will search for the best parameter in a region of the initial parameters with



		Initial	Target
Nr	Material	Thick x Width	Thick x Width
1	A	600x1800	5x1700
2	B	590x1800	5x1700
3	A	610x1750	3x1680
4	D	580x2100	2.7x2000
5	C	590x1900	2.7x1800
6	A	550x1850	6x1780

		Initial	Target
Nr	Material	Thick x Width	Thick x Width
1	A	600x1800	5x1700
3	A	610x1750	3x1680
6	A	550x1850	6x1780

Figure 4.2.: Extraction from data in order to use them in the simulation framework. All data necessary for the calculation are stored into a long term archive database. The data is filled by the real process. Only the material which should be optimized is extracted from this database and transferred to a local SQLite database which is used by the simulation.

given boxed constraints. However, if the best parameter still are not achieving a target performance then the SPO still can be used to search for better solutions. Both optimization scenarios are described in Section 4.4 and Section 4.5 . The first one will contain flow curve parameter optimization for a material which was not known at all before and therefore the initial guess of the material behavior achieved a poor performance.
The second optimization scenario uses some flow curve parameter already adapted to a certain amount. Still, the performance is not sufficient and has to be improved. For the second optimization scenario only the local optimizer will be used. The first algorithm contains the full parameter optimization scenario with initial screening of the parameter range. Both scenarios have the analytical model in common. Therefore the used analytical model with known relationships and used parameters are introduced in the next section. Afterwards both optimization scenarios are discussed in detail.

4.3. Analytical Roll Force Model M_F

In order to understand the optimization problem it is necessary to have some knowledge about the assumptions and dependencies of the analytical models. After a first short introduction about the currently used analytical model, the optimization problem will be specified and different optimization strategies are discussed and compared.

4.3.1. Flow Curve

The prediction of roll force is a central aspect in the pass schedule calculation. As mentioned in Section 2.3 and 4.2.1, the pass schedule contains specifications and settings for each process step which occurs during rolling. It therefore contains settings of thickness reduction and width reductions for each individual pass and the corresponding predictions based on these reductions, i.e. roll force and torque. Besides, also the speed is contained in the settings and is usually varied from start to end of each pass.
To predict the required roll force for a given reduction the so-called flow curves are used. They describe the resistance of a the material during plastic deformation. Intuitively, this resistance depends on

- reduction,
- temperature,
- speed and
- material properties.

The flow curve is exactly using these parameters for the description of the resistance. The dependency of the reduction is usually expressed as effective, *logarithmic deformation* φ and is expressed as:

$$\varphi = \ln \frac{h_0}{h_2},$$

where h_0 is the input thickness of a pass and h_2 is the target or output thickness of the pass as illustrated in Figure 2.2. The logarithmic deformation is used because its dependency to the roll force can more easily expressed in this way.
The temperature is usually denoted as ϑ and the speed is included as *deformation rate* $\dot{\varphi} = \frac{d\varphi}{dt}$ which is the first order derivation of the deformation against time. The material dependency is unfortunately not expressed as a continuous formulation among the chemical compositions. Each material has its own flow curve and corresponding parameters. A material is here defined as a chemical composition close to a somehow specified target composition for an alloy. Usually, the ranges for these chemicals are very small so that no huge difference within one material is expected. This is at least valid for the usual steel grades. Unfortunately, there are different standards for steel grades available and every company may has an own definition. For aluminum grades almost every company uses the ANSI or AA standard[1] for the definition of the grades. The problem within these grades is that unique grades sometimes allow high variation of important chemical elements.

[1] https://www.aluminum.org/standards

For alloy AA5182 wide ranges of magnesium are allowed having a huge impact on the roll force [SS16].
The flow curves are not directly expressing the roll force but they describe the deformation resistance k_f during plastic deformation:

$$k_f = AK_\varphi K_\vartheta K_{\dot\varphi}. \tag{4.1}$$

Here, $A \in \mathbb{R}_+$ is a constant factor, and $K_{(\cdot)}$ are functions of the corresponding variables φ, $\dot\varphi$, and ϑ, respectively.
The first flow curve formulas were developed in the early 50s of the last century by Geleji and Ekelund and the dependency K_ϑ was only linear [HS78]. These formulas were mainly developed for low carbon steel. Afterwards also formula with higher order terms were developed for medium and high alloyed carbon steels [H$^+$72, HZEK72]. A good overview and description of the different flow curves can be found in [HS78, Hin03, TNR81].
Today most common flow curve models were originally developed by Hensel and Spittel [HS78]. They were extended at the University of Freiberg to have a better accuracy of the flow stress for high deformation grades. Thus, these extensions are called *Freiberger Approach*. Some of the available flow curve models, which are typically used in process models for hot rolling, are presented in Table 4.1. Their corresponding equations read as follows.

$$k_f = k_{f,0}A_0A_1e^{m_1\vartheta}A_2\varphi^{m_2}e^{\frac{m_4}{\varphi}}A_3\dot\varphi^{m_3} \tag{4.2}$$

$$k_f = A_0e^{m_1\vartheta}\varphi^{m_2}e^{\frac{m_4}{\varphi}}\dot\varphi^{m_3} \tag{4.3}$$

$$k_f = A_0e^{m_1\vartheta}\varphi^{m_2}e^{\frac{m_4}{\varphi}}(1+\varphi)^{m_5\vartheta}\,e^{m_7\varphi}\dot\varphi^{m_8\vartheta} \tag{4.4}$$

$$k_f = k_{f,0}A_1e^{m_1\vartheta}A_2\varphi^{m_2}A_3\dot\varphi^{m_3} \tag{4.5}$$

The multipliers A_i $(i = 0, 1, 2, 3)$ can be reduced to one parameter, A. The parameters m_j $(j = 1, 2, \ldots, 8)$ are defining the exponential behavior of the materials in dependence of the temperature ϑ, the deformation φ, and the deformation rate $\dot\varphi$. The parameters ϑ, φ, and $\dot\varphi$ are defining the working point in each deformation. The value $k_{f,0}$ used in the equations (4.2) and (4.5) is the *basic deformation* and is calculated by empirical formulations based on the chemistry.
Each material is classified according to its chemistry and gets its own parameter set $\mathcal{M}$ with parameter values m_1 to m_8 and A_0 to A_3, respectively. For the classification the chemistry is compared to a default target chemistry. This means for each flow curve a default chemistry is stored. Then, the distance of the actual chemistry to this target chemistry is calculated. In order to prevent a wrong classification, a weighting of important elements for each material is also made. This ensures that the important, characterizing elements of the material, are more significant than other elements. The procedure is depicted in Figure 4.3.

Table 4.1.: Overview of typical equations for the regression of the material flow stress. The multipliers A_i are reduced to one parameter, A. Parameters m_j are defining the exponential behavior of the materials in dependence of the temperature ϑ, the deformation φ or the deformation rate $\dot{\varphi}$ which defines the working point. Parameter $k_{f,0}$ is defined by a simple equation based on the chemistry. Entries in the column "Equation" refer to the equations defined on p. 37.

Eq.	Name	#Params	Parameter List
(4.2)	Freiberg 1	5	A, m_1, m_2, m_3, m_4
(4.3)	Freiberg 4	5	A, m_1, m_2, m_3, m_4
(4.4)	Freiberg 8	7	$A, m_1, m_2, m_4, m_5, m_7, m_8$
(4.5)	Hensel Spittel	4	A, m_1, m_2, m_3

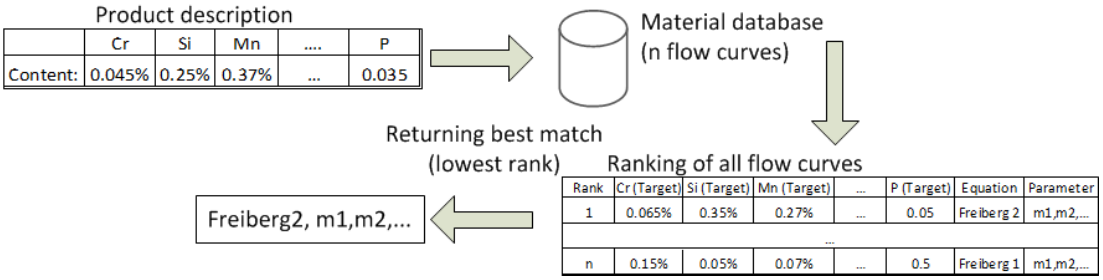


Figure 4.3.: Process for selecting a suitable flow curve. The chemistry of the product which is going to be rolled is send to the material database. Then, a ranking according to the distance to a target chemistry of available flow curves is made. The most suitable flow curve parameter together with the information about the used equation is then send back to the process model.

For each flow curve also the valid ranges of operation for the parameter-set $\mathcal{M}$ is stored. This is done to have a more suitable description for phase transformations within the material and to distinguish between hot and cold rolling. For aluminum hot rolling usually the temperatures are not lower than 250 degree Celsius. For the region below other parameter sets are used. Summarizing, the parameters $k_f, \varphi, \dot{\varphi}$, and ϑ, the multiplicative variables A_i, m_j, $(i = 0, 1, 2, 3; j = 1, \ldots, 8)$, and related functions $k_{f,0}, K_\varphi, K_{\dot{\varphi}}, K_\vartheta$, are used.
Nowadays, hundreds of different materials are known. The parameter k_f is almost linearly correlated with the roll force and roll torque. Thus, the model prediction quality and herewith the process stability and product quality are correlated to the parameter-set $\mathcal{M}$. It is therefore important to optimize those parameters in order to increase the model quality and to ensure a stable process with maximum throughput and product quality. A standard procedure for obtaining these parameters is the measurement of the deformation resistance in a laboratory. Those measurements can be used for a regression onto one of the formulas shown in Equations (4.1) to (4.5). Of course, other formulas exist, and might be used for regression. Especially when trying to model the deformation of micro alloyed or high alloyed steel or when complex materials with phase transformations should be described these other models might be more suitable.

4.3.2. Models Used in Industry

There are many analytical models for the description of the roll force. One of the most common analytical models, which is used for the calculation of the roll force, is based on the elementary rolling theory [Web73, Hin03]. Some of the limitations of that theory are compensated with correction functions. For example, one of the requirements for the elementary theory is the complete plastic deformation during each pass.
As described in Table 2.1, Aluminum ingots have usually a thickness of above 500mm. For thick products like this, the assumption of complete homogeneous deformation is not valid anymore. The analytical model holds compensation and correction function which were empirically determined to correct these inaccuracies. For the calculation of the roll force in each deformation step, the contact zone between product and rolls are divided into strip-like elements and the force balance for each stripe is calculated. The solution of this equation system yields to the basic differential equation of the plastically deformation theory which was developed by Karman in 1925 [Kar25]. The flow resistance for each deformation is a major part of this equation system and is highly dependent on the material, the deformation φ, deformation rate $\dot{\varphi}$ and temperature ϑ. It is calculated with the flow curve

discussed in the previous section. Furthermore, the forces induce temperature into the material so the calculation of the next deformation depends on the previous deformations. This is valid not only for reversing mills. The principle is the same for finishing mills, where the next deformation is made in a separate stand.
The roll force is linearly correlated to the parameter k_f. Intuitively also an optimization without simulating the whole process would be suitable. But since the roll force is inducing and changing the temperature of the product the next deformation would take place at different temperatures and originally calculated values are not valid anymore. Therefore, if the parameter of the flow curve changes, the whole process has to be simulated again. After the new simulation, the calculated roll forces based on the new flow curve can be compared with the original feedback, i.e., measurements of roll force, torque, temperature, and speed. If the parameter are only optimized with data of a single product, the parameters φ, $\dot{\varphi}$ and ϑ may not vary enough. The resulting optimization might not be generalizing good enough for other areas of deformation. Therefore it is preferable to consider a campaign with a wide variation of product geometries, temperature ranges, and deformations.
Currently, the available concepts for the optimization of flow curve parameters are mostly dealing with determination of those parameters in laboratory rather than optimizing those parameters with real process data. Traditionally, the parameters are measured with small samples of one piece in laboratory and are then generalized for every material which is close to the sample in terms of material composition. Some companies are modifying the flow curve parameters with linear models. That is, they are determining the prediction accuracy of their model and are varying some of the influence parameters. Most of the research in this area is on the development of suitable flow curve equations especially for high and micro alloyed steel rather than using a data driven approach for the optimization of those parameters. Lin [YL08],e.g., analyzes the deformation resistance of 42CrMo steel for high strain rates, Hernandez [HMR96] describes the deformation resistance within the austenite phase of micro alloyed steels and in [MRS+09], the authors describe the deformation behavior for Ti-alloyed austenitic steel.

4.3.3. Problem Description

Flow curve parameter determination in laboratory as described in Sec. 4.3.1 requires several weeks and costs several thousand Euro for each material. Sometimes, this is not affordable and therefore not a suitable way to determine those parameters. Hence, cheaper estimation of flow curve parameters have to be found. Due to their highly nonlinear behavior, the flow curve equations cannot be solved directly. Furthermore, a different roll force would result in a different temperature

balance of the product and thus the temperature in the next pass differs from the original calculation. Because these aspects cannot be neglected, the whole process has to be simulated when testing new parameter sets for the flow curve of a specified material. The calculations of the roll forces and roll torques within this simulation are afterwards compared with the measurements to get a quality criterion for the new parameter set. The detailed description of the simulation scenario is presented in Section 4.2.
It is important that the simulation scenario behaves in the same way as the online process. Therefore, the simulation uses feedback of the measured speed, the reduction, and temperature to calculate the new predictions for roll force and roll torque. This enables the simulation to achieve the same working point as in the online process. Another problem can be the amount of data. The simulation of a whole batch where only one material group was rolled consists of thousands of different deformation steps and will therefore be highly expensive in terms of simulation time. Several optimization algorithms require boxed constraints of the optimized parameters. In our case, parameters A and m_i are dependent on each other. The only limitation which can be set is a plausible region for the resulting value k_f of the basic deformation. In hot mills, the maximum basic deformation value for k_f is usually below 300 $\frac{N}{mm^2}$, but always positive. Then, for a given maximum working range of the deformation, deformation rate, and temperature, the feasibility of the parameter set can be tested. Due to the fact that every company has usually its own classification system it might be that materials which are grouped together in one company are separated in other companies. In this case, the optimization, which has been done in the first company cannot be directly used for other companies and has to be renewed every time. Summarizing, it is desirable to optimize the process in order to

- reliably estimate valid flow curves,
- reduce lab costs,
- save time,
- determine parameters in their working environments, and
- make the process more flexible and adapt to new (material) changes quickly.

4.4. Flow Curve Parameter Optimization

For the development of new materials or if the material behavior cannot be deduced from similar grades, the flow curve parameters are unknown and no proper parameter guess is available. Therefore, an initial screening with the SPO framework can be used prior to a local optimization.

Algorithm 3: SPO-based hot mill simulation tuning.

```
// phase 1, collect initial knowledge about the optimized
   process:
```
let A be the hot mill simulation model we want to tune;
generate an initial design DES $= \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$ of n parameter vectors;
let $k = k_0$ be the initial number of replications for determining estimated responses;
foreach $\mathbf{x} \in$ *DES* **do**
    run A with $\mathbf{x}$ to determine the estimated response y of $\mathbf{x}$;

```
// phase 2, building, using and improving a surrogate model:
```
while *stop criteria not reached* **do**
    build surrogate model $Y(\mathbf{x})$ based on DES and $\{y^{(1)}, \ldots, y^{(|\mathrm{DES}|)}\}$;
    optimize the model w.r.t some cost function and constraints, thus produce a set DES' of d new parameter vectors ;
    run A with each $\mathbf{x} \in$ DES' to determine the response;
    extend the design by DES $=$ DES $\cup$ DES';

```
// phase 3, final exploitation and fine tuning:
```
use local optimizer for the best p parameter sets $\mathbf{x}_{1\ldots p} \in$ DES, without constraints

4.4.1. The Sequential Parameter Framework

One framework for surrogate-model based optimization is *sequential parameter optimization* (SPO) [BBLP05]. SPO combines methods from classical Design of Experiments (DoE) and modern *Design and Analysis of Computer Experiments* (DACE) [BB03, BBPV04] based on Kriging models. The algorithm was adapted for hot mill parameter optimization task. Algorithm 3 presents the pseudo code of SPO. Note, that the notation $\left(x^{(i)}, y^{(i)}\right)$ for the data from the i-th pass is used. This data is passed to the surrogate model. The algorithm was divided into three phases. Within the initial stage, SPO explores the search space of the optimization problem A. The optimization is treated as a black box. For the initial phase a certain pre-defined number of design points $\mathbf{x}$ is passed to A. The strategy for the creation if the initial design is usually a space filling design, e.g. Latin hypercube sampling. With every design point the black box optimization problem is called and the objective function returns some output y representing the performance value. In the second phase SPO tries to determine a functional relationship between $\mathbf{x}$ and y. SPO thus uses a model $Y(\mathbf{x})$ as surrogate for the hot mill simulation model A. As mentioned above, the chosen model type is Kriging. Kriging is frequently used for surrogate-model based optimization, because it provides a powerful and flexible predictor. It also provides an estimate

of the variance or error of each prediction. The observations are interpreted as realizations of a stochastic process. A Gaussian kernel is used to model the correlation between observations [SWMW89].
In the sequential improvement loop SPO optimizes the surrogate model $Y(\mathbf{x})$ over the considered space of input variables by means of a cost function. Once the new set of design points DES' has been selected, the required evaluations of DES' are performed. Based on DES', the surrogate model $Y(\mathbf{x})$ is updated. In the second phase at line 8 of the of Algorithm 3, a search on the surrogate model is performed. During that search the inequality constraints of the mill parametrization problem have to be considered. As the constraints are not expensive to evaluate, they are evaluated together with the surrogate model itself. For the inequality constrained optimization, we use the popular method developed by Powell [Pow94, Pow88], which does not require any derivatives of the objective function to be available. During this optimization step 8, the next point $\mathbf{x}$ to evaluate in the sequential loop of SPO is determined. For expensive, global, black-box optimization Jones [JSW98] introduced *efficient global optimization* (EGO). EGO exploits the information given from a Kriging model, i.e., the predicted mean and variance, to compute the *expected improvement* (EI) of a given solution. EI can hence be used as a cost function during step 8, as an alternative to the predicted value of the Kriging model.
In the last phase of the optimization (line 11), the well known downhill simplex algorithm introduced by Nelder and Mead [NM65] is used to improve the best found results by a local optimization procedure. The algorithms uses the downhill simplex implementation in the `nloptr` R package. During local refinement, constraints are disregarded because they no longer play a role in the region of good solutions.

4.4.2. Experimental Setup

For the first mill problem, the parameter optimization was based on Equation (4.2). The feasible range was $0 \leq k_f \leq 300$. That is, solutions that result into negative k_f values or k_f values larger than 300 are considered to be infeasible. The usual working point for the test data was in the following range:

$$0 \leq \varphi \leq 0.5, \quad 0 \leq \dot{\varphi} \leq 600, \quad 500 \leq \vartheta \leq 600.$$

With that said, the optimization problem to be solved in this study is defined as follows:

- Parameters to be changed are the flow curve parameter vector $\mathbf{m}$ and the consolidated parameter A of the flow curve.

Table 4.2.: Upper and lower bounds for the parameter set $\mathcal{M}$ introduced in section 4.3.1 which was used during the optimization. All parameters are of type FLOAT.

Factor	Low	High
A	0	2
m_1	-0.01	0
m_2	-0.3	0.4
m_3	0	0.2
m_4	-0.1	0.1

- The target is to minimize the deviation of simulated roll force from the measured roll force.
- Computational constraints: The evaluation of the objective function is expensive.

The parameters which represent the search space and were subject to optimization in this study are summarized in Table 4.2.

The success and improvement of the optimization which is the central aspect of the roll force optimization problem was validated against a parameter set which was used in practice so far. This parameter set was selected by experts according to the closest distance to a known material. With closest distance the relationship according to chemistry is meant. This is a valid approach since beside the roll force deviation of the model calculation an additional online adaption is used to compensate errors. These online models are described and analyzed in the next chapter. However, a good and solid model calculation is always the best precondition for good rolling quality. The residuals which cannot be compensated with the data-driven approach have to be compensated directly during rolling.
The selected flow curve will therefore be used as a baseline reference for the optimizations. The *SPO toolbox* (SPOT) was selected to conduct the experiments [BBZ12]. Parameter used for SPOT are set as follows:

seq.predictionModel.func: The chosen surrogate model is Kriging, based on code by Forrester et al. [FSK08].

init.design.size: The initial design consists of 40 candidate solutions, which are created by Latin Hypercube Sampling (LHS).

auto.loop.nevals: Overall, the number of evaluations of the objective function are varied (hot mill simulation) with different sequential design. The sequential design either consist of 10 evaluations of the objective function or

60 evaluations of the objective function. Therefore, the total number of evaluations were 40, 50 or 100 which includes the initial design of 40.

init.design.repeats: Since the objective function is deterministic each solution is evaluated exactly once.

seq.design.new.size: In each sequential step, one new solution is evaluated on the target function.

seq.predictionOpt.method: The sequentially created models are optimized by Latin Hypercube Sampling (LHS) and Constrained Optimization by Linear Approximation (COBYLA). The COBYLA implementation from the NLOPT library, included with the `nloptr` R-package is used.

seq.design.size: The sequential step size. LHS evaluates 200 points with the model.

seq.predictionOpt.budget, seq.predictionOpt.restarts COBYLA has a budget of 1 000 evaluations of the surrogate model, and will restart if it converges prematurely.

Table 4.3 summarizes the these chosen values for each parameter.

Table 4.3.: Spot settings. A detailed description of each parameter can be found in Section 4.4.2

Parameter	Value
seq.predictionModel.func	spotPredictForrester
seq.predictionOpt.func	spotModelOptim
seq.predictionOpt.method	`"NLOPT_LN_COBYLA"`
seq.predictionOpt.budget	1 000
seq.predictionOpt.restarts	TRUE
seq.design.size	200
seq.design.new.size	1
init.design.size	40
init.design.repeats	1
auto.loop.nevals	40/50/100
seq.infill	spotInfillExpImp/NA

4.4.3. Optimization Process

The description of the optimization process is shown in Figure 4.4 and is separated into three stages. The first stage is the initial design (I) and is followed by an

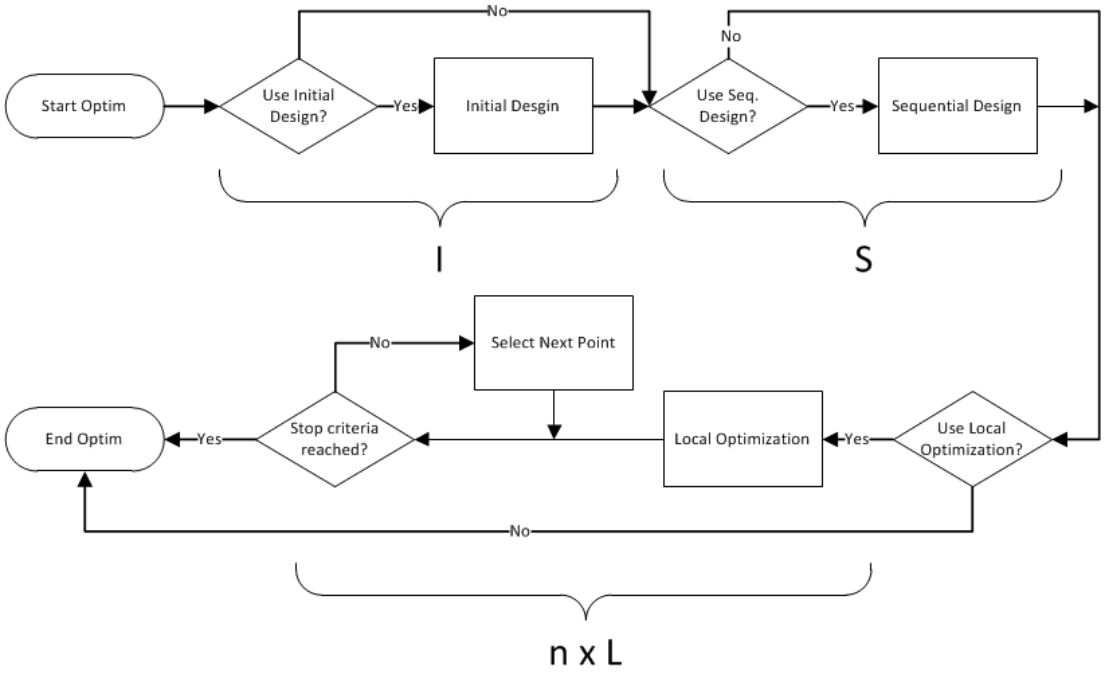


Figure 4.4.: Flow chart of the simulation presented in this section. First of all an initial design (I) is evaluated followed by an sequential design (S). Afterwards a local optimization (L) is performed.

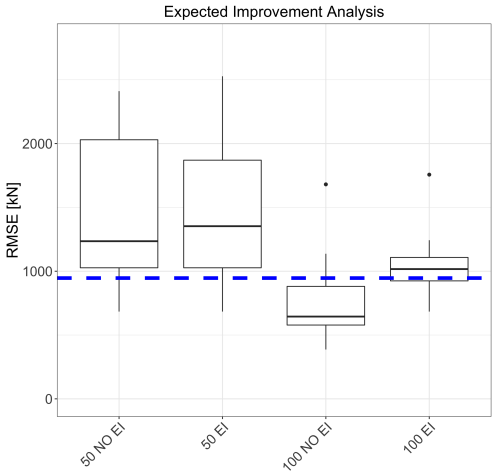
sequential design (S) based on the results of the first stage. The last stage consists on local optimization (L) based on the second stage. In order to distinguish between the different settings the notation $I/S + n \cdot L$ is used to characterize the settings for the different stages. I/S describes the size of the initial design respectively the sequential design and n describes the number of points used for the local optimization. the number of evaluations in the local optimization stage is characterized by L. An experiment conducted with $40/10 + 5 \times 100$ has a design size of 40, 10 sequential designs and based on these results a local optimization on the 5 best points is made.

4.4.4. The Influence of Expected Improvement

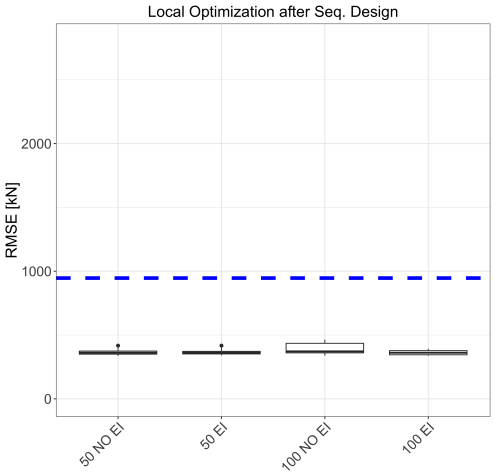
With the used Kriging model the predicted mean and variance are used to calculate the expected improvement (EI). The idea of EI is to generate new designs in areas where only little data is available and the uncertainty of the model is very high. Furthermore, also designs in areas with an expected high performance is generated. For both proposals the constraint is always considered. The question is, if it will be beneficial to use the EI infill criterion in the sequential design. To answer this question, a scenario with an initial design size I=40, sequential design size S=10 or S=60 and 10 repeated evaluations was chosen. The experiments with a sequential design size of 60 outperformed the experiments with a sequential design size of only 10. No statistically significant difference between the experiments with EI and without EI can be seen so EI seems not to improve the results even with larger sequential design. In fact it is quite the reverse: The performance of the model-based optimization decreases if EI is used. This is owed to the fact that the total number of evaluations is higher.
The experiments with a sequential design size of 10 had a median RMSE of 1235 (no EI) and 1352 (EI) and the experiments with the greater design size of 60 had a median RMSE of 644 (no EI) and 1016 (EI). The results are summarized in Figure 4.5(a) . To answer the question of the influence of EI it was shown that EI seems to drop or at least not improve the optimization. A possible reason for this might be that the problem is too simple and that exploring new regions is not beneficial. The major improvement seems to come from the initial design and the local optimization afterwards which is subject of the study in the next section.

4.4.5. Influence of Local Optimization and Initial Designs

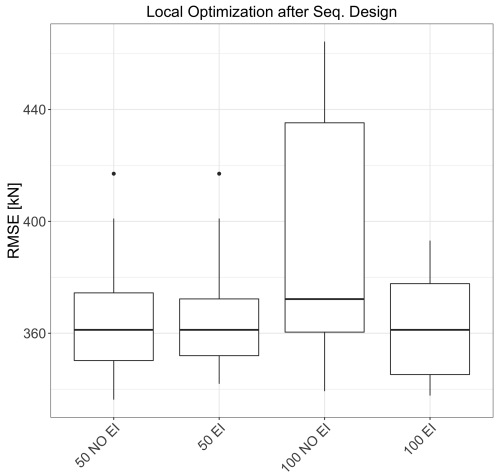
The same comparison as in the previous section was made for the additional refinement of the best-found results with local optimization. Here, the best five parameter sets found by the Kriging-based optimization were selected and opti-



(a) Analysis of the Kriging simulation with an Initial Design Size of 40 and a sequential design Size of 10 and 60. Shown are the results for the SPOT runs with and without EI based on 10 and 60 evaluations in the sequential design phase. The number in the labels are indicating the total number of evaluations on the original problem.



(b) Analysis of the results after local optimization of the previously results found with Kriging. Shown is the best point in each seed after 100 local optimization runs of the 5 best points.



(c) Zoom of the analysis after local optimization shown in Figure 4.5(b).

Figure 4.5.: Comparison of the results for Kriging with and without Expected Improvement after sequential design (a) and after the local optimization (b). The blue dotted line shows the reference value for the original parameter set. In order to compare the local optimizations, Figure (c) shows the boxes with different scaling.

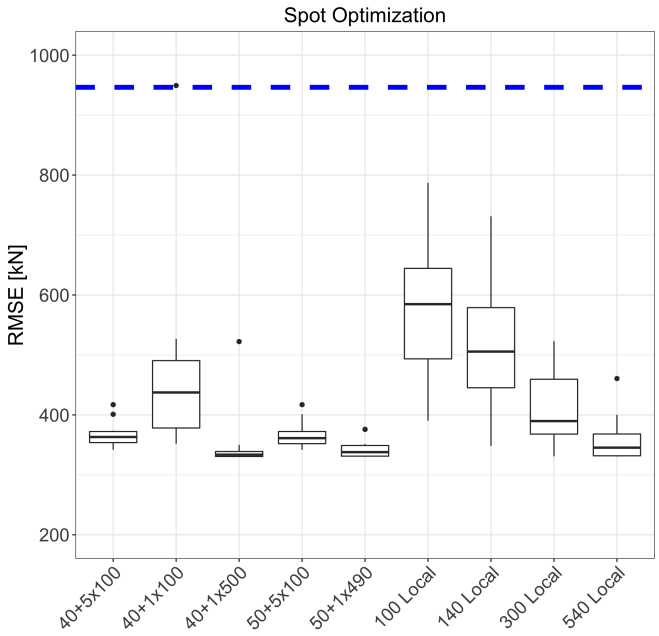


Figure 4.6.: Comparison of simulation results for an initial design of 40 points with different local optimization afterwards against a pure local optimization with a random starting point. As a reference box also a result for a simulation with a sequential design and local optimization is shown ($50+5\times100$). The boxplots are representing the results for 10 different seeds. The number of parameter evaluations in case of a previous initial design is $40+5\times L$, where L denotes the number of local optimization steps. The blue dotted line shows the reference value for the original parameter set.

mized with a Downhill Simplex approach. Each of the five parameter sets received a budget of 100 evaluations for further improvement. The local optimization improved the median of the RMSE to below 500 in all experiments whereby only one experiment without the local optimization had a resulting RMSE below 1000. This was the case for an initial design size of 40, a sequential design size of 60 without EI. The local optimization yields the major improvement independent of the budget of the model-based optimization. The results are shown in Figure 4.5(b). Thus, the question is whether model-based optimization can be skipped, and whether a local optimization directly after the initial design is more profitable. In addition, it was also checked whether it can be beneficial to choose completely randomized starting points for the local optimization. The results of this investigation is shown in Figure 4.6. To have a decent comparison the number of evaluations should also be taken into account. The boxplot is showing the statistic for the best point produced by each of the 10 random number generator

Table 4.4.: Numerical values for the box plots shown in Figure 4.6. Shown are the number of evaluations in the initial and sequential design and during the local optimization. Box plot parameters listed are minimum, lower quartile (Q1), median, upper quartile (Q3) and maximum value.

Init.	Seq.	Local	Min	Q1	Median	Q3	Max
40		5x100	342	352	363	374	401
40		1x100	352	378	437	498	527
40		1x500	331	331	334	340	350
40	10	5x100	342	352	361	374	401
40	10	1x490	331	331	338	351	376
		100	390	479	585	652	787
		140	348	442	506	580	731
		300	331	366	390	470	523
		540	331	331	345	371	400

seeds. To analyze the importance of the initial design and the local optimization the experiments from Table 4.5 were performed.

The completely local optimization (with random starting point) performs similarly good as the two approaches where the five best points are each optimized locally. The best approach seems to be $(40 + 1 \times 500)$.
As a result of the investigation in this section it can be concluded that it is more profitable to optimize just the single best point found in an initial design.

4.4.6. Impact of the Fitness Landscape

However, the completely randomized, local approach is not much worse than the approaches with an initial design. This indicates that the problem landscape is rather simple. If the landscape would be univariate the optimizations would always result in the same best parameter set independent of the seed. There can be seen huge differences in the parameter set which are shown in Table 4.6. This gets more clear if we keep in mind that the first parameter is a multiplicative component and is therefore linearly correlated to the flow resistance. The corresponding contour plot is shown in Figure 4.7. While the contour plot depicts several basins in the fitness landscape, they are all of some considerable size. Note that the contour plots of other parameter combinations result into even simpler landscapes, most giving the impression of being unimodal.

Table 4.5.: Experiments, which were performed to analyze the impact of the initial design and local optimization.

Name	Description	Evaluations
40+5x100:	Initial design of 40 points with 100 local optimization steps of the best 5 points.	540
40+1x100:	Initial design of 40 points with 100 local optimization steps of the best point.	140
40+1x500:	Initial design of 40 points with 500 local optimization steps of the best point.	540
40/10+5x100:	Initial design of 40 points, model-based optimization with 10 points, and 100 local optimization steps of the best 5 points.	550
40/10+1x490:	Initial design of 40 points, model-based optimization with 10 points, and 490 local optimization steps of the best point.	540
100 Local:	Random start point with 100 local optimization steps.	100
140 Local:	Random start point with 140 local optimization steps.	140
300 Local:	Random start point with 300 local optimization steps.	300
540 Local:	Random start point with 540 local optimization steps.	540

Table 4.6.: Parameter comparison for local optimizations with an arbitrary starting point and a budget of 540 evaluations. Only seed 2 stopped earlier, i.e., after 431 evaluations. Shown is the best parameter of each seed with the corresponding RMSE value. The parameters are used to calculate the flow stress according to 4.2.

Seed Unit	A	m1 e-03	m2 e-02	m3 e-02	m4 e-02	RMSE kN
1	1.2	-4.8	-0.2	8.2	2.6	331
2	1.2	-4.8	-0.3	8.1	2.6	331
3	0.9	-4.6	-2.7	11.0	2.8	339
4	1.2	-5.0	-5.7	9.8	2.3	335
5	1.0	-5.3	-18.0	14.0	1.8	371
6	0.7	-4.2	-3.2	11.0	2.9	359
7	0.8	-3.5	16.0	5.9	3.9	400
8	1.2	-4.8	-0.5	8.2	2.6	331
9	1.4	-5.5	-6.0	11.0	2.7	352
10	2.9	-5.7	0.8	-0.58	0.95	461

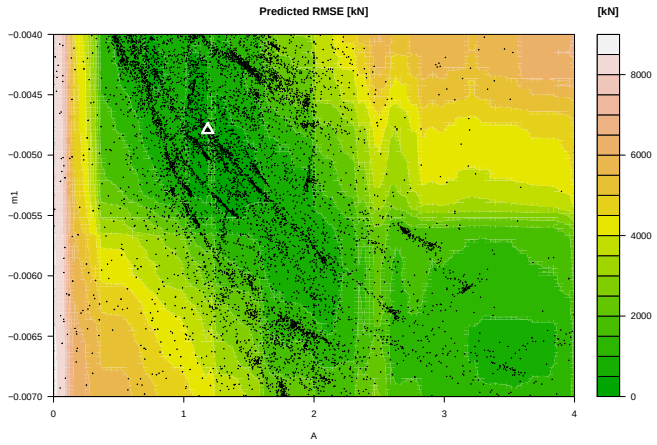


Figure 4.7.: Contour plot for parameters A and m1. All other variables are kept constant at the same value as for the best case. A random Forest model was created with all available simulation results and is used here for the prediction of the RMSE, denoted by the background coloring. Each black point indicates the result of a simulation. The white triangle marks the simulation with the best RMSE.

4.4.7. Comparison of the Resulting Roll Force Deviations

Figure 4.8 shows the deviations of the pure model roll force (red), the adapted roll force (blue) and the measured roll force (green) of the original parameter set, before optimization. Figure 4.9 shows the same for the optimized parameter set. The model roll force is highly correlated with the calculation of the parameter k_f which was described above. The calculation of the roll force was done in a production scenario. According to previous knowledge of the rolling process the system learned the roll force deviation for this material and made an online regression for the correction of the calculation. Furthermore a pass to pass adaption was used to correct the roll force deviation. That is the reason why the adapted roll force is often close to the measured roll force. Due to limitations to the adaption system some large deviations remain, especially when the model roll force estimate is way off.
Typically, model corrections of up to 30% are tolerated in the online scenario. This limitation is used because something else has to be wrong if calculation errors are exceeding this value. Furthermore, the adaption is reset from time to time. In case of the original parameter set, the deviation were more than 30% and therefore we still have a residual deviation for the rolling force.
In both figures, Figure 4.8 and Figure 4.9, the values for the pass number 15 are missing. This is due to a special strategy during rolling of this material. In this early stage of commissioning there were still some difficulties during measurement preparation. Usually, pure model deviations without any optimization of about 10% are tolerated and deviations of less than 5% are considered to be good. In Figure 4.10, the roll force deviation of the model with the original, standard parameter set is shown. Next to it, the roll force deviation of the model with the new parameter set is plotted. It can be seen, that the deviation is very good in the first passes but still shows some deterioration in the last passes. This phenomenon accounts for wrong temperature calculation and the process was often delayed which was not correctly modeled during this phase of the process. If these parameters are optimized in a later stage of the project, the deviation in the last passes will also be reduced to a suitable amount. When the calculated roll force deviates from the actual measurement the automation system is correcting the roll gap position based on the current roll force. This is done in order to compensate for the mill spring. The mill spring is the spring which occurs in the stand when we suddenly have to apply a roll force. This means that the gap between both rolls will be higher and we have to adjust the gap setting to reach a specific thickness. If we consider this opening as a linear function of the roll force and assume a value of 1mm at a roll force of 6 000kN then we would have an error of 0.5mm when the prediction is 3 000kN away from the measured roll force. When optimizing this prediction, the length of the final product, which is

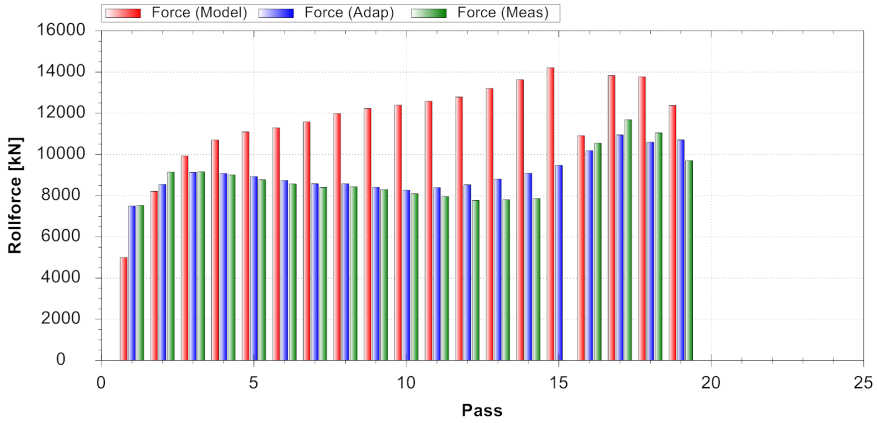


Figure 4.8.: Roll force of pure model (red), model with online adaption (blue) and measured roll force (green) with original parameter set. The online adaption is typically a simple regression based optimization which learns the difference between model and measured of the last 2 month of production. Additionally the difference of the last pass between prediction and measurement is taken into account. The adaption is used as an additive offset to the model prediction. Because it can be reset and should only dynamically react on slight deviations from unknown source the correction amount is limited. This limitation can be seen especially in passes 12-14 where we still have a residual error. The missing measurement in pass 15 is due to a special strategy and problems during the measurement preparation in this pass.

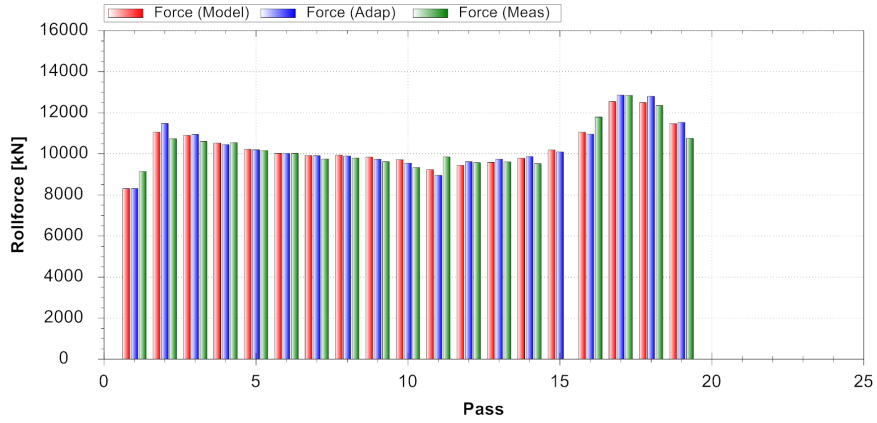


Figure 4.9.: Roll force of pure model (red), model with online adaption (blue) and measured roll force (green) after optimization. Here we have used the parameters of our offline simulation, i.e., the result of the local optimization. This figure clearly demonstrates that the performance of the model has vastly improved and that the online regression could have been switched off.

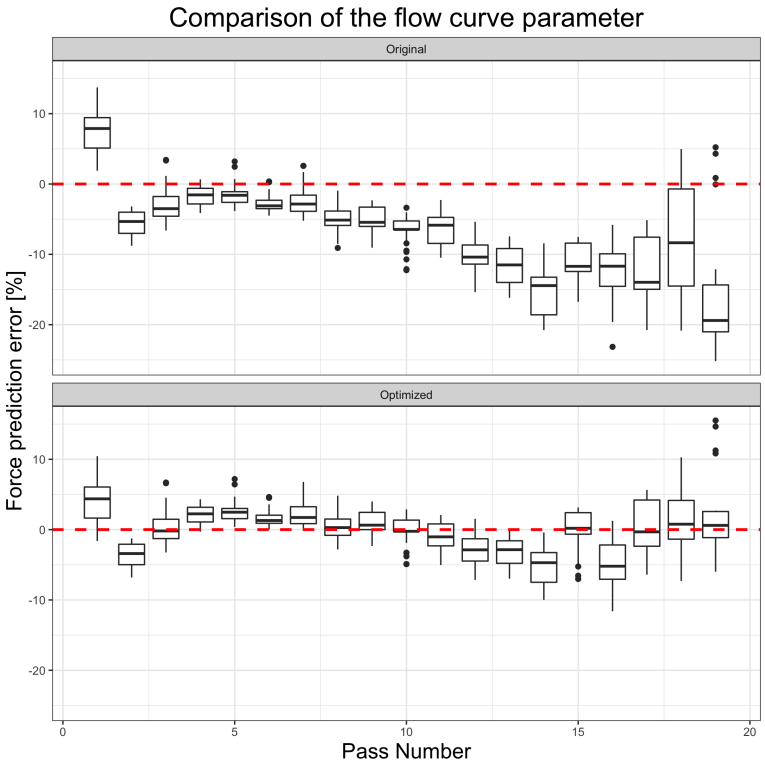


Figure 4.10.: Relative roll force deviation plotted for each pass. The error is plotted as the relative roll force error `Measured Force - Calculated Force`. The top plot shows the prediction error with the original parameter-set and the bottom one uses the optimized parameter-set. The drift which is present in the original parameter set is reduced with the optimized parameters. Furthermore, also the IQR is reduced with the usage of the optimized parameter.

in an acceptable range, can be increased.

4.5. Parameter Optimization for Known Materials

There are several situations where materials are well known but still achieve a performance which is not sufficient. Those situations can be caused by the following problems.

- Measurement equipment or test scenario in laboratory.
- Different chemical analysis than in laboratory, but within the specifications.
- Process description not fully available.

Sometimes, also parameters of a similar material are used to have an initial guess of the behavior. If the deviations are not caused by some failure in other parts of the description and can surely be addressed to the flow curve, then an optimization of the parameters have to be made. It will now be assumed that the performance is already within some reasonable limits and that the optimization is only used for fine tuning of the parameters.
Then, instead of initial screening of the whole region of interest, a simple local optimization with the downhill simplex algorithm can be performed. The algorithm and concept of the previous section can still be used. The result of the local optimization of such a scenario is shown in Figure 4.11 where the minimum achieved RMSE over the number of local optimizations is plotted. The data were coming from a aluminum finishing mill and the material which was optimized was a magnesium based alloy.
It can be seen that the major improvements were achieved during the first 100 evaluations. The initial RMSE off 11.5% was already in a reasonable region. During the optimization it was decreased to a value of 8.6%. This value might be further decreased by the use of online optimization algorithm which will be introduced in Part II.

4.6. Parameter Optimization for Multiple Mills

Up to now the optimization procedure was conducted on real-world data for a single mill type only. All mill specific errors are compensated now by the flow curve parameters. Common mill specific errors are

- measurement errors (device specific),

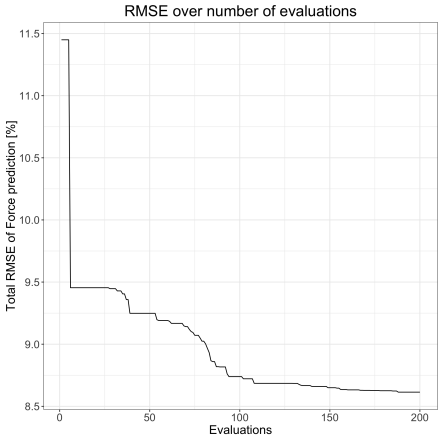


Figure 4.11.: Minimum RMSE over a local optimization with 200 evaluations. The performance is measured as total RMSE of the roll force for all simulated products.

- construction errors (stand specific),
- different constitution of same material grade and
- input errors (production planning or calibration).

Especially input errors should usually be avoided and are sometimes very hard to detect. An example of a trivial input error is the roll diameter which is coming from the mill operator or the roll shop where the rolls are grinded. Also the ingot description may cause deviations which are not always easy to detect. For aluminum rolling, where the ingots are usually faced, at least the geometry description should be accurate. Nevertheless, deviations of the ingot description are very common. For steel rolling the chance of input errors is much higher since the slabs are not faced.
Other problems for aluminum rolling may origin in the used emulsion. The used emulsion is often kept secret and is very specific to the mill owner and mill type. Therefore, no complete description of it is available but the impact on friction and rolling is quite high [SLSJ02].
As a consequence, optimized parameters for one rolling mill might not be ideal for another mill of the same type. The problem gets worse if different mill types are considered. E.g. the usual temperatures in aluminum roughing mills are 400-500 degree C and usual target temperatures for the finishing mill are between 250 and 350 degree C. So the optimization of parameters with data from the roughing mill are optimized in a different region compared to the usual rolling in finishing mill. Furthermore, when rolling in finishing mill multiple stands are rolling the product at the same time and a specific tension between those mills is used and controlled.

This tension can also have a high impact on the roll force.
So, two questions arises for the optimized parameters.

- How is the performance of the optimized parameters on other mills of the same type?
- How is the performance of the optimized parameters on other mill types?

For answering the first question a new experiment has to be conducted since no data on other mills were available for the material optimized above.
First, a flow curve is optimized with real-world data in the same way as described in the previous section. The data is coming from an aluminum hot strip finishing mill. To answer the first question, the optimized flow curve parameter are used on another finishing mill. Here, real-world data of the rolling process for the same material is used. The chosen performance parameter is the RMSE of the roll force deviation for the simulated products.
The box plot in Figure 4.12 shows the achieved performance for both mills. The boxes are representing the force prediction errors of all products and all stands for two different mills, A and B. Further, the Figure shows the result for flow curve parameter optimized on the specific mill, $A(Opt)$ and $B(Opt)$ and the performance when the parameters are optimized on the other mill, $A(BestB)$ and $B(BestA)$. The simulated material was a manganese aluminum alloy. The RMSE of the first mill (FM A) cannot be compared to the RMSE of the second mill (FM B) since the product variety, the measurement equipment and the working areas in terms of φ, $\dot{\varphi}$ and ϑ are different. This is also the reason why the initial RMSE shows huge differences, although the same original flow curve parameter are used. These parameters were chosen because the material behavior was expected to be close to another known material group. The optimization of the flow curve with real-world data from Mill A lead to an enormous improvement of roll force prediction accuracy. This can be seen in the second box (A Opt.). The optimization procedure presented in the sections before could not only improve the mean and median prediction error but also the standard deviation of the accuracy. While the prediction error with the original parameter had a mean of -4 413kN and a standard deviation of 3 359kN, the optimized parameters achieved a mean prediction error of 161kN with a standard deviation of 1 766kN.
Unfortunately, the same parameters did not manage to improve the prediction accuracy on another mill of the same type (B Best A). Although the standard deviation was improved from 4 991kN to 2 028kN, the mean prediction error increased from 2 732kN to 4 690kN.
The reason for this behavior can be caused by the material description and classification. The considered material was a magnesium based alloy. The deformation resistance and herewith the roll force is highly dependent on the magnesium

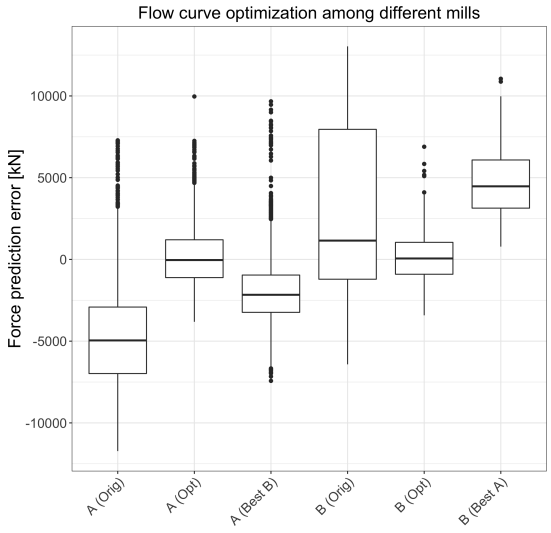


Figure 4.12.: Optimization of the flow curve among different mills of the same type. In this experiment, the flow curve was optimized on two aluminum finishing mills (A and B) separately. Then, roll force prediction error on the other mill was measured. Additionally the residual errors of the original flow curve on both mills is shown.

content. The range for a specific type of aluminum alloy may be e.g. 4.5%±0.5% which can cause differences in roll force of approx. 10%-20%. This variation of magnesium can lead to higher variance in the prediction and incorrect description of the flow curve. A possible solution would be either to create subgroups for this alloy with closer ranges for elements with high impact on roll force or to create a flow curve formula which reflects theses changes.
Also mill related phenomena like the composition of the applied emulsion or measurement inaccuracies can lead to these effects.
Since also all errors which occur in mill A are compensated by the optimization, a suitable solution would also be to optimize the multiplicative factor for the flow curve. Reconsider formula 4.1 which is the basic description of all flow curves. The multiplicative factor A can be used here to optimize the flow curve prediction. This is reasonable because the basic shape of the flow curve which is reflected in the variance of the prediction has improved for both mills. The factor can now be optimized among different mills or just for every mill. An optimization among different mills would decrease the performance for mill A but improve it for mill B. The resulting residuals can then be compensated by online algorithms which are discussed in detail in the next chapter.
Because the flow curve should represent the material behavior and not mill characteristics the answer to the second question is of great importance. To test the

Table 4.7.: Summary of the boxplots shown in Figure 4.13.Beside the quartile values, the median the minimum and maximum, also the standard deviation is given. All values are in kN.

Name	Median	Std. Dev.	Min	Max	25%	75%
RM (Orig.)	112	7764	-3173	23506	-2379	12239
RM (HSM Opt.)	-6977	7835	-9637	23300	-8881	-1528

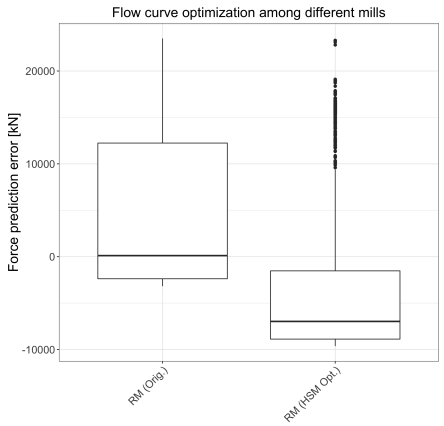


Figure 4.13.: Boxplot for the roll force prediction error when using a flow curve from a similar material (left) and a flow curve optimized from the finishing mill (right).

optimized flow curve parameter, they were used for roll force prediction of an aluminum roughing mill (RM). The same material as in the previous test was used for the simulation, i.e., a magnesium based aluminum alloy. Figure 4.13 shows the residual roll force error for an aluminum roughing mill. The box on the left side represents the errors when a flow curve of a similar material is used. Here, the material was selected due to best knowledge of process experts. These predictions have been compensated by other algorithms during rolling. The box on the right side shows the residual prediction errors from an optimized flow curve. The optimization of the flow curve was done on a finishing mill.

The summary of the boxes is given in Table 4.7. It can be seen that the standard deviation cannot significantly be reduced in comparison to the original curve. Both flow curve parameters show high residual roll force deviations which have to be compensated by other algorithms. Otherwise the rolling process will be unstable. The reason for this behavior is due to the different regions of the flow curve which are used in both mills. In a finishing mill, the usual temperatures are below 400 degree Celsius. This is almost the minimum temperature for roughing

mills.
A possible solution would be to extend the optimization scenario to consider multiple mills. Then, the number of samples available for both mills have to be considered and a global performance value has to be selected. Another solution would be to optimize the flow curve for the selected mill types. Then different parameter sets will be generated dependent on the temperature or other criteria.

4.7. Summary

The procedure presented in the previous section showed a model-based optimization for flow curve parameters which are typically used in rolling mills. A combination of global and local optimization approached were used. Data from various sources and mill types were used to conduct the experiments. For the first study, data was collected from an aluminum rolling mill where no information about the material was known prior to the first rolling trials. The second study is based on data from various type of mills, i.e. two different aluminum finishing mills and a roughing mill. Here, some knowledge about the material was available and an initial setting of flow curve parameter was used. Both studies showed a significant improvement in model prediction accuracy which makes it possible to evaluate and optimize flow curves without conducting time consuming and expensive measurements in a laboratory.
Especially the study in Section 4.6 showed the generalizability of the optimized parameters. When optimizing parameters on a specific mill they may not be suitable for the usage on other mills since all plant specific errors are compensated during the optimization procedure. Instead, conducting experiments for multiple plants and especially multiple mill types results in optimized parameters for the material. This also ensures that a broad range in terms of validity of the flow curve which was used by the algorithm. When only data of a single mill is available then the optimization procedure should be slightly modified. In roughing mills the temperature usually does not drop below 400 °C for aluminum and 1 000 °C for steel. Therefore, the temperature may not be optimized for low temperatures. A possibility to overcome this problem is to neglect the temperature parameter during the optimization and to optimize only the remaining parameters. This is especially helpful when default parameters for a material are available and the task is only to optimize those parameters. In case no parameters are available a default value can be retrieved from *similar* materials where a similar material has to be chosen by process experts.
After optimizing the flow curve parameters they should be visualized and compared against known flow curves of similar materials.

Finally, the surrogate-modeling method employed here may profit from computationally cheaper information that can be gained from a coarse variant of the objective function. In detail, a parametrization of the mill model may be tested with less data. This results into faster evaluation times, but yields a less accurate estimation of the quality. This cheaper information can be used in tandem with the more accurate, expensive data to train a better surrogate-model. To that end, Co-Kriging [FSK07] can be employed. Co-Kriging allows to exploit correlation between coarse and fine target functions. This may result into a much improved surrogate-model of the hot-mill, without the requirement of additional expensive evaluations.

Part II.

Online Optimization

5. Introduction to Online Optimization

Industrial processes require a high flexibility without losing robustness and generalizability. Process predictions have to be fast and accurate. Offline models as described in Chapter 4 are suitable for improving the general process models but fail to adapt to dynamic changes. For these dynamic adaptions, online models are perfectly applicable. Throughout this thesis, the term online does not mean that the process is connected to the internet, but instead refers to the *on-the-fly* adaption of the algorithms.
This chapter presents several concepts and algorithms to improve predictions without reducing the robustness that are provided by offline algorithms. Typical online optimization mechanisms, which are traditionally used within many industrial processes, are building clusters and are using average values within each cluster. The clusters are build either by discrete values directly or by partitioning of continuous variables. Prominent values which are used within the context of hot rolling are e.g. the material or the used reheating furnace. The most important continuous values are thickness, width, profile and temperature. For each unique combination of theses values an average prediction error is collected. Problems which arise with this concept lie in the inhomogeneous boundaries. Most often, building several groups out of a continuous variable does not make any sense and using a regression is a traditional approach to solve this problem. One of the promising candidates for industrial online optimization is online support vector regression which is an enhancement to classical support vector regression. Standard support vector regression lacks the possibility to incrementally add new observations without retraining of the complete data. It has already been successfully applied to hot rolling problems [BYHB10, HELR08]. Theoretically, regular support vector regression can also be used for online optimization. Applications that are not time critical can integrate new samples by building a new model with the complete data set. In this way, dynamic effects can be taken into account with the classical approach. But if new samples are arriving very fast the computation time for model building is too long. This is where online SVR can be used. It offers incremental and decremental learning. This chapter will give an introduction to online optimization. Afterwards, the datasets used for

the optimization are presented in Section 6. Section 7 will introduce three online algorithms and show their individual performance. After a solution for online parameter optimization is presented in Section 8 the performance of the three presented algorithms are compared in Section 9. Parts of these chapters have been taken from [JBBR18].

5.1. Definition

Before the main requirements and the methods are presented, a central question has to be answered first.

- What exactly is online optimization?

There are several definitions of online optimization available:

- "Online learning is the process of answering a sequence of questions given (maybe partial) knowledge of the correct answers to previous questions and possibly additional available information."[SS12]
- "... an algorithm runs online if it makes a decision (computes a partial solution) whenever a new piece of data requests an action."[GKR$^+$01]
- "An on-line algorithm is one that receives a sequence of requests and performs an immediate action in response to each request."[Kar92]

All of those definitions have in common that there needs to be an immediate response on arrival of a question / request. The information about the past is very limited. In the case of hot rolling Online optimization procedures will continuously receive data and try to optimize one or multiple objectives with limited or without any information about the past. Let $\boldsymbol{y}_t = (y_{1t}, \ldots, y_{mt})^T$ define the m objectives at time t and $\hat{\boldsymbol{y}}_t$ defines the predictions of the online algorithm at time t. Then the optimization procedure tries to minimize residuals:

$$\boldsymbol{e}_t = \boldsymbol{y}_t - \hat{\boldsymbol{y}}_t. \tag{5.1}$$

Additionally, the procedure incorporates new samples and updates its prediction for the future. For the prediction of future values it may or may not incorporate additional input data $\boldsymbol{x}_t = (x_{1t}, x_{2t}, \cdots x_{nt})^T$ received at the same time t. It will be further assumed that only a limited amount of data from the past can be stored and used for prediction.
Since some aspects may be common also to other optimization procedures, another important question is:

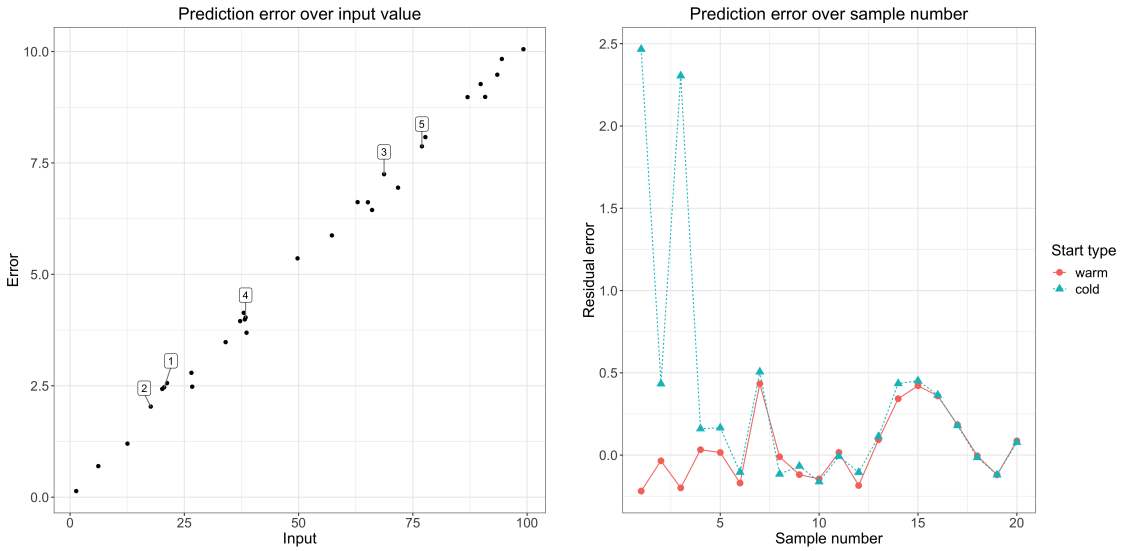
- What exactly is the difference to classical (offline) optimization?

In contrast to traditional (offline) optimization, no or only few samples are already known and may be used for initialization of the algorithm.
Another important difference to classical optimization is the update procedure. New data will be incorporated in an incremental way. For classical optimization, usually the whole dataset has to be used to update the predictions.

5.2. Initialization Problems

The potential problem, which might occur if no data has been retrieved before, is called *cold-start* problem. Otherwise it will be called a *warm-start* problem [GG01, Gon98, MTP03].
The fact that data may or may not be available does not mean that no information about the samples is known prior to training. Information about the number of dependable variables, type of variables and their usual range is assumed to be known. This assumption can be made because process know-how is mandatory for the task of optimization in the industrial environment. The previous assumption allows furthermore to scale the data prior to training. Scaling is used in most common algorithms like artificial neural networks (ANN) or kernel methods. Scaling refers to a linear transformation of data. After scaling, the data will have pre-defined upper and lower boundaries. Additionally to scaling, also normalization is used in many offline algorithms. Since no information about the variable distribution is known, normalization cannot easily be used in online algorithms.
When the performance of cold-start and warm-start problems are compared there will be a significant difference at the beginning. The performance of cold-start problems are inherently very poor at the beginning and will increase significantly after some data have been processed.

Example 1. To illustrate this behavior, a one-dimensional test data problem is analyzed. The selected model for this case is a simple linear regression model. Let us assume the error is linear correlated as shown in Figure 5.1(a). Here, the error is shown over the dependable variable x. The first five samples are labeled separately. For sake of simplicity the regression formula $y = 0.15 + 0.1 \cdot x + \epsilon$ has been used while the noise term is uniform randomly generated within an interval of $[-0.4, 0.4]$. Figure 5.1(b) shows the residual error over the sample numbers. This error is used to train the online algorithm. A regular linear regression model without any outlier detection or learning rate was used. The warm-start model was fed in advance with the first ten training samples which were randomly generated. The error in case of a *cold-start* for the first sample has exactly the same value as the first labeled value shown in Figure 5.1(a). The second error is then the difference between the first value and the second value. Afterwards the



(a) Test data

(b) Prediction error

Figure 5.1.: Comparison of residuals between warm-start and cold-start. The one dimensional test data was generated according to $y = 0.15 + 0.1 \cdot x + \epsilon$. The y-axis value in Figure 5.1(a) represents the target value which should be predicted by the online algorithm, while the x-axis value represents the value of the dependent variable. The values were randomly generated and the first five data points are labeled separately. Figure 5.1(b) shows the prediction errors over the sample numbers. The red curve shows the warm-start online algorithm, which was trained already using the first 10 samples. The blue curve represents the residual error term when no training was applied before (cold-start). The online algorithm was a linear regression model in both cases.

residual prediction errors results from the linear regression model of the processed samples. This simple example nicely illustrates the effect. After five samples have been received, the difference between *warm-start* and *cold-start* is almost negligible. The point or sample number where the difference between cold-start and warm-start gets marginal is highly dependent on the number of features in the dataset, the number of categorical variables and the complexity of the dataset. Regularly, it will take much longer than in this case.

Detection of outliers is almost impossible during the initialization of the algorithm and thus the model is very sensitive to them at the beginning. Inherently, the performance will also be very sensitive to possible outliers at the start. To prevent this, different strategies can be implemented. A simple strategy would be to use only a scaled version of the gradient error term for learning. Learning rates are also quite common in stochastic gradient descent (SGD) [Zei12] where the step size is influenced by the learning rate or neural networks [ST17].

5.3. Demands and Properties

There are different types of algorithms which may be applicable for real-world online optimization. The industrial application that generates the data stream will define the most relevant criteria for the selection of the specific algorithm. There are large varieties of industrial applications where online algorithms may be used. Different kinds of streaming data, like weather data, stock data, energy costs may be used for forecasting or general prediction. Also in the field of hot and cold rolling, many different data streams occur. Some criteria typical to the field of hot rolling and some assumptions are further described in this section.

5.3.1. Amount of Data

Algorithms used for the task of online optimization should assume to have an endless data stream. One aspect of this is, that algorithms may use already received data which are stored in memory and on hard disk. The total available memory is presumed to be limited.
The number of features is also assumed to be limited. Features are specifying here the number of different dependent variables. If y_t is defining our target variable at time t and $\mathbf{x}_t = (x_1(t), x_2(t), \cdots, x_n(t))$ defines the input data at time t, then n corresponds to the number of features. This limitation of n has to be presumed because a higher number of features would also require more memory. How much

more memory has to be allocated with a higher number of features also depends on the algorithm type. Some algorithms will store received samples or intermediate calculation results like kernel matrices. Some algorithms will only store coefficients for each feature. Nowadays, the algorithm used for hot rolling should not allocate more than approximately 100 MB. With a restriction to this amount it can be assured that at least the data can easily be kept in memory. The reason for such a small amount of memory is due to the fact that additional memory allocations are needed for the process where these algorithms are embedded.
Storing data on hard disk has to be avoided since it will inherently increase computation time because accessing data in memory is much faster. To have a fair comparison every algorithm is allowed to allocate the same amount of memory. Storage size refers to the internal memory allocated by the stored samples.
The storage size will define also the possibility of the algorithm to react to changes. If a large amount of samples is used by the algorithm, the performance is usually very robust and will adapt to current changes very slowly. Vice versa, within a small storage each sample gets a higher weight. The adaption to current changes is performed very fast with the drawback of lower robustness.
Once the amount of data which the algorithm is allowed to allocate hast been determined the question is which samples should be kept in memory. If the algorithm requires access to already received samples, a strategy for the management of these samples has to be evolved. For some applications this selection is straight forward.

Example 2. Assume a manufacturing process which is always producing the same product and all dependable variables are assumed to be similar, i.e. $\mathbf{x}_t \approx \mathbf{x}_{t+1}$. Further denote the actual predictions for the process as $y_t = f(\mathbf{x}_t, \mathbf{x}_{t-1}, \cdots, \mathbf{x}_{t-s+1}) + \epsilon$, where ϵ represents some random noise and s defines the number of samples which are stored. Then the quality of the prediction is mainly influenced by the current settings at time t and the actual process state. The dependency or correlation to the past production is not given. Therefore it is favorable to store only the latest samples.

For many other processes, like the highly flexible rolling process, this is not true. For special materials, which are only rolled on rare occasions, the observations of the actual behavior is strongly correlated with the last time this material was rolled. On conventional hot strip mills, a change of material may occur in every second or third product. Surely, the performance will also depend on the current process state, so a mixture between both data would be beneficial. The management, which defines the samples that should be stored, is therefore highly correlated to the concrete real-world problem and its complexity.

5.3.2. Arrival of Data

Another important question is how fast the samples are arriving. This is the most limiting requirement for the selection of the algorithm type [Kru02].
Typical applications in hot rolling are using TCP telegrams for the communication and have cycle times of some milliseconds up to some minutes. Cycle time is defined as the average distance between the arrival of two consecutive telegrams. Cycle times of more than one minute typically occur when the optimization problem should consider only data on slab-to-slab basis. An example of such a scenario is the width prediction in finishing mills or the correction of the so-called zero point.
Although the data may be received very fast, the necessity for an algorithm update is not always given. During rolling of a single pass the measurements collected are usually arriving very fast, i.e. in some milliseconds. But usually, the variance during this pass is handled by real-time control mechanisms. Those controls are far more capable to react in real-time but are not designed for complex calculations. Online algorithms are usually applicable if no real-time requirement is given. Preprocessing of those measurements and the calculation of averages over a certain time period is usually done for the hot rolling problems. Afterwards, a statistical summary will be used to update the online algorithm or the data is processed as a batch. This leads to the next criteria for algorithm selection: The number of samples which is used to update the algorithm.

5.3.3. Batch Versus Sequential

If the algorithm is receiving streaming data, the data may occur sequentially or it may always be a block of data which is received. There are special algorithms which are using the whole chunk of data (batch) to update their prediction and some are using every single data sample for updating. An example of batch learning algorithms is presented e.g. in [LZHM12] where the authors are using a principal component regression on batches for iterative updates. In principle, all algorithms which are able to be updated with a single sample can also be updated with the whole batch. Then, all individual samples of the batch are fed to the model. Algorithms, which are using batches have to collect samples before updating. They cannot be updated with every received sample.

5.3.4. Source of Data

On the first glimpse it does not seem to be important where the data is coming from. But there is a huge difference, if the underlying process is just periodically sending measurements which will be processed, or if some event is causing the processing of data, which will be referred to as event driven data processing. Periodically measurements are for examples temperature measurements or energy consumption. This is defined as time driven data processing [Mis18]. The rolling process is an example for the event driven processing of measurements. Only if products are rolled the algorithms will be fed with data. When monitoring the arrival time only, it will be impossible to distinguish between event based or periodically triggering.
If there is a continuous measurement and periodically some data will be collected and sent to the online optimization algorithm, then frequency, amount of data and other behavior of the stream is assumed to be always the same during normal production. For some time, it may look similar to the collection of continuous measurements as in the case of weather data. At some point, there will be a disruption in this data stream. For real-world scenarios this is often the case due to some maintenance or scheduled downtime where parts of the production equipment is not available. If this time is known in advance the algorithm may perform some more cost-intensive optimization which would not be easily possible during the continuous production. It may also be achieved during the production if the cost-intensive optimization can be outsourced to other machines. Then again, the time for takeover of the improved optimization has carefully be selected. For rolling mills, there are always some scheduled break-down events. This can be e.g. some roll change or some regular scheduled maintenance.

5.3.5. Pre-Processing

The pre-processing of data can be realized in several ways and can get very demanding. It can and should be installed on the sender side or on the receiver side of the data. The general process of filtering and processing for the hot rolling application is almost identical for all scenarios discussed throughout this thesis. The description of the process is illustrated in Figure 5.2 and comprehends the following steps.

1. Calculation of Setup

First the automation system calculates settings based on the information about the initial product and the target product. With these information, the schedule

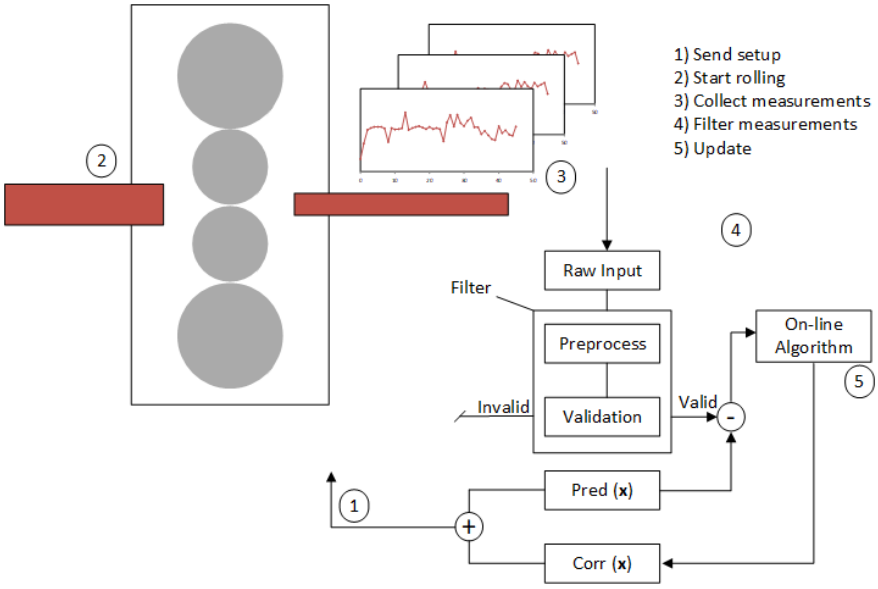


Figure 5.2.: One cycle for updating the online algorithm. First, the analytical process models are calculating and predicting the process parameters (1) based on the information ($\boldsymbol{x}$) on the product and the corrections received from the online algorithm. They are used to start the rolling process (2). As soon as the rolling starts all measurements are collected and sent to the process system (3). There, the raw values are pre-processed and validated (4). If they are valid the prediction error is calculated and fed to the online algorithm which will update the subsequent predictions (5).

how to roll this product is summarized in the pass schedule. It will contain every process step. For each of these process steps, the setup is send to the mill and the basic automation system (1). This may already include a correction term from the online algorithm.

2. Rolling
With the settings and predictions from the process models, the rolling of the next pass will start (2). During rolling, the operator or the basic automation system may adapt settings in case of unexpected situations.

3. Collecting Measurements
During rolling all measurements are send to the measurement pre-processing system where they are collected (3). The most relevant measurements are roll force, roll torque, thickness, width and temperatures.

4. Filtering and Validation
From all of these measurements outliers will be removed by simple statistical filters. One example would be to a apply a filter based on the mean and the standard deviation. All measurements which are not within the range of $\mu \pm 2\sigma$ are excluded directly (4). Here, μ defines the mean of the process variable and σ the standard deviation. The predictions for all parameters in Step 1 are only done on several discrete positions of the product, i.e. three or five discrete length positions. After removal of outlier, a statistical summary for these positions will be made. The mean, minimum, maximum and standard deviation are calculated individually for each position. Therefore, only measurements which are close to these length positions are taken into consideration. If, e.g., a length position of 10m was calculated, then only measurements for 10m $\pm$ 2m are taken into consideration.
These values are used by the process models to decide whether the measurements are valid and can be used for optimization. If they are within a pre-defined range, the average value will be used for calculation of the prediction error which is sent to the online algorithm.

5. Update
If necessary, the data will be scaled to defined upper and lower boundaries before the online algorithm will update the corrections (5) and a new cycle may start. In general, a filter can also be applied on the receiver side of the online algorithm. But then the online algorithm should know some details about the process to be

able to detect outliers.

Another difficulty is the selection of variables which are used for regression. This is a similar and well known problem also for offline algorithms. An introduction to variable selection is, e.g., given in [GE03]. An additional problem for variable selection in online algorithms is the time dependency. If a variable shows a time dependent behavior it is usually not known in advance and is especially not known for cold-start problems.
Sometimes process experts know, that a certain variable has an influence on the prediction performance but the level of significance is not known. So process knowledge is used for variable selection. After some samples have been received the decision can be re-evaluated based on these samples.

5.3.6. Data Types

Data streams may contain continuous and categorical variables. A categorical variable usually only has a limited number of possible values. These values may be numeric but can also contain arbitrary characters. The different values are called levels throughout this thesis to distinguish them from the continuous, numeric values. Categorical variables are also called discrete variables or qualitative variables [Agr07]. If the number of levels is two than it is a dichotomous variable. If it has more than two levels the categorical variable is called polychotomous [Enc08].
For hot rolling, all categorical variables are assumed to by polychotomous. Examples of categorical variables for hot rolling are alloy code, coiler number, furnace number or time information like the working day or the season.
If there is an order within the different levels of the categorical variable, the categorical is also called ordinal variable. This is usually the case for variables which are the result of a grouping based on numerical values. Consider e.g. the categorical variable size of a person with values small, medium and tall where the grouping is done according to the absolute size. Surely, the categorical can be seen as ordinal. Basically the same is often done for hot rolling to build groups for width, thickness or temperature. Although the categorical itself may be ordinal it does not mean the effect on the optimization problem has the same relationship. It may be, that wide and narrow products have the same effect on the problem but medium products have a different effect. If no direct relationship between the different levels of a categorical variable can be made, the categorical may be converted to an ordinal variable. But this is not always possible and desirable.
There are numerous methods available to handle categorical variables [PPP17, CCWA03]. The simplest one is dummy coding, where each level of a categorical

Table 5.1.: Contrast coding and dummy coding for categorical variables for the categorical variable Furnace with three levels (F1,F2,F3). Two columns (C1,C2) have to be added to the dataset to represent this categorical variable.

Coding scheme	Furnace	C1	C2
Contrast	F1	-1	-1
	F2	1	0
	F3	0	1
Dummy	F1	0	0
	F2	1	0
	F3	0	1

variable is added as a new feature to the dataset and is coded with 1 if this level is present or 0 if it's not. In total for c numbers of different levels $c-1$ additional variables are added to the dataset.
An alternative coding scheme is contrast coding. In contrast coding each level of the categorical is assigned a specific weight and the sum of the weights for each category has to be 0. The numerical weight expresses the relationship between the different levels, i.e. the mean value for this group. Usually the coding scheme is applied in such a way that a chosen referential level will be assigned a value of -1. But also other codings are possible. A coding example for a rolling mill with three different furnaces $F1, F2, F3$ and two possible coding schemes are given in Table 5.1. Note that only two variables, i.e. $C1$ and $C2$, have to be added and F1 is the reference furnace. The regression coefficient for C1 in case of dummy coding can be interpreted as the effect of F2 compared to F1. In case of contrast coding C1 expresses the difference of F2 and F1 to the overall mean. There a also other coding schemes which can be used. For the case of ordinal variables special coding schemes like polynomial coding or Helmert coding can be applied [Wen04].
If there is an interaction between the categorical variables and other variables the previously described coding schemes alone will not be suitable to improve the predictions. Interaction terms have to be added to the regression formula. This has to be considered during the design of the algorithm, so a profound understanding about the process is needed.
For example in the case of roll force optimization in the hot rolling process the categorical *Furnace* is considered to be independent of all other dependent variables. But the categorical variable, which is used for the alloy specification, has usually a strong correlation to other parameters. For one alloy there might be a linear correlation between the error of the roll force calculation and the temperature where another material has a descriptional error in dependence of the deformation speed. The reason for such differences can be found in the analytical model which

uses different parameter settings to predict the material behavior. Therefore, interaction terms of alloy specification and the parameters which are describing the flow curve, i.e. deformation, deformation speed and temperature, have to be added to the dataset.
A major drawback of the these coding schemes is their extensive memory requirement. For offline optimization, because of the features selection and dimensionality reduction possibility, it may be possible to use this kind of coding schemes. For online optimization where the number of different levels of such categorical variables may be very high, this cannot be applied.
Another problem is the lack of knowledge about the different levels of the categories. Materials not yet observed by the algorithm might be rolled and would introduce a new level of this categorical. This cannot be handled by most common algorithms properly.
Instead of adding a new feature to the data, a new instance of the algorithm can be used on each different level of such categories. A new instance refers to a new, independent model which only receives data, if a specific combination of all categorical variables are occurring in the data stream. The memory requirements will also be very high but there is no need to have all different versions in memory. The process will know in advance which furnace will be used for the next product and which material is going to be rolled. These information are used to load the specific algorithm version prior to rolling. A problem with this approach is that current process trends which are independent from the categorical variable will not be recognized by each model instance. This will make drift detection more complicated. The scenario is depicted in the upper part of Figure 5.3.

Another possibility is to use a baseline model and only use a specific correction model for each different occurrence of the categorical variables. The baseline model will consider each drift and the specific models are only compensating the difference to this baseline. Somehow the idea is similar as the dummy coding scheme. But instead all numerical variables are also used in the correction models. This is depicted in the lower part of Figure 5.3. A more detailed description of this strategy is given in the corresponding algorithm section.
Summarizing, the following different treatments of categorical variables will be analyzed.

Remove: Removing the categorical variable. The model will then be trained only with the numerical values.

Numerical: Each level of the categorical variable will be represented by a numerical value.

Parallel 1: Creating a model instance for each combination of levels.

Parallel 2: Creating a correction model instance for each combination of levels and use a baseline instance.

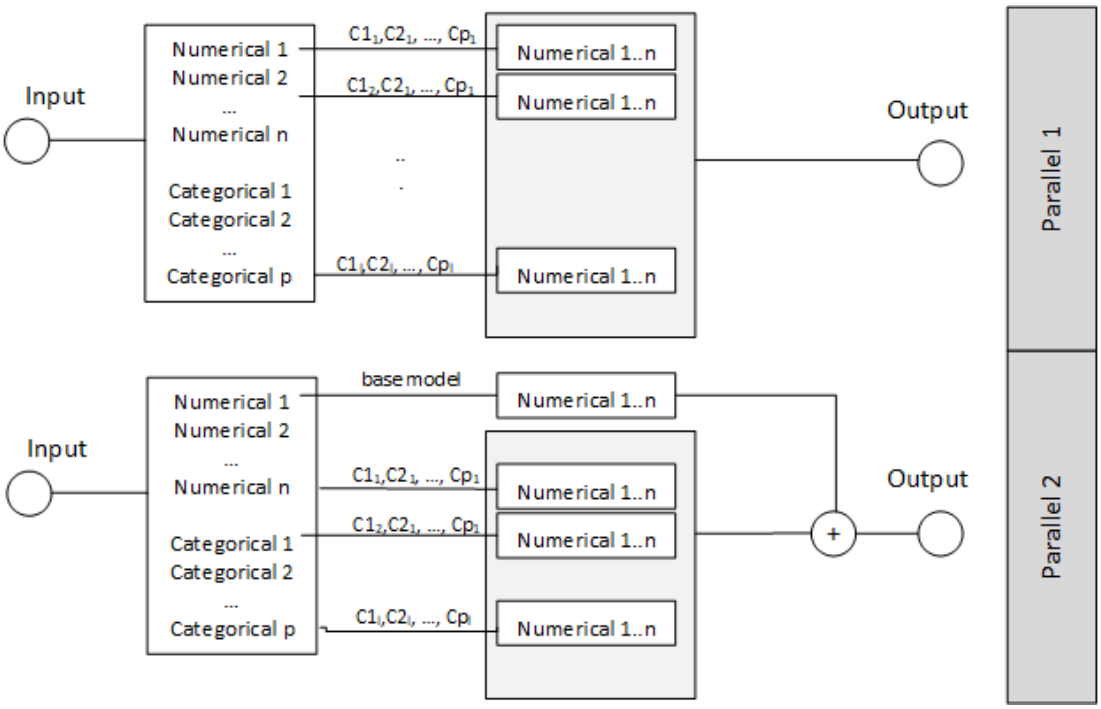


Figure 5.3.: Two variants for handling categorical variables. In the scenario depicted at the top, for each unique combination of the different levels of each categorical variable $C_1 \cdots C_p$ with number of levels l, an own model instance will be used. This specific model is then trained and used for prediction with all n numerical variables. In the lower part, a baseline model is trained and only the deviation to the baseline is handled by the specific model instances for prediction and training.

For comparisons and also for offline algorithms however, there are additional possibilities.

Dummy 1: Regular dummy or contrast coding.
Dummy 2: Dummy coding including all interaction terms.

5.4. Performance Indicator

If the main interest is not the exact value of the total performance but the dynamic behavior, then MSE or MAE described in Section 3 are not always suitable. If the dataset contains a large number of samples then the deviations at the end of the dataset will have no huge impact on the performance. Instead, a windowed version of the MSE and MAE should be used.
For offline algorithms, the data is usually split into three different sets [RH95]. The training set is used for training of the model, i.e. the initialization. The validation set which is used to compare different models and to validate the performance. And the test set which should confirm the measured generalization performance and is used because of the possibility of over fitting. The training set is usually a randomly chosen subset of the whole data set and covers a much smaller size than the test set. The size of the set also depends on the model type and the total amount of data available.
For validation and comparison of online algorithm performance this kind of approach is inappropriate. The main field of application for online algorithms are dynamically changing optimization problems. The selection of randomly chosen data is not reflecting the real process behavior and cannot be used for performance evaluations. Only if the process is stable without any kind of change in data the same procedure as for offline algorithms can be chosen. But then the usage of online algorithms lose their advantage of being able to adapt to changes.
For a fair comparison of online and offline algorithms the offline algorithms will be trained on consecutive data only. Otherwise the time dependency of the features would be averaged out of the data.
For a real-world application the online algorithms are used for both, warm-start and cold-start optimization problems. The ability to quickly react on observed errors is also a quality criteria which should be used for proper model selection. Ideally, the algorithm can react very fast to dynamic changes in the data whereby being robust in its predictions.

5.5. State of the Art

Modern hot rolling mills will not achieve a high performance without any kind of online algorithms. Different strategies have been evolved to achieve best possible solutions. Several years ago, where data storage was very expensive, the data collected during rolling was not stored for more than 4-6 weeks. Databases were also not widely used and statistical analysis was hardly applicable.
Process experts discovered inaccuracies and were usually able to define critical products, i.e. products where the prediction shows inaccuracies or high deviations. The usual method to compensate these inaccuracies was to define groups of products where these errors occur. Continuous variables like width and thickness were subdivided into groups together with an alloy identifier and other classification variables [FF09].
The usual approach was now to store a sliding average for the n-dimensional array and to correct the prediction based on the observed errors. The idea of this approach was driven by multiple reasons. On the one hand side, the complexity of this approach is very low and correction terms can easily be calculated by hand. On the other hand side, there was a risk to influence other regions where no similar prediction inaccuracies could have been observed.
Since cost for data storage, the processing power of the machines and the automation systems became more complex and capable, databases were used to store more and more process related data. This enabled the usage of sophisticated statistical analysis. Consequently, online algorithms were also getting smarter and more complex. Analysis of certain behavior over several weeks or even month are now possible and parameters explaining the prediction variance can be detected more easily. The usage of more complex algorithm like artificial neural networks (ANN), Bayesian networks [LRT01] and regression algorithms [ÖÖ00, AY01] were implemented. ANN was and still is a black box algorithm for most people and it never was used on a very broad basis. However, ANN is applied to several optimization problems in the hot rolling area but is most often used offline to improve the general prediction [DSP+19, SK08]. Sometimes they are also used as so-called long term adaption which is characterized by a slow learning rate [FF09]. Long term adaptions have the task to correct almost constant prediction errors. Common algorithms for long term adaption are moving averages and recursive regression but also other may be used. Among the other algorithms, ANN is the most popular algorithm.The easiest approaches for the moving averages are given in the following. With the residual $e_k = y_k - \hat{y}_k$ at time instance k, $\hat{e}_{k+1}$ as prediction error with corresponding weights w, $\sum_{i=0}^{n-1} w_i = 1$ and with window size n

we get:

$$
\begin{aligned}
\textbf{Moving average:} \quad \hat{e}_{k+1} &= \tfrac{e_k + e_{k-1} + \cdots + e_{k-n+1}}{n} \\
&= \tfrac{1}{n} \sum_{i=0}^{n-1} e_{k-i}. \\
\textbf{Weighted moving average:} \quad \hat{e}_{k+1} &= \tfrac{w_k \cdot e_k + w_{k-1} \cdot e_{k-1} + \cdots + w_{k-n+1} \cdot e_{k-n+1}}{n} \\
&= \tfrac{1}{n} \sum_{i=0}^{n-1} w_{k-i} e_{k-i}. \\
\textbf{Exponential moving average:} \quad \hat{e}_{k+1} &= \begin{cases} e_k, & k = 1 \\ w_k \cdot e_k + (1 - w_k)\hat{e}_k, & k > 1 \end{cases}.
\end{aligned}
$$

The window size is defining the number of samples which are taken into account. The proper choice of weights is usually done empirically by process experts. Beside the correction of long-term trends and effects, there is a strong need to quickly react on actual deviations. This cannot be guaranteed with most of the long term approaches alone. Additional algorithms are implemented and so the usual approach is to have another class of adaptation algorithms. Consequently it is called short term adaption because it will react instantaneously to actual deviation. Usually a simple moving average algorithm will be used. In comparison to the long term adaption, the number of samples n, which are used for calculation, is much lower. As a consequence the average value will react much faster on higher deviations. The combination of both adaptation classes is usually additive.

6. Datasets for Online Optimization

The concept and methods discussed and presented in this chapter are applicable for most real-world optimization problems. Although the special focus is on the complex rolling process, the methods are also tested for other datasets which are publically accessible. There are three main resources which are used. The Delve datasets[1] hosted by the University of Toronto, the UCI datasets[2] with more than 400 different datasets and the StatLib datasets[3]. In the following, first a brief description of the default datasets is given. Afterwards the datasets used in the field of hot rolling are described.

6.1. Default Datasets

In order to compare the implemented algorithm and validate the implementations the algorithms are compared to other real-world datasets which are publicly accessible. Ikonomovska [Iko12] compared different algorithms, mainly regression trees, on a large variety of real-world regression datasets which can ideally be used as comparison to our implementations. The datasets are available on her homepage[4] or from the original source. Most of the examples are taken from the UCI database, the StatLib database or the Delve database. Table 9.1 summarizes the reference datasets used throughout this thesis. Clearly the size of these datasets is significantly smaller than in usual online applications. The datasets are also used to discuss the different parameter effects of the corresponding algorithm. The datasets Abalone and Mv delve are also used to show the impact on the different treatment of categorical variables. The usage of these default datasets should also emphasize the wide application range of the online algorithms.

[1] http://www.cs.toronto.edu/ delve/data/datasets.html
[2] https://archive.ics.uci.edu/ml/datasets.html
[3] http://lib.stat.cmu.edu/datasets/
[4] http://kt.ijs.si/elena_ikonomovska/

Table 6.1.: Overview of samples and data types in the default datasets

Problem	Samples	Numerical	Nominal
Abalone	4 977	7	1
Cal housing	20 500	8	0
Elevators	16 599	18	0
House 8L	22 784	8	0
House 16H	22 784	16	0
Mv delve	40 967	7	3
Pol	15 600	48	0

Table 6.2.: Number of samples and variable types of the rolling datasets.

Problem	Samples	Numerical	Nominal	Mill Type	Material
Width	93 935	34	1	Roughing Mill	Steel
PlateForce	21 840	8	1	Plate Mill	Steel
StripForce	25 721	9	2	Hot Strip Mill	Aluminum

6.2. Rolling Datasets

There are three rolling datasets used for testing various requirements scenarios. A short description for each dataset is given in the following section. Table 6.2 is giving a summary of the used rolling datasets.

For all rolling datasets, the task is to compensate the difference between a measurement from the real process and a prediction made by an analytical model. The target value for the online algorithm is then the residuals e calculated as the difference between measurement y and prediction $\hat{y}$. The predictions made by the analytical model and the measurement will not be included in the dataset since the focus will not be to replace the analytical model, but to improve the residuals.

Width Dataset

Width control and prediction is very complex and has multiple dependencies. The Width dataset contains approximately 94 000 samples of a steel roughing mill. The target variable is the width prediction error during rolling. An analytical model M_W, briefly introduced in Section 2.3 was used for the prediction of the width. Basically a slightly modified version of the Nippon Kokan [TOY$^+$82, THW$^+$83, OANY80] model was used. The modification was made to optimize the predictions for high width reductions and to distinguish between advanced steel grades. This

model was already tuned so that only the residuals should be compensated online. The width measurement is taken at the exit side of the mill with an installed width gauge. The residuals e are therefore the difference between the prediction $\hat{y}$ of the analytical model and the measured width y of the installed width gauge. The dataset consists of 34 numerical features which are describing the geometrical deformations, temperatures, roll state and the material. Additionally, the furnace number is included as a categorical variable. For the material description 21 of the most important chemical elements are used.
The dataset is well suited for the analysis of dynamic behavior and drift. A common issue in roughing mills is that the roll diameter of the edger rolls are measured at the wrong position. Edger rolls usually have a conical shape to apply a certain force towards the roller table. The diameter at the top is therefore usually a bit larger than the bottom diameter. When the roll diameter is measured at the wrong position, the automation system will have a bias in the roll gap calculation and the expected width reduction will not be achieved. This phenomenon is visible in this dataset, approximately at samples number 40 000. Here, either the assumed diameter or the conical shape of the rolls are different to the actual used rolls. After roll change, the width reduction was higher than expected which caused the measured width after the mill to be more narrow than expected. Somehow the behavior changed over the next samples and the initial accuracy was reached after approximately 20 000 samples. The residuals represent only the deviation of the analytical model. During the collection of the process data, an online model was used to compensate this deviation.
The categorical variable "furnace" cannot be used as ordinal variable since each furnace has an own predictive model associated to. The number of furnaces is known a priori and can therefore also be used in the traditional dummy coding schemes. Nevertheless also the performance when converting this variable to a numerical value is shown.
To get an idea about the target value distribution Figure 6.1 shows the target value and the trend of the target value. It can be seen that there is a sudden drop of the target value and afterwards a trend towards zero. Zero in this case means, that the width prediction of the analytical model and the measured width are exactly the same. The main goal for the online optimization is to reduce the variance in this prediction. The standard deviation of the analytical model is approximately $4.5mm$. A standard deviation of below $3mm$ would be ideal.
Throughout this thesis this dataset will be referred to as Width.

PlateForce Dataset

The PlateForce dataset contains roll force errors for a heavy plate steel mill. It will be referenced as PlateForce throughout this thesis.

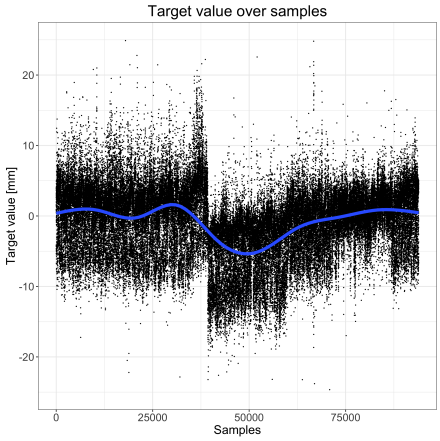


Figure 6.1.: Scatter plot of the target value. The blue solid line represents the average value.

The target value represents the predicted roll force error e. The errors are collected as the difference of measured roll force y and the predicted roll force $\hat{y}$ in the body part of the plate, i.e. $e = y - \hat{y}$ The body part has the big advantage that the force is usually very stable since the temperature in the body part is also very stable. On both ends of the plates the temperature losses are higher and therefore the forces may show larger differences than in the body part. For each plate an average value of the body will be calculated for each pass. The predicted values are calculated by an analytical model M_F which was briefly described in Section 4.3.
The measured values were pre-processed and filtered so it is assumed that no measurement errors are present in the data. The dataset has eight numerical features and one categorical. The categorical variable represents the material description which is a mapping of the alloy against some target values. There are 12 different levels of the categorical variable within the whole dataset. Since the data is only an extract of the production, it might be possible that new materials are rolled. The only way of handling this categorical variable is therefore to remove it, convert it to a numerical value or add parallel instances of the algorithms. The numerical values represent the description of the deformation, i.e., geometrical attributes, temperatures and resistance values. A total number of 21 840 samples are collected. More than 90 % of the examples contain measurement for one dominating material. The flow curves are validated for those rolled materials. The data can therefore be seen as the logical next step in improving roll force prediction after a flow curve has been found.
Figure 6.2 shows boxplots for the target value, divided into groups according to the rolled material. It can be seen that the target value has different deviations for

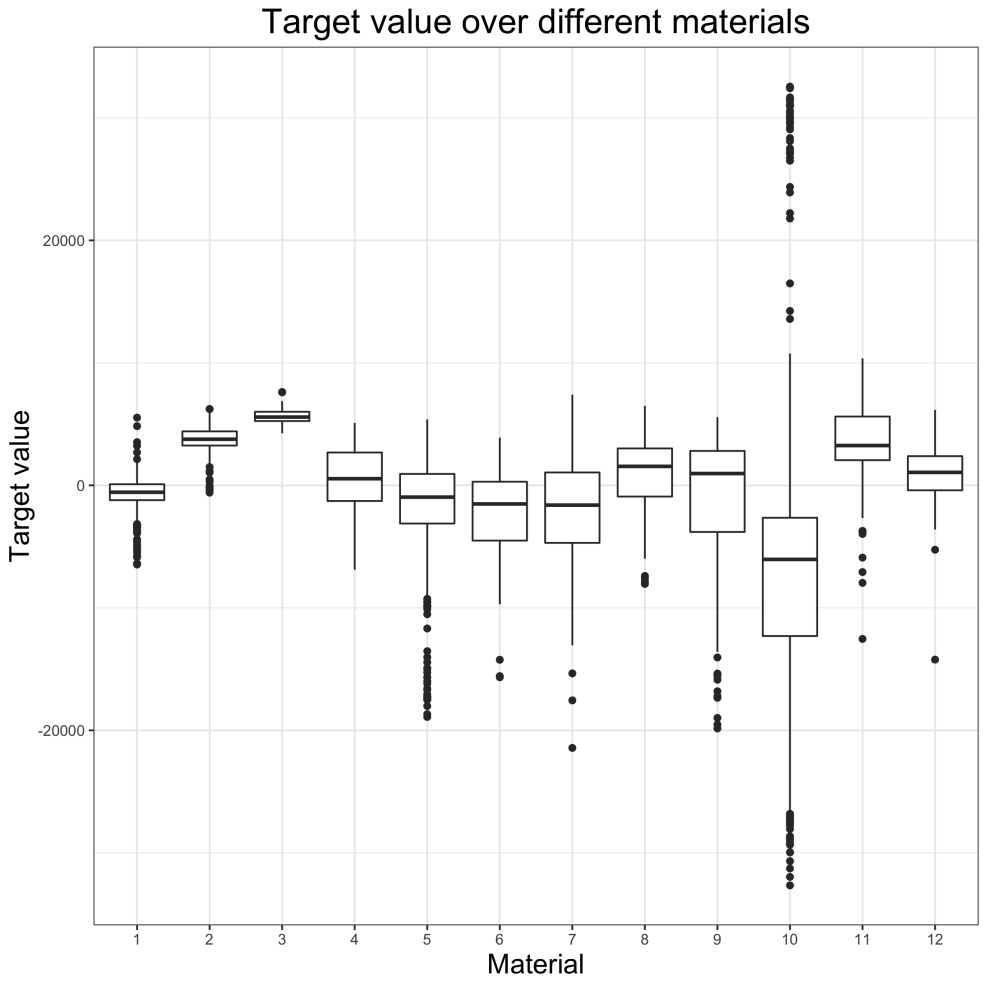


Figure 6.2.: Target value boxplots for different occurence of the material group for the PlateForce dataset. High deviations may indicate inaccurate description of the material.

the material groups. For some material groups, e.g. the first one, the prediction is already very good. The mean deviation is approximately zero and the standard deviation is very small. For other materials, a high mean deviation and much higher standard deviation can be observed. This should be corrected by the online algorithm. From Figure 6.2, it can already be seen that neglecting the material group will probably not be sufficient. The high deviation and large variance observed for material 10 indicates an inaccurate prediction. This material was used for thermo-mechanical rolling practice where material is first rolled to an intermediate thickness and afterwards cooled to a specific temperature. If this temperature is reached, the rolling continues. This procedure is well established and is used to achieve specific mechanical properties, especially for pipeline steels [WZZ16, TWW+18].

StripForce Dataset

The previous rolling datasets originate from a single stand mill. The StripForce dataset is coming from a continuous aluminum finishing mill with four stands. The target value is the roll force error e which is calculated as the difference between the measured value y and the predicted value $\hat{y}$ of the roll force model M_F. The measured value is taken this time from different length positions over the length for each stand. Usually five different length positions are taken into consideration. The measured value will be calculated as average value at the corresponding length coordinate. The predicted value is calculated by an analytical model which is similar to the one described in 4.3 but was modified and tuned for this type of mill. The target value does not correspond to the real error which was present during rolling, since other online algorithms were used to minimize this error.
The data contains nine numerical features and two categorical features. The categorical variables are the material group and the stand number. 24 different materials, from soft alloys like pure aluminum to some hard Mg-based aluminum alloys, have been rolled and are present in this extract. The numerical features are geometrical values, temperatures and the inter stand tensions. In total 25 721 samples are in this dataset.
In order to see the influence of the different material groups on the target value, Figure 6.3 shows a boxplot of the target value over all different material groups. The plot does not distinguish between the different mill stands, because the main influence is assumed to be correlated to the material.
Obviously, the distribution of the target value is strongly correlated to the material group and the removal of this categorical will probably not be beneficial. Such high deviations would cause severe problems to the rolling process. For the real process, simple online corrections have been applied to compensate these errors.

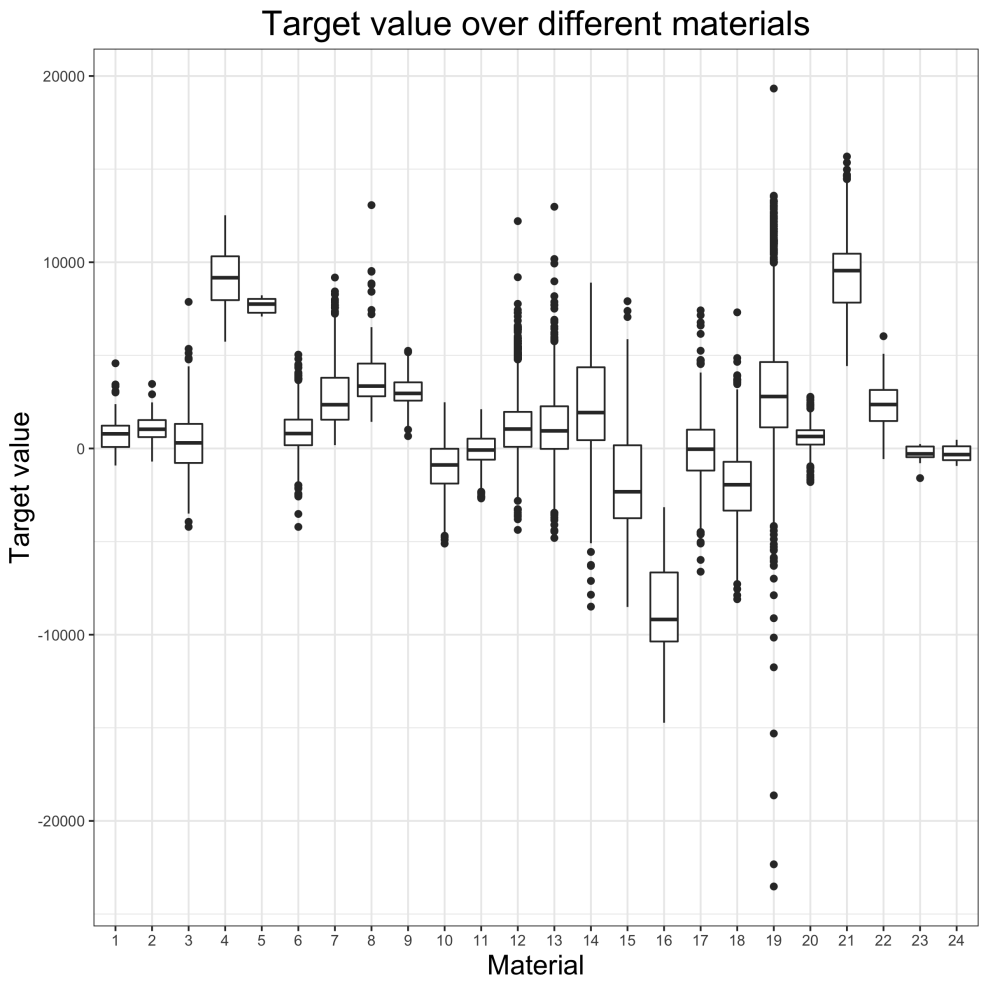


Figure 6.3.: Boxplot for the target value over each material group in the StripForce dataset. The target value is the difference between the measured roll force and the prediction made by the roll force model M_F. High target values indicate a poor performance of the prediction.

6.3. Requirements and Assumptions

After the general description of the demands and properties and the discussion of the different datasets some limitations have to be set for the development of new methods and algorithms. The following restrictions have to be considered for the application of the online algorithms that are introduced in Section 7.1, 7.2 and 7.3. Parts of this list are taken from [BHKP10].

- The dataset consists of a limited amount of columns / features.
- Datasets are received one at a time. They may be pre-processed and used for batch learning.
- The data consists of some factors with a limited number of levels. The datatype for all other data are floating point numbers.
- The data stream is endless.
- The number of features is fixed through the entire stream although there may be some new levels of known factors.
- Algorithm should use a limited amount of memory.
- New samples should be incorporated in a limited amount of time ($\leq$ 100ms).
- Prediction should be available at any time in a limited amount of time ($\leq$ 10ms) as long as no optimization is running. A parameter optimization may be conducted if there will be a scheduled maintenance. During maintenance, no process prediction is required.

During this thesis, there will not be distinguished between batch learning and learning of sequential samples one-by-one. If data are arriving in batches, these batches can be split into single samples and used for training.

7. Algorithms

The following chapter will describe three different online algorithms which were implemented and extended for the hot rolling process. First, a simple first order algorithm, Passive Aggressive will be introduced in Section 7.1. Afterwards the popular Recursive Least Squares algorithm is discussed in Section 7.2. This chapter ends with the discussion of the properties and the extensions made for the online Support Vector Regression in Section 7.3.

7.1. Passive Aggressive

The Passive Aggressive (PA) algorithm was developed by Crammer et al. [CDSSS03]. It is a simple first order online algorithm. First order is referring to algorithms that are taking the error directly into account without considering higher order losses. The algorithm was first developed for binary classification and was extended afterwards to the regression case.
Let $\boldsymbol{x}_t = (x_{1t}, x_{2t}, \cdots x_{nt})^T$ define the input data vector at time t of n features and y_t define the target value at time t. Further, let $\boldsymbol{w}_t = (w_{1t}, w_{2t}, \cdots, w_{nt})^T$ define the coefficient vector for those features at time t. Then the prediction of y_t is calculated using the dot product as:

$$\hat{y}_t = \boldsymbol{w}_t \cdot \boldsymbol{x}_t \tag{7.1}$$

After receiving the true target value y_t and calculating the absolut prediction error, the coefficients $\boldsymbol{w}_t$ are now updated according to the ϵ-insensitive loss function

$$l_\epsilon(\boldsymbol{w}, (\boldsymbol{x}_t, y_t)) = \begin{cases} 0 & |\mathbf{w} \cdot \mathbf{x}_t - y_t| \leq \epsilon \\ |\mathbf{w} \cdot \mathbf{x}_t - y_t| - \epsilon & \text{otherwise} \end{cases}$$

where $\epsilon \geq 0$ is a parameter which controls the sensitivity to any prediction error. The parameters $\boldsymbol{w}$ are updated using the solution of the optimization problem.

$$\mathbf{w}_{t+1} = \underset{\mathbf{w} \in \mathbf{R}^N}{\operatorname{argmin}} \frac{1}{2} ||\mathbf{w} - \mathbf{w}_t||^2 \text{ s.t. } l_\epsilon(\boldsymbol{w}, (\boldsymbol{x}_t, y_t)) = 0. \tag{7.2}$$

The detailed solution of this optimization problem is presented in [CDK+06] and yields:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \text{sign}\,(y_t - \hat{y}_t)\,\tau_t \boldsymbol{x}_t \tag{7.3}$$

with

$$\tau_t = \frac{l_t}{||\mathbf{x}_t||^2}. \tag{PA}$$

Each update of the coefficient vector $\boldsymbol{w}$ would now ensure, that the constraint held by the actual sample is satisfied. For real-world optimization problems with the presence of noise and following measurement inaccuracies, this update strategy is too aggressive and can lead to dramatic consequences. To avoid this problem, two variants of this algorithm were developed[CDK+06] by reformulation of the optimization problem:

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\mathbf{R}^N}{\text{argmin}} \frac{1}{2}||\mathbf{w} - \mathbf{w}_t||^2 + C\xi \tag{7.4}$$

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\mathbf{R}^N}{\text{argmin}} \frac{1}{2}||\mathbf{w} - \mathbf{w}_t||^2 + C\xi^2 \tag{7.5}$$

The introduced parameter C is the so-called aggressiveness parameter and defines how aggressive the updates should be. The variable $\xi, \xi \geq 0$ is a slack variable and controls the objective function, i.e. $l\,(\boldsymbol{w}, (\boldsymbol{x}_t, y_t)) \leq \xi$. The optimization problem in Equation 7.4 is called PA-I and 7.5 is called PA-II. The update of the coefficient vector will still be calculated using Equation 7.3, but parameter τ_t is now calculated according to:

$$\tau_t = \min\left\{C, \frac{l_t}{||\mathbf{x}_t||^2}\right\} \tag{PA-I}$$

$$\tau_t = \frac{l_t}{\frac{1}{2C} + ||\mathbf{x}_t||^2} \tag{PA-II}$$

From Equation PA-I it can be seen that all three variants will perform similar if the parameter C is chosen to be very high. For PA-I this will already be the case if $C > \frac{l_t}{||x_t||^2}$ whereas in PA-II the algorithm will continuously tend towards the PA algorithm with increasing C.
In the remainder of this section the different settings and variants of the PA algorithm will be compared and their effect on real-world scenarios are discussed.

7.1.1. Comparison of the Three Different Variants

To compare the performance of all three PA algorithm variants (PA, PA-I, PA-II) four of the default datasets were used to train the algorithm. Figure 7.1 shows the MSE over the whole datasets. For PA-I and PA-II the algorithm was used with various values for the parameter C, but only the algorithm which achieves the best performance over the whole dataset in terms of MSE is shown.

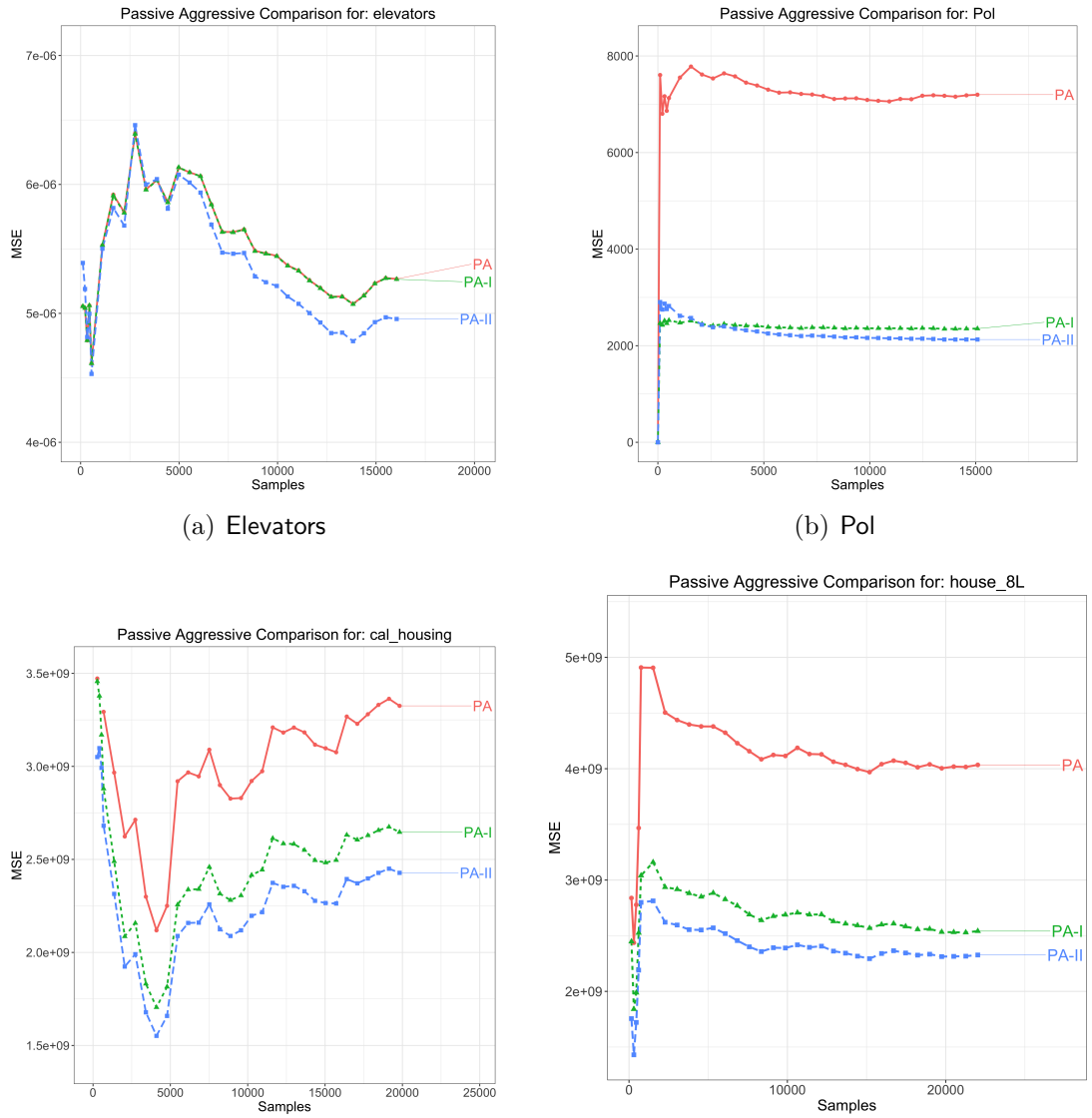
It can be seen from Figure 7.1 that in general the original algorithm (PA) has the worst performance. This is obvious because both variants must have at least the same performance if parameter C is chosen to be very high. The datasets *elevator*(Fig. 7.1(a)), Pol (Fig: 7.1(b)) and House 8L (7.1(d)) show no dynamic behavior and the prediction is almost constant over time. For all shown datasets the variant PA-II achieves the best performance, followed by PA-I and the original algorithm. The dataset Cal housing (Fig. 7.1(c)) shows a high dynamic behavior over the number of samples but also here, the PA-II is superior to the other variants.

7.1.2. Influence of Aggressiveness Parameter

The parameter C for variants PA-I and PA-II is of great importance and has major impact on the performance of the algorithms. To illustrate its impact, Figure 7.2 will show the prediction of two different parameter settings for each algorithm type on some artificially generated data. The data was generated according to:

$$y = \begin{cases} 2.6 + 0.4x + \epsilon, x \leq 5 \\ 2.6 + 0.2x + \epsilon, x > 5 \end{cases} , \tag{7.6}$$

where ϵ is the realization of a normally distributed random variable with mean zero and variance 0.1. The function in Equation 7.6 will further be referred to as *gap5*. Samples were fed to the algorithm in ascending order of x. Therefore the algorithm suddenly has to face a target value shift at $x = 5$. For the PA-I, parameter C was chosen to be 1 and 0.01. During initial learning and also at the sudden drop in data, the performance with parameter $C = 0.01$ was worse than in the case of a $C = 1$. This can easily be explained because the parameter is defining the maximum incremental update rate for learning of new behavior. The same behavior can be seen when looking at the performance of PA-II in Figure 7.2(b). The only difference is, that in comparison to PA-I, the initial learning is much faster even for the same Parameter $C = 0.01$. For PA-II, a small value of C will cause a small value of τ_t which is responsible for the update procedure. To analyze the effect on our datasets, both variances of the PA algorithm were trained with

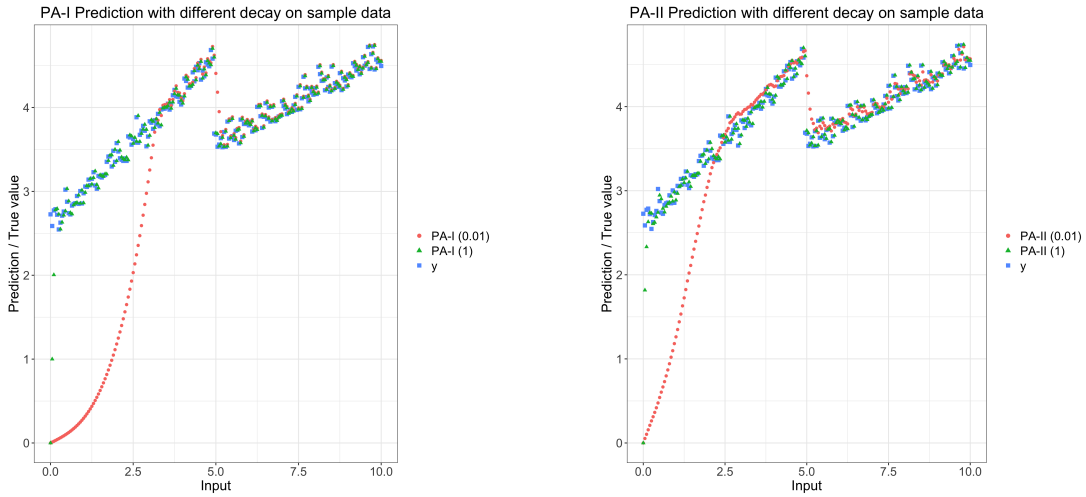


(a) Elevators (b) Pol

(c) Cal housing (d) House 8L

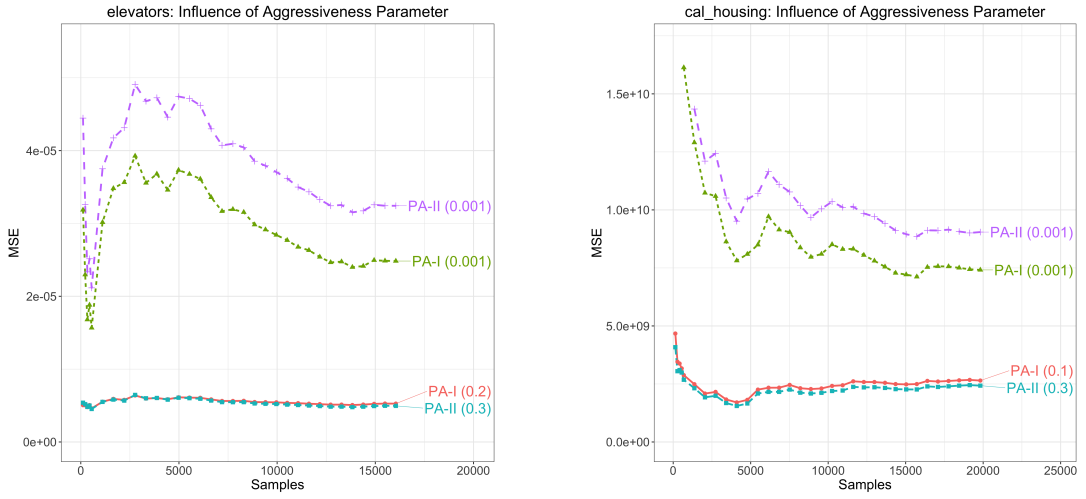
Figure 7.1.: Mean squared error (MSE) performance for four selected default datasets. Figure 7.1(a) shows the elevator dataset, 7.1(b) the Pol dataset, 7.1(c) the calhousing dataset and 7.1(d) the house8L dataset.

variation of parameter C between 0.001 and 100, focusing on the range between 0.1 and 1.0. Figure 7.3 show the performance on the default datasets Elevators and Cal housing. The performance of the best and the worst algorithm setting for PA-I and PA-II is shown. Following values for the parameter C were analyzed:



(a) PA-I (b) PA-II

Figure 7.2.: Prediction with PA-I ((a)) and PA-II ((b)) on test data according to Equation 7.6. Two different settings of hyperparmeter C, i.e. $C = 0.01$ and $C = 1$, are shown to demonstrate its impact.



(a) Elevators (b) Cal housing

Figure 7.3.: MSE Performance of both PA variants with different settings for the aggressiveness parameter C. Figure 7.3(a) shows the performance on the Elevators dataset and Figure 7.3(b) shows the performance on the Cal housing dataset.

Algorithm 4: PA algorithm for categorical variables with parallel algorithm instances.

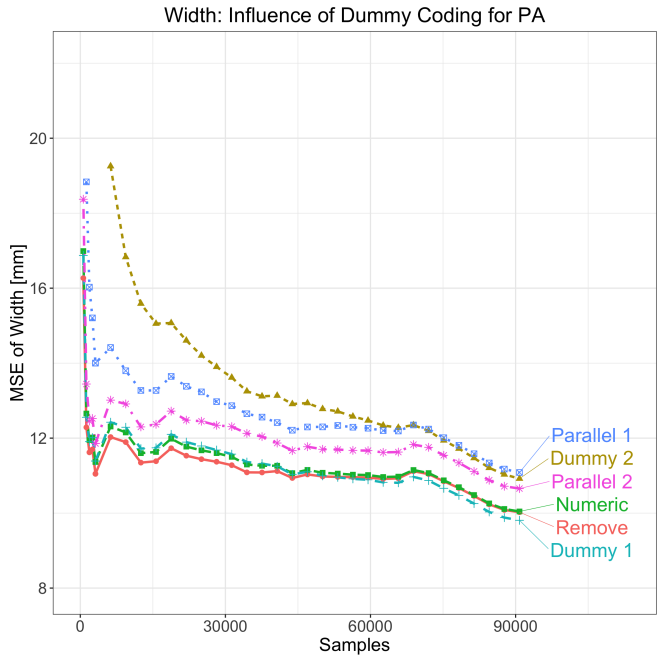
1 let $\boldsymbol{P}$ be a vector of k categorical variables;
2 let i be the new sample with $\boldsymbol{P_i} = (P_1(i), P_2(i), \cdots, P_k(i))^T$;
3 let $\boldsymbol{S}$ be a set of models $M(\boldsymbol{P})$ for each combination of the k categoricals;
4 **foreach** *new sample* i **do**
5 update coefficients $\boldsymbol{w}$ of base model with all numerics in current sample i based on the chosen PA-variant (PA, PA-I, PA-II);
6 **if** $M(\boldsymbol{P_i}) \notin \boldsymbol{S}$ **then**
7 create new instance of algorithm with same variant (PA, PA-I, PA-II);
8 initialize the coefficients $\boldsymbol{w}$, the tolerated loss ϵ and the aggressiveness parameter from the base model.;
9 update $\boldsymbol{w}$ of specific model $M(\boldsymbol{C_i})$;

Values of C: 0.001, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 10, 100

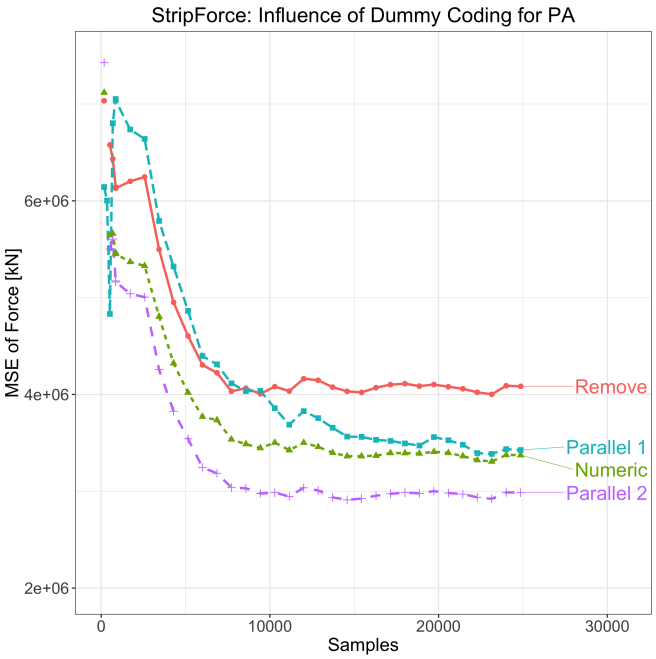
The reason for choosing these values is because the best performance is expected between 0.1 and 1.0. It can be seen, that the parameter C has a huge impact on total performance as well as on the dynamic behavior.

7.1.3. Handling Categorical Variables

The usage of different dummy coding schemes as described in Section 5.3.6 for the PA algorithm is affecting the performance on the datasets depending on the influence of the categorical variables and its correlation. For online processes only the parallel algorithm instances, the conversion to numeric or the removal of the categorical variable is applicable since the different levels are not known. If they are known in advance, regular dummy coding may also be used. The pseudo-code in Algorithm 4 describes the algorithm for the case, that parallel algorithm instances are used. If the parallel algorithm instances should only compensate the differences to the main model then the handling has slightly to be adopted. Instead of Step 8, during the initialization, the coefficients are not copied. Instead, in Step 9 only the residual error from the main model is passed to the specific model algorithm.
The MSE performance with different handling of categorical variables is exemplified using the datasets Width and StripForce in Figure 7.4. Its very interesting to see that for the *width* dataset the performance of the different coding strategies has only minimal impact on the performance, while the performance on the StripForce dataset is highly dependent on the coding strategy. For reference, also the classical



(a) Width



(b) StripForce

Figure 7.4.: MSE over the different datasets and samples with different dummy coding strategies. The rolling datasets Width 7.4(a) and StripForce 7.4(b) are shown to illustrate the behavior. For the Width dataset also the strategies for offline algorithms are shown as comparison. For the StripForce only the online usable dummy coding strategies Numeric, Remove, Parallel 1 and Parallel 2 are shown.

dummy coding and the dummy coding with all combinations of other features are shown in Figure 7.4(a).
It seems, that the categorical variables in the *width* dataset have almost no or even a negative impact on the performance so that the relationship between our target value and the numeric features can be best described with just neglecting the categorical variables. The reason why the parallel algorithm instances has the worst performance is obvious. The basic correlation has to be learned by each of the algorithm instances and if dynamic changes are occurring it has to be recognized by each instance. This will take much more time than if this change has only to be learned by one instance. If, on the other hand, some changes occur specifically within one level of a categorical variable, then this concept can highly benefit from the implementation.
In the StripForce dataset however, the categoricals and also the correlations between them and the numerical features have a greater impact. The strategy for using a correction algorithm for each combination of categorical achieves the best performance while removing the categorical has the worst performance. The reason why the algorithm for each combination of level, i.e. the parallel version, performs worse is based on the fact that this dataset consists of two categorical variables with 24 and 4 different levels. In total, this will result in 96 different instances of the algorithm. Since they will occur already at the beginning the material independent relationship between the numerical variables and the target value has to be compensated by each single instance before the accuracy improves. This can be seen when looking at the performance on the first 10 000 samples. Here, the parallel version achieves almost the worst performance among all dummy coding strategies. This clearly indicates that this strategy should not be used for a warm-start application since it will take too long to achieve a certain performance.

7.1.4. Performance

The default datasets were chosen in order to compare the implementation against publicly available performance achievements. Table 7.1 lists the performance of the PA variants, an offline linear model and the results achieved by an online regression tree described in [Iko12]. The best online performance is marked bold. It can be seen that the PA algorithms are in most cases superior to the offline version. Furthermore, also the MSE performance is better than the achievements with online regression trees in four cases.
Result for the rolling dataset are compared with the other implemented online algorithms after their discussion.

Table 7.1.: Performance on the default dataset for the PA algorithm variants. As comparison also the offline linear model and the results achieved by an online regression tree described in [Iko12] is shown. Only dummy coding strategies which are applicable are used for comparison. The online algorithm which achieves the best performance is marked bold.

Problem	Lin	PA	PA-I	PA-II	IKO
Abalone	8.4	6.5	4.3	**4.1**	5.7
Cal housing	7.5e9	3.3e9	2.6e9	**2.4e9**	5.1e9
Elevators	1.0e-5	5.3e-6	5.3e-6	**4.9e-6**	2.2e-5
House 8L	1.8e9	4.0e9	2.5e9	2.3e9	**1.1e9**
House 16H	2.1e9	4.4e9	2.7e9	2.7e9	**1.6e9**
Mv delve	1.5	9.0	8.6	**6.6**	17
Pol	9.9e2	7.2e3	2.4e3	2.1e3	**2.3e2**

7.2. Recursive Least Squares

The least squares method is a very popular method used in many fields for regression analysis and dates back to 18th century. The main idea of the least squares method is to estimate the n coefficients $\boldsymbol{\beta}_i\,(i = 1, \ldots, n)$ in the linear regression formula,

$$y(t) = \beta_1 x_1(t) + \beta_2 x_2(t) + \cdots + \beta_n x_n(t) \tag{7.7}$$

where $\mathbf{x}_t = (x_1(t), x_2(t), \cdots, x_n(t))^T$ represents the input data at time t. The coefficients $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_n)^T$ are estimated in such a way that the squared sum of residuals for the whole dataset of size N is minimized.

$$E\left(\boldsymbol{\beta}\right) = \sum_{i=1}^{N} \left(y_i - \mathbf{x}_i^T \boldsymbol{\beta}\right)^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \tag{7.8}$$

If Equation 7.8 is minimized with respect to $\boldsymbol{\beta}$ it follows:

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^\mathbf{T}\mathbf{X}\right)^{-1} \mathbf{X}^\mathbf{T}\mathbf{y}. \tag{7.9}$$

This solution is only valid if all rows of $\mathbf{X}$ are linear independent and thus $\mathbf{X}^\mathbf{T}\mathbf{X}$ is positive definite and the inverse matrix exists. For the calculation of n coefficients at least n observations of $\mathbf{x}$ and y are required. The estimation of $\hat{\boldsymbol{\beta}}$ requires the computation of the inverse of a matrix product and has therefore a high complexity of $\mathcal{O}(N^3)$. Furthermore, with each new observation the complete matrix has to be inverted and with each observation the input matrix $\mathbf{X}$ grows. The Recursive Least Squares (RLS) algorithm computes the coefficients recursively without the requirement of a matrix inversion. The idea for using a recursive variant should be illustrated on the following example taken from [Str10].

Example 3. Let us assume we have received 99 values of variable y, i.e. $y_1, \ldots, y_{99}$. The average value is $m_{99} = \frac{1}{99}\sum_{i=1}^{99} y_i$. Now we want to update the calculated average on arrival of a new sample y_{100}, without calculating the total sum again. We can use m_{99} for the calculation of m_{100} and get:

$$m_{100} = \frac{99}{100} m_{99} + \frac{1}{100} y_{100} = m_{99} + \frac{1}{100}\left(y_{100} - m_{99}\right). \tag{7.10}$$

This can be seen as an recursive update instruction. It contains the old value m_{99} and an *innovation* term which expresses the new information in sample y_{100}.

This method gains popularity because of its low complexity and is widely used in the field of control systems [AW94]. Let us assume that $\hat{\boldsymbol{\beta}}_i$ are the estimated

coefficients at time i and that $\mathbf{X}_i$ is the input matrix at time i. Then the prediction at time t is:

$$\mathbf{y}_{t+1} = \mathbf{X}_{t+1}\hat{\boldsymbol{\beta}}_{t+1}. \tag{7.11}$$

With equation 7.9 and with

$$\mathbf{X}_{t+1} = \begin{bmatrix} \mathbf{X}_t \\ \mathbf{x}_{t+1} \end{bmatrix}$$

we obtain:

$$\begin{aligned} \hat{\boldsymbol{\beta}}_{t+1} &= \left(\mathbf{X}_{t+1}^T\mathbf{X}_{t+1}\right)^{-1}\mathbf{X}_{t+1}^T\mathbf{y}_{t+1} \\ &= \left(\begin{bmatrix}\mathbf{X}_t^T & \mathbf{x}_{t+1}\end{bmatrix}\begin{bmatrix}\mathbf{X}_t \\ \mathbf{x}_{t+1}\end{bmatrix}\right)^{-1}\begin{bmatrix}\mathbf{X}_t^T & \mathbf{x}_{t+1}\end{bmatrix}\begin{bmatrix}\mathbf{y}_t \\ y_{t+1}\end{bmatrix} \\ &= \left(\mathbf{X}_t^T\mathbf{X}_t + \mathbf{x}_{t+1}\mathbf{x}_{t+1}\right)^{-1}\left(\mathbf{X}_t^T\mathbf{y}_t + \mathbf{x}_{t+1}y_{t+1}\right) \end{aligned}$$

With equation 7.11,

$$\mathbf{P}^{-1} = \mathbf{X}_t^T\mathbf{X}_t \tag{7.12}$$

and the matrix inversion lemma [PTVF92]

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B\left(C^{-1} + DA^{-1}B\right)^{-1}DA^{-1} \tag{7.13}$$

we get:

$$\hat{\boldsymbol{\beta}}_{t+1} = \mathbf{P}_t - \frac{\mathbf{P}_t\mathbf{x}_{t+1}\mathbf{x}_{t+1}^T\mathbf{P}_t}{1 + \mathbf{x}_{t+1}^T\mathbf{P}_t\mathbf{x}_{t+1}}\left(\mathbf{P}_t^{-1}\hat{\boldsymbol{\beta}}_t + \mathbf{x}_{t+1}y_{t+1}\right) \tag{7.14}$$

$$= \hat{\boldsymbol{\beta}}_t + \frac{\mathbf{P}_t\mathbf{x}_{t+1}}{1 + \mathbf{x}_{t+1}^T\mathbf{P}_t\mathbf{x}_{t+1}}\left(y_{t+1} - \mathbf{x}_{t+1}^T\hat{\boldsymbol{\beta}}_t\right) \tag{7.15}$$

Equation 7.15 represents the determination of the coefficients without the expensive matrix inversion. The update of the coefficients is expressed as a scalar correction factor which is multiplicated by an error vector. The only problem is that this equation still contains the matrix $\mathbf{P}$ but this matrix can also be expressed in a recursive form [You14]:

$$\mathbf{P}_{t+1} = \mathbf{P}_t - \frac{\mathbf{P}_t\mathbf{x}_{t+1}}{1 + \mathbf{x}_{t+1}^T\mathbf{P}_t\mathbf{x}_{t+1}}\mathbf{x}_{t+1}^T\mathbf{P}_t \tag{7.16}$$

With equations 7.15 and 7.16 all coefficients can easily be updated without taking into account previous samples. The memory consumption is only dependent on the size of each sample.
One problem which arises is that the initialization of $\mathbf{P}$ is only possible if the matrix $\mathbf{X}_t^T\mathbf{X}_t$ is invertible and this will only be the case after the number of samples is greater than the number of features received with each sample. A workaround described in [AW94] is to initialize the matrix $\mathbf{P} = \mathbf{P}_0$ where $\mathbf{P}_0$ is a positive definite matrix with sufficient large values on the diagonal.
If the process is time variant and the coefficients may therefore be changed over time it is beneficial to introduce a decay factor or forgetting factor λ with $0 \leq \lambda \leq 1$. This factor is interpreted as weighting factor for the new data. While the actual data are weighted with $\lambda^0 = 1$, old data is weighted with $\lambda^i, i > 0$. The matrix $\mathbf{P}_{t+1}$ and the coefficients $\hat{\boldsymbol{\beta}}_{t+1}$ can then be calculated by:

$$\hat{\boldsymbol{\beta}}_{t+1} = \hat{\boldsymbol{\beta}}_t + \frac{\mathbf{P}_t\mathbf{x}_{t+1}}{\lambda + \mathbf{x}_{t+1}^T\mathbf{P}_t\mathbf{x}_{t+1}}\left(y_{t+1} - \mathbf{x}_{t+1}^T\hat{\beta}_t\right) \tag{7.17}$$

$$\mathbf{P}_{t+1} = \frac{1}{\lambda}\left(\mathbf{P}_t - \frac{\mathbf{P}_t\mathbf{x}_{t+1}}{\lambda + \mathbf{x}_{t+1}^T\mathbf{P}_t\mathbf{x}_{t+1}}\mathbf{x}_{t+1}^T\mathbf{P}_t\right) \tag{7.18}$$

A major advantage of the RLS algorithm is its low computational complexity. The number of features which should be taken into account have to be defined prior to its usage. Adding or removing of features requires resizing of the matrix $\mathbf{P}$. When the exponential forgetting factor is incorporated, the algorithm can also handle time varying effects like concept drift in the data or abrupt changes of features. However, the forgetting factor λ is normally fixed so the factor should be chosen as a trade off between taking into account new data and keeping old information. A further advantage is that the sample data doesn't have to be stored because the coefficients can be calculated sequentially.
Several variants of the RLS algorithm were developed. Some of them are are using different weighting functions, like linear weighting of the last n samples, others are making other assumptions about the noise term. A very good overview and detailed analysis over all these algorithms can be found in [You14].

7.2.1. Influence of Exponential Decay

To analyze the impact of the exponential decay λ, Figure 7.5 shows the influence of the exponential decay factor on sample data generated with the *gap5* function introduced in Equation 7.6.
After half of the total samples have been seen by the algorithm, i.e. at $x = 5$, the slope is instantly halved and an offset is introduced. This data is used to train three different RLS variants with exponential decay λ set to 0.9,0.98 and

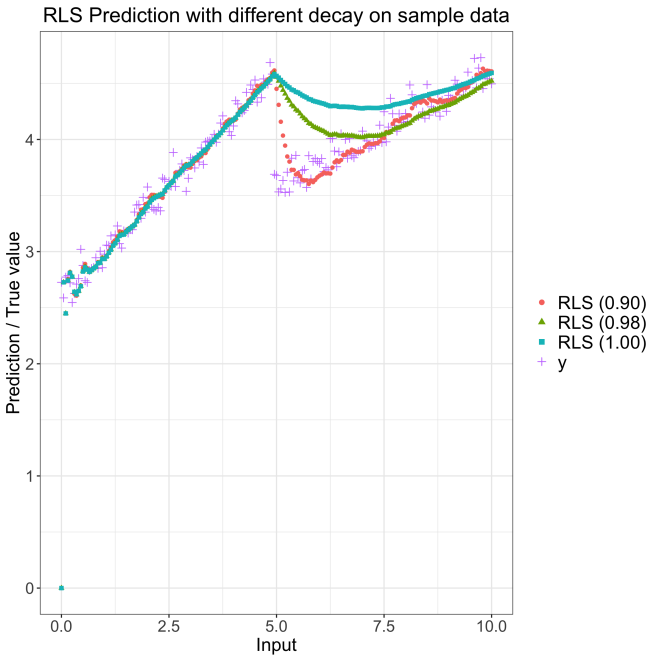
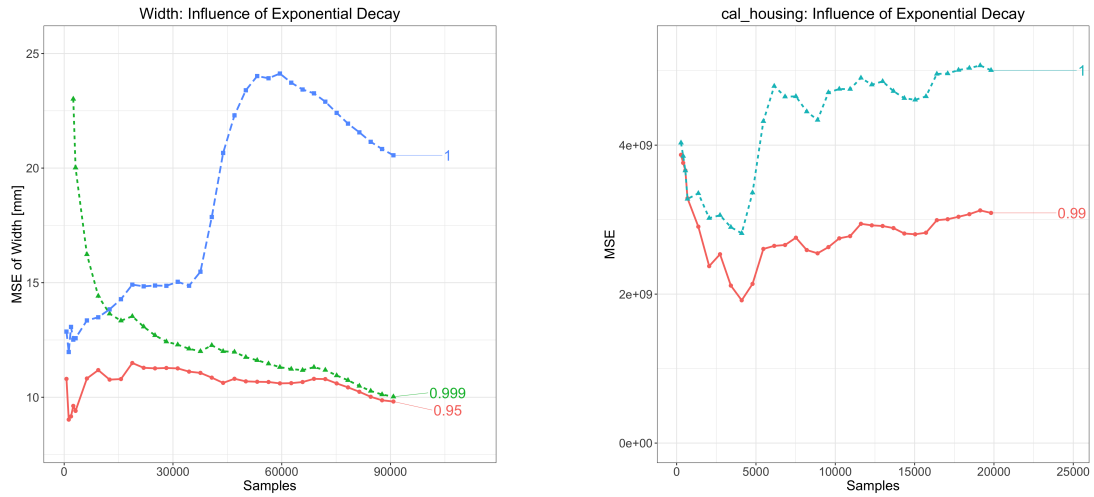


Figure 7.5.: Influence of exponential decay on test data. Shown is the prediction on generated data according to Equation 7.17. y denotes the true training sample, RLS(λ) denotes the prediction of the RLS algorithm with exponential decay factor set to λ.

1.0. Until the sudden change in data occurs all three RLS variants have almost the same prediction performance. At the point of the sudden change in data, the variant with lowest decay (0.9) is adapting to that change much faster than the other RLS variants. The slowest adaptation is done when no decay is (1.0) is used.

To analyze the impact on real-world applications, the performance for the rolling data set width and the default dataset calhousing is analyzed. The performance of the Width dataset is shown in Figure 7.6(a) and the performance of the Cal housing dataset is shown in Figure 7.6(b) respectively. Experiments were conducted with different exponential decays λ from the interval $[0.9, 1.0]$ and the two best results are shown in Figure 7.6. Some of the experiments had numerical instabilities. This happens if the arriving data are linearly dependent and the matrix $\mathbf{X}_i^T\mathbf{X}_i$ becomes almost singular. Then, the corresponding matrix $\mathbf{P}$ tends towards infinity. To overcome this problem a proper variable selection is required. Using only a small set of variables is therefore preferred if a cold-start problem has to be optimized. The performance for the three RLS variants in the Width dataset shows the huge impact of the exponential decay. For the decay factor $\lambda = 1$, the impact on the event between samples 35 0000 and 50 000 of the dataset can clearly be seen. If



(a) Width dataset

(b) Cal housing dataset

Figure 7.6.: MSE over the whole dataset for the Width dataset 7.6(a) and the Cal housing dataset 7.6(b) for different values of the exponential decay. The two best experiments are shown for both datasets. To demonstrate the sensitivity of the decay, an additional experiment is also shown for the Width dataset.

lower values are used this dynamic change can easily be corrected and almost no dynamic effect in the residual MSE will be visible.
The Cal housing dataset on the other hand indicates that there is some dynamic behavior in the data which can be reduced with a lower exponential decay, but not completely corrected. Experiments with lower values of the exponential decay showed instabilities during prediction and are therefore not shown. The choice of the exponential decay should therefore be made carefully.

7.2.2. Handling Categorical Variables

For the handling of categorical variables, all online methods mentioned in Section 5.3.6 are compared. For the parallel algorithm instances the categorical RLS algorithm will look for an already existing model for the specific group and will instantiate a new one if no group has been found. The pseudo code for this case is depicted in Algorithm 5. The baseline model will always be fed with the available sample and represents the general correlation. If the parallel algorithm instances are used to correct a difference to the baseline model, then the initialization in Step 8 is modified. Instead of copying the coefficients from the main model, the initialization is done from scratch because it is assumed that the correlations are

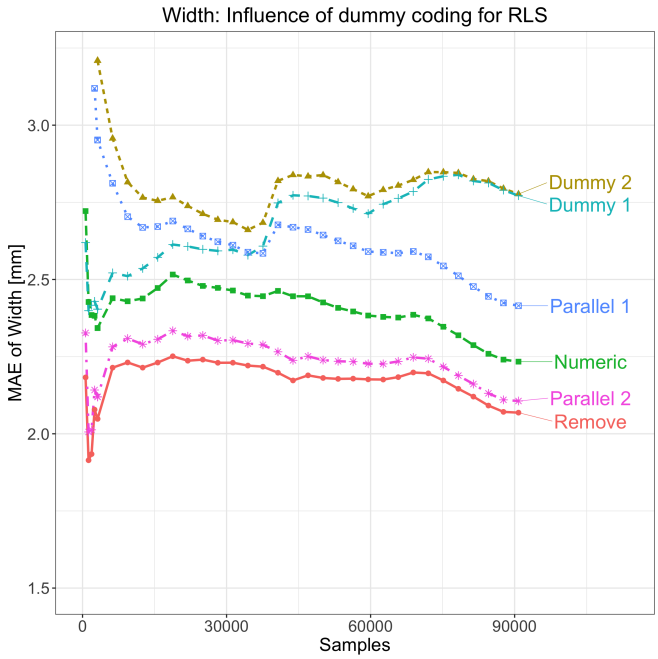
Algorithm 5: RLS algorithm for categorical variables.

```
1 let C be a vector of k categorical variables;
2 let i be the new sample with C_i = (C_1(i), C_2(i), ··· , C_k(i))^T;
3 let S be a set of models M(C) for each combination of the k categoricals;
4 foreach new sample i do
5     update matrix P and coefficients β̂ of main model with all numerics in
        current sample i;
6     if M(C_i) ∉ S then
7         create new instance of algorithm;
8         initialize the exp. decay, matrix P, and estimated coefficients β̂ from
            the main model.;
9     update P and β̂ of specific model M(C_i) ;
```

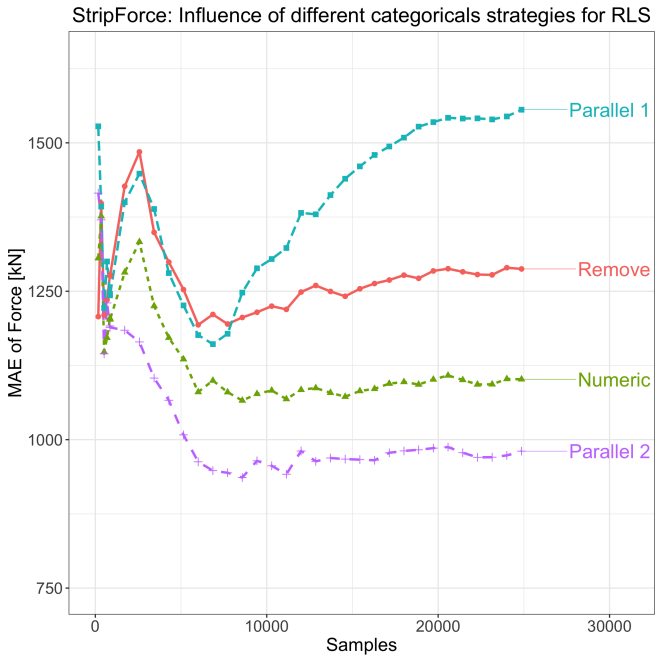
very specific for each instance and no general error has to be compensated. Then, in Step 9, the residual error of the main model is used to train the specific models. Two of the rolling datasets, i.e. Width and StripForce, had also at least one categorical variable. Their performance is depicted in Figure 7.7.

The Width dataset contains only the furnace as categorical variable and according to the performance in Figure 7.7(a) it will have no positive effect on the performance. Yet, it will decrease the performance, if it will be included in the model. This is the reason why the strategy to remove the variable achieves the highest performance. If the categorical variable is used with all possible correlation terms, the performance will drop significantly. This is shown within the performance for the strategy of dummy coding with all possible correlations (Dummy 2). In this case, the model will have many features. Due to the fact, that the drift in the data cannot be explained by the categorical variable, the performance will drop. This is especially visible within strategy Parallel 1, where for each level of the categorical a separate model instance is used.

The RLS performance on the StripForce dataset is showing a completely different behavior. Inclusion of the categorical variables in the online algorithm is increasing the performance significantly. Since the number of levels within the categorical is very high for this data set, the parallel treatment of those variables will take some time to learn the basic relationships. Furthermore, if dynamic changes occur within the main process and not within a certain category, the change has to be learned by each instance of the algorithms. This can clearly be seen when looking at the performance development at approximately sample number 9 000. The performance when using a separate version for each occurrence of the categorical (Parallel 1) is continuously getting worse, while the other strategies are almost constant (Numeric, Parallel 2) or have a significantly less performance



(a) Width dataset



(b) StripForce dataset

Figure 7.7.: Mean absolute error (MAE) for the RLS algorithm for different categorical handling strategies on two different real world datasets. For the Width dataset 7.7(a) additionally to the online applicable coding strategies (Numeric, Remove, Parallel 1 and Parallel 2), the dummy coding and full dummy coding strategy is shown. Figure 7.7(b) shows the performance for the StripForce dataset.

Table 7.2.: Performance on the default datasets for the RLS algorithm. On the categorical variable, only dummy coding strategies which are applicable online are used for comparison. Shown is the mean squared error (MSE) for the complete dataset. As reference the performance of a linear model which was trained on the first 10% of the data is also shown. Column IKO shows the performance achieved with online regression trees by Ikonomovska [Iko12].

Problem	Lin	RLS	IKO
Abalone	8.4	4.6	5.7
Cal housing	7.5e9	3.0e9	5.1e9
Elevators	1.0e-5	8.9e-6	2.2e-5
House 8L	1.8e9	1.9e9	1.1e9
House 16H	2.1e9	2.0e9	1.6e9
Mv delve	1.5	2.0	17
Pol	9.9e2	1.9e3	2.3e2

drop (Remove). The usage of a base model with separate model instances which are calculating corrections for the specific material group seems to be clearly the most favorable solution.
Based on these observations, we recommend to neglect the categorical variables for cold-start problems. After some samples have been received the impact of the categorical should be analyzed. Then a decision can be made if the categorical should be considered in the algorithm or not. Depending on the levels of the categorical variables a proper strategy can be selected.

7.2.3. Performance

A summary of the RLS performance in terms of MSE on the default datasets is given in Table 7.2. As a reference, also the performance of an offline linear model which was trained on the first 10% of the data is shown. Furthermore, a comparison to the results achieved by an online regression tree from [Iko12] is made. The comparison on the rolling datasets Width, StripForce and PlateForce is made after the discussion of the next online algorithm.

7.3. Online Support Vector Regression

Support Vector Machines (SVMs) have been developed in the late nineties at AT&T Bell Laboratories by Vapnik and others [SS04] as a nonlinear generalization of the Generalized Portrait algorithm [VL63]. The Generalized Portrait algorithm, also originally developed by Vapnik, Lerner and Chervonenkis in the sixties, was grounded in the statistical learning theory. Most of the literature about SVM is focusing on classification rather than regression problems with SVMs. Support Vector Regression (SVR) is considered as a special case of SVMs with engineering as main application field [SS04].
Cauwenberghs and Poggio[CP01] developed an exact solution for recursively training of SVMs for classification. Ma and Theiler [MTP03] and Martin [Mar02] extended this algorithm for the regression case. Although this algorithm would allow its online usage, there are still some problems which need to be solved and understand. An important aspect of this is the potential optimization possibility of the corresponding SVR parameter and the determination of a suitable storage size.
In this chapter first the idea of standard SVR is presented and afterwards the incremental and decremental variant of SVR, based on the Accurate Online Support Vector Regression (AOSVR) Algorithm presented in [MTP03], is shown. After demonstrating the effect of the storage size and the management of it, two variants for optimization are introduced and the handling of categorical variables are compared. The chapter is concluded with recommendations for different real-world scenarios.
A more detailed introduction to support vector regression is given in [SS04].

7.3.1. Introduction to Support Vector Regression

The support vector algorithm has its origin in the statistical learning theory and dates back to the sixties but was further developed at AT&T Laboratories [SS04]. Due to this, the main focus has always been real-world applications although the application field was more in the area of optical character recognition. There are two variants of support vector regression available: ϵ-SVR and ν-SVR. ϵ-SVR was originally developed 1995 by Vapnik [Vap95] for pattern recognition whereas ν-SVR was developed three years later by Schölkopf and Smola [SSWB00].
The main advantage of ν-SVR is the fact, that it eliminates the usage of the parameter ϵ in case of regression and the regularization constraint C in case of classification. This is done by reformulation of the optimization problem and controlling the amount of support vectors with the parameter ν. For real world applications the target is to establish a best possible prediction. The usage and

optimization of parameter ϵ is of great interest and can easily be interpreted physically. The number of support vectors used in the algorithm is somehow a parameter which is hard to translate and therefore the focus in this thesis will be on ϵ-SVR. A detailed comparison between both variants is given in [CL01].
The main idea for support vector regression is to find a function that fits some training data $\mathbf{x}_i \in \mathbf{R}^N, i = 1, \ldots, l$ with a maximum deviation of ϵ, i.e. :

$$f(\mathbf{x}) = \mathbf{w}^T \Phi(X) + b. \tag{7.19}$$

$$|y_i - f(\mathbf{x}_i)| \leq \epsilon, \forall i \tag{7.20}$$

Function 7.19 can be seen as linear combination of basis functions $\Phi(X)$ with weighting factors $\mathbf{w}$ and some additional offset or bias b. The basis function Φ maps the inputs to a feature space F and is known as kernel function in literature. The usage of this kernel functions allows the regression to be extended to nonlinearity. Valid kernel functions should in general be symmetric, continuous and their correlation matrix should be positive semi-definite. All kernel functions which fulfill these conditions are called Mercer kernels. Somehow, there have also been found some kernels which are not satisfying these conditions but still may be used. A prominent example is the sigmoid kernel with mapping:

$$\Phi(x, z) = \tanh\left(\vartheta + \kappa x^T z\right)$$

Here, ϑ is an intercept constant and κ is the slope constant. It is also possible to define a linear combination of two kernels because it can be shown that a linear combination of two Mercer kernels is again a Mercer kernel. Throughout this section the SVR with linear kernel, polynomial kernel and radial basis function (RBF) kernel are used. They are defined as follows:

$$\Phi(x, z) = x^T z \quad \text{(Linear)}$$

$$\Phi(x, z) = \left(x^T z + \gamma\right)^p \quad \text{(Polynomial)}$$

$$\Phi(x, z) = e^{-\gamma \|x - z\|^2} \quad \text{(RBF)}$$

The parameter γ for the polynomial case defines whether the kernel should be homogeneous ($\gamma = 0$) or inhomogeneous ($\gamma > 0$). Parameter p defines the degree of the polynomial. The RBF kernel has a parameter γ which will initially set to $\gamma = \frac{1}{2\sigma}$ in this thesis where sigma is the variance of the target value. All those parameters have to be tuned in order to get the best results for each kernel and each dataset. The optimization problem itself will be the same for each kind of kernel and therefore we will not specify the kernel in the following discussion of the SVR algorithm.
Equation 7.19 should approximate our function with an accuracy of ϵ. This means

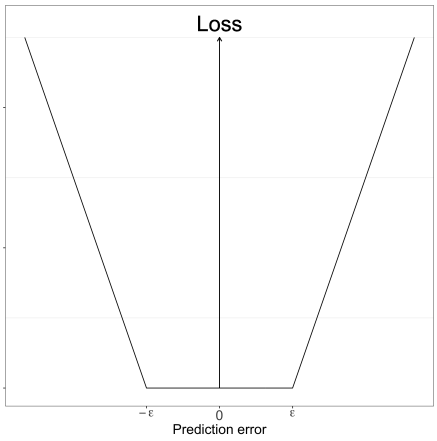


Figure 7.8.: ϵ-insensitve loss function for SVR. Prediction errors with deviation less than ϵ will not be penalized. A deviation above will be penalized linearly dependent on the parameter C

that every sample which has a prediction error of less than ϵ will not affect the regression algorithm. This is known as ϵ-insensitive loss which is shown in Figure 7.8. Here, the loss for a deviation $\leq \epsilon$ is zero. Clearly, if the parameter ϵ is chosen too high, than the approximation would be very poor. A too small parameter ϵ would sometimes be infeasible because of the presence of noise. Therefore, this parameter has to be optimized for each dataset. A good starting point can be selected if the root cause of a random error, e.g. for an physical experiment, is known. Typical values can also be derived from the measuring accuracy with which the data have been stored. The regression function should be as flat as possible which means that the weighting factors $\mathbf{w}$ should be as small as possible. This can be represented as a convex optimization problem:

$$\text{minimize} \left\{ \frac{1}{2}||w||^2 + C \sum_{i}^{l} (\xi_i + \xi_i^*) \right\} \tag{7.21}$$

$$\text{subject to:} \begin{cases} y_i - \mathbf{w}^T \Phi(\mathbf{x}_i) - b \leq \epsilon + \xi_i \\ \mathbf{w}^T \Phi(\mathbf{x}_i) + b - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0, i = 1, \cdots, l. \end{cases} \tag{7.22}$$

The slack variables ξ and ξ_i^* in Equation 7.21 are introduced because the problem is not always feasible and a solution would not always exist. The constant C penalizes the data points which lie outside of the desired ϵ band and can therefore been seen as an optimization variable for our problem. It also defines how flat our function will be and how big the penalty on positive and negative deviations

greater than ϵ can get. The underlying loss function which was used here is known as the ϵ insensitive loss function shown in Figure 7.8. Points within a prediction error of ϵ will have no associated loss while the loss on errors above $|\epsilon|$ will have a loss increasing linearly with parameter C. This is the most typical loss function for SVR but also other loss functions may be used. In [SS04] the authors are deriving solutions for other loss functions, like Laplacian or Gaussian loss functions.
After introducing Lagrange multipliers α_i, α_j^* and defining $Q_{ij} = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$ as a general kernel function, the optimization leads to the Lagrange formulation:

$$\begin{aligned} L &= \tfrac{1}{2} \sum_{i=1}^{l} Q_{ij} \left(\alpha_j - \alpha_j^*\right) (\alpha_i - \alpha_i^*) + \epsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) - \sum_{i=1}^{l} y_i (\alpha_i - \alpha_i^*) \\ &\quad - \sum_{i=1}^{l} (\delta_i \alpha_i + \delta_i^* \alpha_i^*) + \sum_{i=1}^{l} (u_i (\alpha_i - C) + u_i^* (\alpha_i^* - C)) \\ &\quad + b \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) \end{aligned}$$

Here, $\delta_j, \delta_j^*, u_j, u_j^*$ are Lagrange multiplier. If we optimize this Lagrange formulation we get the Karush Kuhn Tucker (KKT) conditions:

$$\begin{array}{llllll} \frac{\partial L}{\partial \alpha_i} & = & \sum_{j=1}^{l} Q_{ij} \left(\alpha_j - \alpha_j^*\right) + \epsilon - y_i + b & = & \sum_{j=1}^{l} Q_{ij} \beta_i + \epsilon - y_i + b \\ \frac{\partial L}{\partial \alpha_i^*} & = - & \sum_{j=1}^{l} Q_{ij} \left(\alpha_j - \alpha_j^*\right) + \epsilon + y_i - b & = - & \sum_{j=1}^{l} Q_{ij} \beta_i + \epsilon + y_i - b \\ \frac{\partial L}{\partial b} & = & \sum_{j=1}^{l} \left(\alpha_j - \alpha_j^*\right) & = & \sum_{j=1}^{l} \beta_j = 0 \\ u_i^* (\alpha_i^* - C) & = & 0, \qquad u_i (\alpha_i - C) & = & 0 \\ \delta_i \alpha_i & = & 0 \qquad \delta_i^* \alpha_i^* & = & 0 \end{array}$$

with the simplification $\beta_i = (\alpha_i - \alpha_i^*)$. Analogues to [MTP03] a margin function $h(\mathbf{x}_i)$ is defined

$$h(\mathbf{x}_i) = f(\mathbf{x}_i) - y_i = \sum_{j=1}^{l} Q_{ij} \beta_j - y_i + b \tag{7.23}$$

and each sample can be assigned to one out of three subsets.

Error Set: $\begin{cases} h(\mathbf{x}_i) \geq \epsilon, \quad \beta_i = -C \\ h(\mathbf{x}_i) \leq -\epsilon, \beta_i = C \end{cases}$

Support Set: $\begin{cases} h(\mathbf{x}_i) = \ \epsilon, -C \leq \beta_i \leq 0 \\ h(\mathbf{x}_i) = -\epsilon, \ 0 \leq \beta_i \leq C \end{cases}$

Remaining Set: $\left\{ -\epsilon \leq h(\mathbf{x}_i) \leq \epsilon, \beta_i = 0 \right.$

Example 4. To illustrate the SVR principle an easy example is shown in Figure 7.9. Samples were created using a linear combination of a sine function with a linear dependency and some random noise σ_e, i.e.:

$$f(x) = 0.3\sin(5x - 2) + 0.4(x - 1) + \sigma_e.$$

The true function is drawn without noise and is depicted as blue solid line. 20 samples which are equally distributed along the x-axis are used for training and marked green. After training of the SVR - we used here already the online SVR algorithm without loss of generality - the prediction of the SVR along the x-axis is shown as red solid line. The maximum allowed deviation to this prediction is shown as a dashed solid line. The margin functions for support vectors have to be exactly ϵ or $-\epsilon$ and are exactly on the dashed lines. These samples are marked "S" for Support Set Samples. Samples outside the desired ϵ band belong to the error set and are therefore marked "E". All other samples lie in the ϵ band and are belonging to the remaining set. Note that these samples have no influence on the prediction because their weight is zero.

A more detailed introduction to SVR can e.g. be found in [VGS97].

7.3.2. Incremental Support Vector Regression

The standard SVR requires to solve a convex optimization problem which might take too much time for the usage within an online system, where a continuous stream of data is received. A possibility to use this offline version of the SVR is to window the dataset and to calculate the solution every time from scratch. But this is only feasible with a very small amount of data and dependable variables. For larger datasets and fast incorporation of new samples an iterative solution is needed.
The first incremental algorithm for SVMs for classification was developed by Cauwenberghs and Poggio [CP01]. This algorithm was further developed for the regression case by Martin [Mar02] and Ma and Theiler [MTP03]. The main difference to standard SVR is its incremental and decremental update mechanism to calculate the support vector and the corresponding sets. Regularly, the quadratic equation given in Equation 7.21 has to be solved for the whole data set. Incremental SVR ensures that with each new sample which is added, the KKT conditions are still fulfilled for all samples rather than computing everything from the beginning. Therefore, a suitable value for the β_c for a new sample c has to be determined. Also, while changing the coefficient β_c for the new sample, all other coefficients and the offset b must be updated. With $\Delta b, \Delta\beta$ and Δh specifying the change of

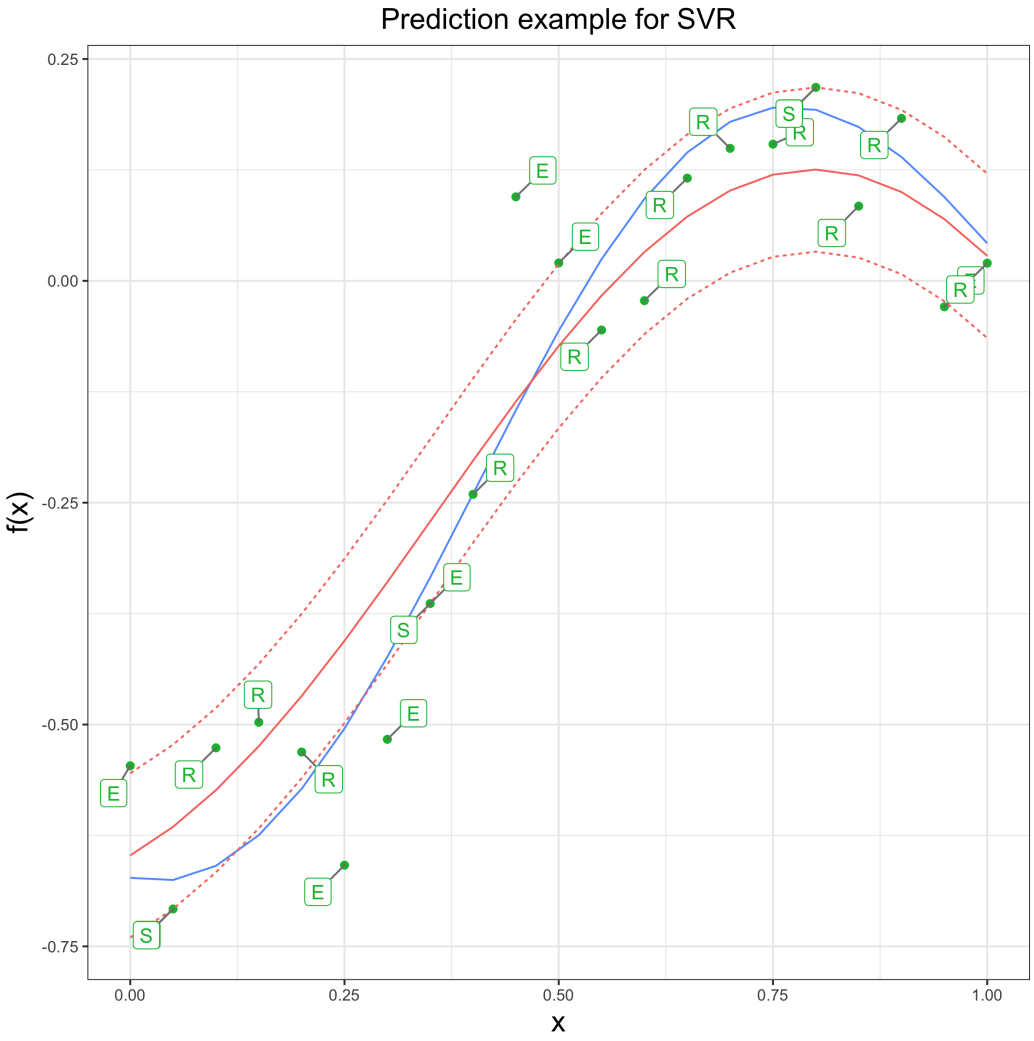


Figure 7.9.: Prediction example to illustrate the SVR principle. The blue solid line is the true function and the green dots represent the samples which were created based on the true function with additional noise added. The red solid line is the prediction of the SVR after training of all examples. The dashed lines represent the maximum allowed deviation ϵ. Every sample which is directly on the dashed lines belong to the support set (S). Samples with higher deviations are belonging to the error set (E), samples with lower deviation than ϵ are belonging to the remaining set (R).

the offset, the coefficients and the margin vector and

$$\sum_{j=1}^{l} \beta_j = 0, \tag{7.24}$$

we get:

$$\Delta h(\mathbf{x}_i) = Q_{ic}\Delta\beta_c \sum_{j \in S} Q_{ij}\Delta\beta_j + \Delta b. \tag{7.25}$$

For samples which are in the support set S, $h(\mathbf{x}_i)$ will only change if the samples migrate to another set. Therefore for samples which stay in the support set $h(\mathbf{x}_i) = 0$ and it follows $\forall i \in S$:

$$\sum_{j \in S} Q_{ij}\Delta\beta_j + \Delta b = -Q_{ic}\Delta\beta_c \tag{7.26}$$

$$\sum_{j \in S} \Delta\beta_j = -\Delta\beta_c \tag{7.27}$$

With

$$Q = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & Q_{s_1,s_1} & \cdots & Q_{s_1,s_{ns}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{s_{ns},s_1} & \cdots & Q_{s_{ns},s_{ns}} \end{bmatrix}^{-1}, \tag{7.28}$$

and $s_1 \cdots s_{ns}$ specifying all support vectors it follows that:

$$\begin{bmatrix} \Delta b \\ \Delta\beta_{s_1} \\ \vdots \\ \Delta\beta_{s_{ns}} \end{bmatrix} = -Q \begin{bmatrix} 1 \\ Q_{s_1 c} \\ \vdots \\ Q_{s_{ns} c} \end{bmatrix} \Delta\beta_c \tag{7.29}$$

For samples within the Error or Remaining Set, $h(\mathbf{x}_i)$ will change according Equation 7.25. $\Delta\beta_j$ and Δb can be calculated using Equation 7.29 and $\Delta h(\mathbf{x}_i)$ can therefore be easily calculated.
Up to now only the case where all samples will stay in their corresponding set was considered. The idea for the migration of samples is, that if $\Delta\beta_c$ is too high and samples would migrate to another set, than the change of the weight is done in multiple steps. The maximum allowed change $\Delta\beta_{cmax}$ will be calculated such that all samples will just stay in their sets. Therefore for each sample a corresponding maximum allowed change of their weight has to be calculated.

Migration of Samples in Support Set

Samples within the Support Set have a deviation of $\pm\epsilon$ and a weight of $0 < \beta_i < |C|$. The samples are migrating to the remaining set if their weight is approaching 0. They are migrated to the error set if β_i gets C. With Equation 7.27 and

$$\vartheta = -Q \begin{bmatrix} 1 \\ Q_{s_1 c} \\ \vdots \\ Q_{s_{ns} c} \end{bmatrix} \tag{7.30}$$

it follows from equation 7.29 that:

$$\max\Delta\beta_c = \begin{cases} -\frac{\beta_i}{\vartheta_i}, & \text{sign}\left(\vartheta\Delta\beta_c\right) > 0, h(x_i) = \epsilon \\ -\frac{C+\beta_i}{\vartheta_i}, & \text{sign}\left(\vartheta\Delta\beta_c\right) < 0, h(x_i) = \epsilon \\ \frac{C-\beta_i}{\vartheta_i}, & \text{sign}\left(\vartheta\Delta\beta_c\right) > 0, h(x_i) = -\epsilon \\ -\frac{\beta_i}{\vartheta_i}, & \text{sign}\left(\vartheta\Delta\beta_c\right) < 0, h(x_i) = -\epsilon \end{cases} \tag{7.31}$$

Migration of Samples in Remaining Set

Samples from the remaining set have a deviation $h(x_i) < |\epsilon|$ and can only migrate to the support set. This will happen when the deviation $h(x_i)$ changes to $h(x_i) = \pm\epsilon$. With $\{r_1 \dots r_n\}$ representing the samples in remaining set, $\{s_1 \dots s_n\}$ representing the samples in support set, c representing the actual sample and

$$\gamma = \begin{bmatrix} Q_{r_1 c} \\ Q_{r_2 c} \\ \vdots \\ Q_{r_n c} \end{bmatrix} + \begin{bmatrix} 1 & Q_{r_1 s_1} & \cdots & Q_{r_1 s_n} \\ 1 & Q_{r_2 s_1} & \cdots & Q_{r_2 s_n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{r_n s_1} & \cdots & Q_{r_n s_n} \end{bmatrix} \vartheta \tag{7.32}$$

the maximum allowed change of $\Delta\beta_c$ can be calculated before a sample migrates to the support set:

$$\max\Delta\beta_c = \begin{cases} \frac{\epsilon - h_i}{\gamma_i}, & \text{sign}(\gamma\Delta\beta_c) > 0 \\ \frac{-\epsilon - h_i}{\gamma_i}, & \text{sign}(\gamma\Delta\beta_c) < 0 \end{cases} \tag{7.33}$$

Migration of Samples in Error Set

Samples within the error set have a deviation $h(x_i) > |\epsilon|$ and can only migrate to the support set. The calculation of the maximum allowed change is similar to the

Algorithm 6: online SVR algorithm.

```
let S be the a vector of samples in Support Set, E a vector of samples in Error
  Set, and R be a vector of samples in Remaining Set ;
let c be the new sample which should be incorporated;
calculate margin (error) h_c for current sample;
if h_c ≤ ε then
    add sample c to remaining set and exit
while c not added to S or E do
    calculate minimum Δβ_c until sample c migrates to set S or E;
    foreach sample i in R do
        calculate minimum allowed change of β_c until sample i migrates from R
          to S;
    foreach sample i in S do
        calculate minimum allowed change of β_c until sample i migrates from S
          to R or E
    foreach sample i in E do
        calculate minimum allowed change of β_c until sample i migrates from E
          to S
    set Δβ_c as minimum allowed change for all stored samples;
    change weight of current sample c by Δβ_c and migrate samples if necessary.
      update weight β for all samples and calculate new margin h;
    if min Δβ_C was determined by sample c then
        add sample c to E or S
```

remaining set and we get:

$$\max\Delta\beta_c = \begin{cases} \frac{-\epsilon-h_i}{\gamma_i}, & \text{sign}\,(\gamma\Delta\beta_c) > 0, h(x_i) < 0 \\ \frac{h_i-\epsilon}{\gamma_i}, & \text{sign}\,(\gamma\Delta\beta_c) < 0, h(x_i) > 0 \\ \infty, & \text{otherwise} \end{cases} \tag{7.34}$$

Then the sample, which defines the minimum change, will be migrated into another set and the weight β_c is further changed until all samples meet the KKT criteria. A pseudo-algorithm of this procedure is shown in Algorithm 6.
A detailed description of this procedure can be found in [Mar02, MTP03, CP01]. The removal of a sample is done in the same way: The weight of the specific sample is reduced in multiple steps until it is zero. If it is already in the remaining set it directly can be removed.

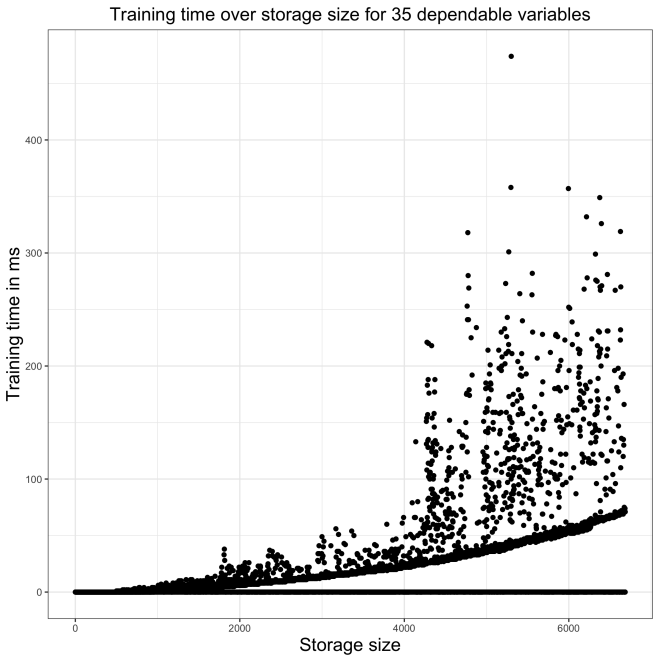


Figure 7.10.: Training time in ms over sample size for incremental SVR. The storage size is assumed to be indefinite and no sample was removed.

Storage Management

The incremental behavior for the SVR algorithms enables it for applications where the time for a complete training cannot be afforded. For incorporation of a new samples, the algorithm is not forced to learn from scratch. So the expected time to incorporate a new sample is significantly reduced. When a new sample arrives, it has to be guaranteed that all samples in storage still fulfills the KKT condition, while the weight of the new sample is incrementally modified. So the time is strongly correlated to the storage size of the SVR. Figure 7.10 shows the training time of an industrial problem with 35 dependable variables, although the computational power of industrial machines are much higher than on notebooks. This test was done on a regular i5 notebook and should only give an indication on training time.

The plot shows, that the training time continuously increases but also the variation of training time is increasing. For this dataset and parameter setting, up to 4 000 samples can easily be integrated with training times below 100ms. Above this value, the variation of the training time is too high, which might lead to infeasible computing times. It can also be seen that some of the samples have training time of almost 0. This is the case for samples which are predicted within the tolerance of ϵ and can directly be added to the remaining set.
Let us assume a storage limit of 4 000 and that already 4 000 samples have been

received. For the next sample the algorithm has to decide, which samples are stored and which will not be considered anymore. This can be achieved with the decremental behavior of the online SVR algorithm which will work similar to the incremental behavior. The weight of the sample, which should be deleted, is continuously decreased while all other samples still have to fulfill the KKT condition. If the weight is zero, it can be removed.
The question is now how to select the proper candidates which should be stored and how to decide which should be removed? Cauwenbergs et al.[CP01] suggest to only take so-called "reserve" vectors into account which have a high chance to migrate to support vectors. But this would still only reduce the memory usage and not limit it within an infinite dataset. Another issue is caused by the dynamic behavior of the optimization problem. Actual samples with high prediction errors might end-up in the error set with no or only small chance to migrate to support vectors. But they are reflecting at best the current process with all its states. In [TL03] for statically incremental support vector machines also the least relevant values will be kept instead of always removing the last one.
Four different strategies to select the samples which should be removed are considered. The most intuitive solution would be to balance the usage of new samples and old ones in order to adapt to new situations very fast without loosing the capability for the prediction of rare situations. In general, also the storage size should be considered. In detail, the following strategies are used:

1. **Oldest:** The strategy Oldest is just removing the oldest sample in the storage and is based on the idea, that the best description of the actual process state is obtained by the most recent samples. This will ensure that the algorithm will always adapt to new situations.
2. **Distance:** The distance based strategy (Distance)will compare on arrival of a new sample the Euclidean distance to the stored samples. The sample with the closest distance will be removed first and afterwards the new sample will be added. To ensure that the latest sample are not removed on arrival of the next sample, the most recent 30 % of the samples are protected. The idea behind this strategy is to balance between rare situations and dynamic changes.There will be additional costs for the distance calculation or the calculation of the least promising candidate which should be considered when parameters like storage size are chosen.
3. **Random:** Strategy Random will just remove a random sample from the dataset.
4. **Reserve:** When using the strategy of removing the least promising candidate (Reserve), the possibility for migration of samples belonging to the remaining set or error set will be considered. The sample with the lowest chance to migrate into the support set will be removed.

There will be additional costs for the distance calculation or the calculation of the least promising candidate which should be considered when parameters like storage size are chosen.

7.3.3. Stability

Stability is a central aspect for real-world applications. It is essential that the algorithm will not show any side effects or unexpected behavior. There are certain situations where the iteration might become unstable, e.g., if duplicate samples or negative eigenvalues occur or if the process limits are violated. These problems will be discussed in the following.

Duplicates

The first reason for instability which should be considered are duplicate samples. Duplicate samples may cause the iteration process to run into an endless loop. Assume a sample which is already stored in one of the three sets is fed again to the online algorithm. If the stored duplicate is not in the remaining set, it will have a weight unequal to zero. But this would mean, that the new arriving sample, which starts with weight zero, will also not belong to the remaining set since it will have the same prediction error as the already evaluated duplicate. The new sample will start with a weight of zero and has to incrementally update its weight. With steps 7 through 15 of the Algorithm 6, both weights will be incrementally updated and the iteration process may oscillate. Duplicates in hot rolling may occur if the online process is not properly designed. An example of such a scenario can be seen if the measurement values are send twice to the process which triggers the online algorithm.
The implemented online SVR algorithm will check each new sample and will reject any duplicate. The check is done based on the euclidean distance of the new sample to the set of stored samples. A disadvantage of this check is that also updates of the error are not possible since the check is only based on the independent variables. However, this feature can easily be implemented. The stored sample can be removed and afterwards the algorithm will be fed with the updated version.

Negative Eigenvalues

Laskov et al. [LGK+06] demonstrated that the phenomena of immediate cycling is only possible if the kernel matrix is not positive semi-definite. Immediate cycling refers to a situation where a sample migrates from one set to another and will immediately migrate back to its origin within the next iteration.
To prevent the kernel matrix from becoming not positive semi-definite, a small value, i.e., a regularization constraint, is added to the kernel matrix. The constraint should express the highest possible negative eigenvalue and will be added to the diagonal of the kernel matrix. Since it is not known how high this value might get, two situations are addressed within the code. The first one regularly introduces a small regularization constraint for the kernel matrix calculations. The second one is detecting problems during iterations. The detection is basically an iteration counter which assumes a maximum number of possible migrations during the procedure. Once such a situation was detected the algorithm will recalculate the complete kernel matrix and add a higher regularization constraint. For some cases even this will not help because there is another possible reason for an instability which can be found in the numerical representation of the data.

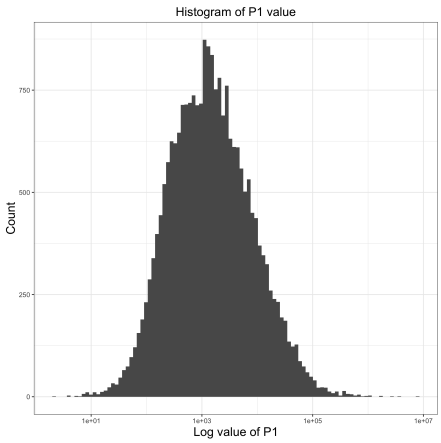


Figure 7.11.: Histogram of the target value from the House 16H dataset.

Double Precision

Every system has certain precision boundaries. The programming languages R and C++ are both using the norm IEEE 754 which defines the standard for floating point computations. It is important that both are using the same standard because otherwise data would already be lost or precision would be wasted at the interface between both languages. To be able to properly use the online SVR algorithm, the data is scaled in advance as described in Section 5.3.5. The scaling limits are either set by a process expert or, in case of a warm-start usage, based on the samples already received. If the samples have the same magnitude and ranges the scaling procedure can also be skipped. The question is now what would be the best way to scale the data? A usual naive approach would be just to scale them to minimum and maximum. This would lead to a range between zero and one or minus one and one. If the data contains extreme outliers, this scaling method can cause unexpected effects. Consider for example the distribution of the P1 value from the House 16H dataset. Figure 7.11 shows the distribution of it with a logarithmic x-axis. The minimum value of P1 is zero and its maximum is approximately 5.5e5. When the scaling to minimum and maximum is applied here most of the values fall in the range between 5e-5 and 6e-4. This is unfortunate for the algorithm and the precision ranges can easily be exceeded. Clearly, if this information is known in advance a different scaling to this value should be used. The online SVR approach is vulnerable to floating point errors. A huge number of computations have to be made and the scaling of the input and output values is of great importance. With an improper kernel choice and scaling the double precision limit can easily be violated. This has to be prevented. Therefore the

recommendation is not to scale the values between zero and one but to scale according to specified percentiles. A profound and robust choice is to scale to the first and third quartile, i.e., the 25th and 75th percentile.

7.3.4. Parameter Influence

Regular SVR performance as well as the incremental SVR performance are strongly dependent on the specific parameters of the algorithm. These are ϵ for the maximum tolerable error, C which describes the penalty term and the kernel parameter(s). While the traditional offline SVR algorithm can directly optimize the parameter for any given dataset, the online algorithm lacks this possibility. Strategies which are successfully applied for optimization, e.g., cross-validation, can not easily be transferred to the online variant. The dynamic behavior of datasets would be completely neglected for traditional cross-validation.
For cold-start optimization problems, where no data have been seen, a good initial parameter set is hard to find. There are some parameter settings which can be seen as "default settings" but are definitely not the best parameters for every data set. This is one of the consequences of the No-Free Lunch theorem [Haf16]. Once the parameters have been chosen there is no strategy for switching over to new parameters as this would require a completely recalculation of every single data point.
The initial parameter settings for all considered SVR algorithms throughout this thesis were optimized prior to their usage. The algorithms were used with optimized parameter settings in dependence on their storage size. This means, that an online SVR algorithm with a storage size of 200 samples was optimized with parameters that achieve the best performance on the first 200 data. For higher storage sizes more samples were used, i.e., it was assumed that the algorithm will process at least the number of samples corresponding to the maximum capacity of the storage. The parameter optimizations were performed with the R package nloptr [Joh18] with parameters shown in Table 7.3.
The initial parameters for γ and the penalty C were chosen to be 1 since it turned out to be a reasonable start value for most of the datasets. The value for ϵ is set to represent the half of the standard deviation of the target value.

7.3.5. Influence of the Storage Size and Storage Management

The sample storage for the SVR is a central aspect and will significantly influence the performance in terms of accuracy but also properties like time needed for

Table 7.3.: Parameters passed to the optimization function. The package nloptr[Joh14] was used to conduct the optimization.

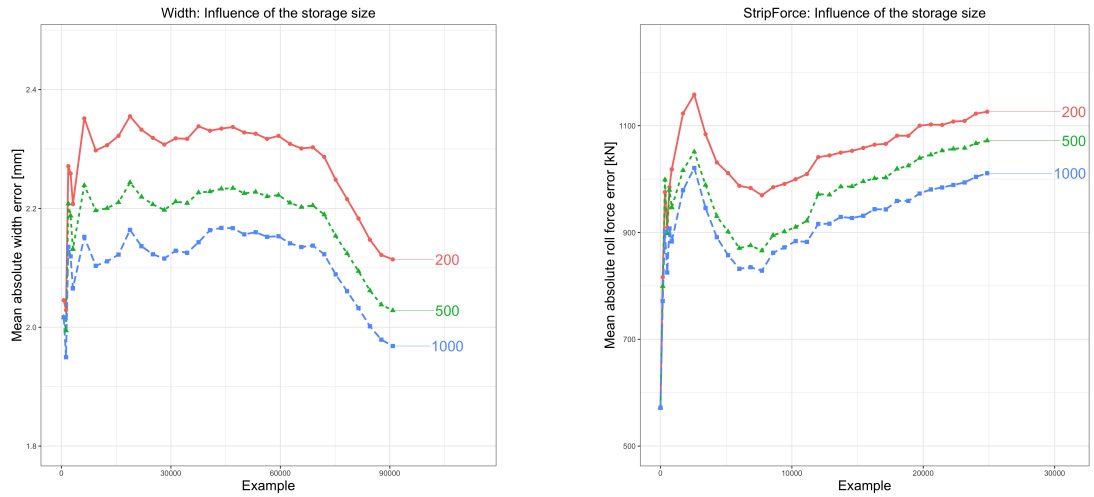
Name	Value	Description
algorithm	NLOPT_LN_SBPLX	A variant of the Nelder-Mead algorithm which is operating on sequences of subspaces. It was originally developed by Tom Rowan [Row90].
x0	$(0.5\sigma, 1, 1)$	Start values for insensitivity loss ϵ, penalty C and RBF kernel parameter γ.
lb	$(0.0001, 0.0001, 0.0001)$	Lower bound for (ϵ, C, γ).
ub	$(0.5, 10, 10)$	Upper bound for (ϵ, C, γ)
maxeval	50	Function evaluations for nloptr

learning and ability to predict rare process states. Mainly, two parameters determine these properties: Storage size N_s and the storage management strategy M_s. The storage size N_s defines the maximum number of samples which may be stored internally and the management strategy M_s defines a way to select samples which should be removed once this maximum is reached. Both parameters are discussed in the following.

Storage Size

The storage size N_s within the online SVR determines how much samples are stored and therefore also how much memory of the previous measurements is kept. It will have a great impact on both, performance and run-time. Special materials may be rolled only on rare occasions and keeping them in memory would probably increase the performance. But since the system might be very dynamic this cannot be guaranteed and an adaptation to the new learned samples can be favorable. With a proper choice of the storage size the process experts have therefore to balance between fast adaptation and a more robust performance but slower adaptation.

The choice of the storage size will further highly correlate with the training time which can be seen as a limit to the storage. The performance for the online SVR with storage sizes of 200, 500 and 1 000 is depicted for two rolling datasets in Figure 7.12. On the left side the performance for the Width dataset is shown, and on the right side the performance of the StripForce dataset is shown. Performance in both cases is the mean absolute error (MAE). For industrial applications, the MAE is favorable because of its interpretability. The algorithms were used without considering the categorical variable. To have a fair comparison, all versions are using the same kernel (RBF) and the same strategy for the deletion of samples



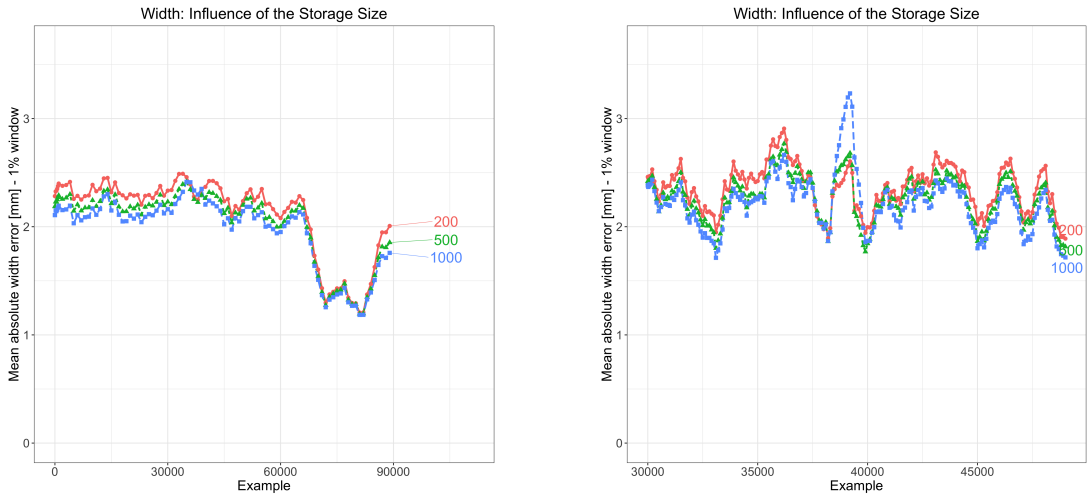
(a) Width (b) StripForce

Figure 7.12.: Mean absolute error of the roll force for the two rolling datasets Width and StripForce with three different storage sizes. The categorical variables were not considered (removed) and the RBF kernel was chosen. Further, the oldest sample was removed (Oldest).

(Oldest).

It can be seen that the performance in terms of MAE is increasing with a higher value of storage size. For the first part of the StripForce dataset, it seems that the performance of the SVR with 500 and 1 000 samples is almost the same. Only after a while they are continuously drifting away from each other. The dataset has no high dynamic and almost all of the changes are occurring because of material changes.

The Width dataset has a strong dynamic impact at approximately sample 40 000 (see Figure 6.1). Here, the performance of the SVR with 200 samples will be reacting faster than the variants with more storage. The curves will get closer together. In order to proof this, Figure 7.13 shows a the mean absolute performance over a fixed windows size for 5% of the data. Therefore, the total achieved performance at the end will also only present the last 5% of the data but the dynamic behavior can be analyzed much better. Between samples 30 000 and 50 000, it can be seen, that the algorithm with storage size of 200 is achieving an even better performance than both other version at this point of the dataset (see Figure 7.13(b)), although the improvement is only marginal.



(a) Full dataset

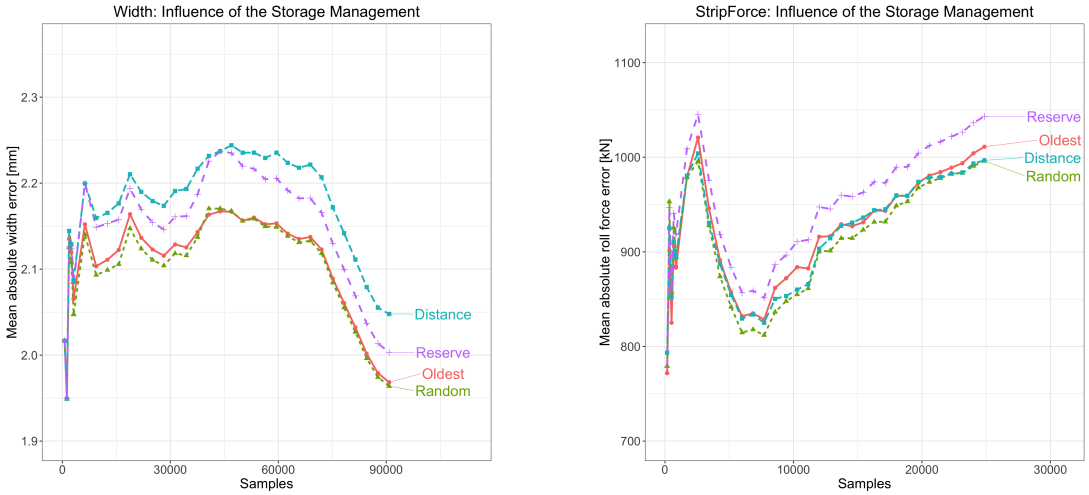
(b) Samples 30 000 to 50 000

Figure 7.13.: Windowed mean absolute error of the Width dataset. The values are representing the mean MAE of the last 5% of the data. This is done in order to analyze the dynamic behavior of the data. Figure (a) shows the full dataset and Figure (b) focuses on samples 30 000 to 50 000

Storage Management Strategy

The storage management strategy M_s is deciding which samples should be kept and which should be removed. This might be a simple choice a posteriori, but the decision is difficult if no information about future samples can be retrieved. Storing really rare samples will not be beneficial if this working area is not contained in some future samples. However, storing only similar samples is also not very promising when sudden changes occur. To analyze the impact on the selection of the strategy, Figure 7.14 depicts the performance for rolling datasets Width and StripForce in terms of mean absolute error. All categorical variables have been removed and only the RBF kernel performance is shown. Storage size was chosen to be 200, 500 and 1 000. Only the best performance for each strategy is shown.

For the Width dataset in Figure 7.14(a) the strategies to remove the oldest and to remove a random sample are performing almost identical. The distance-based strategy and the strategy to keep the reserve vector are slightly worse. The reason for this is the high dynamics within this dataset. All strategies which are keeping old samples in the dataset would cause a decrease in performance since they will not be suitable to describe the current state. A different behavior can be seen in Figure 7.14(b). The dataset contains some dynamic behavior but this is mainly caused by the categorical variable. To remove this effect, a special instance of the algorithm was used to correct the behavior to the main model (Parallel).



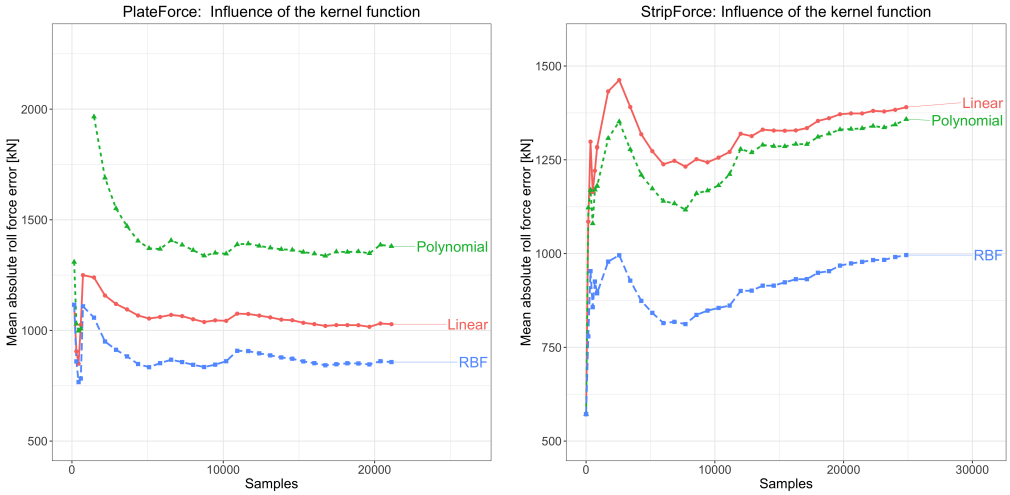
(a) Width (b) StripForce

Figure 7.14.: Mean absolute error of the width (a) for the Width dataset and the rolling force (b) for the StripForce dataset with four different storage management strategies. The decision which sample should be removed from storage is either done randomly (random), based on the sample distance (distance), the age (oldest) or on the possibility to migrate into the support vector set (reserve). The categorical variables were removed for the Width dataset. For the StripForce dataset a separate model instance was used (Parallel 2).

Therefore, the algorithm can benefit from storing some more rare sample rather than deleting them. The total mean absolute error of the distance based strategy has the best performance, followed by the random deletion of samples. Removing the oldest sample or keeping the reserve samples achieve the worst performance.

7.3.6. Selecting the Kernel Function

The kernel selection is a very difficult choice since it usually requires some knowledge about the data structure and the distribution of it. If those information are available prior to application, there are some general rules which can be taken over from regular SVR. Linear kernels and polynomial kernels may suffer from some numerical difficulties, especially if the kernel parameters are chosen inappropriate. The selection of an adequate degree of the polynomial, i.e., p, is a crucial issue. A too large degree can cause the polynomial kernel, $\Phi(x, z) = \left(x^T z + \gamma\right)^p$, to go to infinity if $x^T z + \gamma > 1$. This is also another reason why scaling is so important for kernel methods.
RBF and polynomial kernels are highly dependent on the settings of the hyper parameter of the kernel. The linear kernel does not need to be parameterized. The RBF kernel and the homogeneous polynomial needs tuning of one parameter and the inhomogeneous polynomial requests two parameter to be tuned.
The computational complexity of the online SVR algorithm will be highly dependent on the kernel choice. The kernel choice will define which correlations of the dependable variables can be taken into account for the optimization. The linear model allows only a linear separation in hyperplanes while polynomial of order $d > 1$ and RBF kernel allows the features to have correlations of higher order. So ideally, the choice would be to choose the kernel with the lowest computational complexity, which satisfies the desired performance.
The authors of libsvm software package advise to use the linear kernel instead of the RBF or polynomial kernel when the number of features is much higher than the number of instances [HCL16].
Figure 7.15 shows the MSE performance over the whole datasets for the rolling dataset PlateForce and StripForce. The polynomial kernel was a polynomial of degree five with the linear offset set to one, i.e. $\Phi(x, z) = \left(x^T z + 1\right)^5$. For the rolling datasets it seems that the RBF kernel has a very good performance. The kernel choice is closely related to the optimization problem and therefore it cannot be assumed that this behavior will always be the same. Interestingly, the kernels behavior at the beginning of the dataset is quite different. Within the Width dataset the polynomial kernel has huge inaccuracies at the initial stage of the prediction. This is again something which is on concordance to our expected per-



(a) PlateForce (b) StripForce

Figure 7.15.: Performance for datasets PlateForce and StripForce for different kernel choices. All dummy variables have been removed from the dataset. All other parameters are varied as described in previous sections. Only the best performing kernel algorithm is shown for each dataset.

Table 7.4.: Percentage of training time which was spent in the corresponding kernel calculation routine.

Kernel	Function	Consumption[%]
Linear	$\Phi(x,z) = \left(x^T z\right)$	11
Polynomial	$\Phi(x,z) = \left(x^T z + 1\right)^5$	56
RBF	$\Phi(x,z) = e^{-\|x-z\|^2}$	12

formance and closely related to the extrapolation difficulties. This will separately be addressed in Section 9.1. Beside the CPU power and the number of selected features, the kernel choice has also a great impact on the training time. The kernel choice is significantly consuming a great chunk of the whole training time for each individual sample. Table 7.4 shows a profiling result for all three different kernel selections with a storage size of 2 000 samples on the width problem with five features. The percentages are representing the amount of time for the training of one sample which was used for the calculation of the kernel products.

This means, that the choice of the kernel should be chosen carefully, especially for time critical applications and large storage sizes.

Based on the previous analysis, the best initial choice of the kernel function seems to be the RBF kernel. It offers a robust initial prediction accuracy with

a moderate complexity in terms of time consumption. After an initial phase of learning the kernel choice has to be rethought. This can be done by evaluating the performance of other kernels on the already seen data. After the discussion of the different treatment of the categorical variable, the training time will be discussed.

7.3.7. Handling Categorical Variables

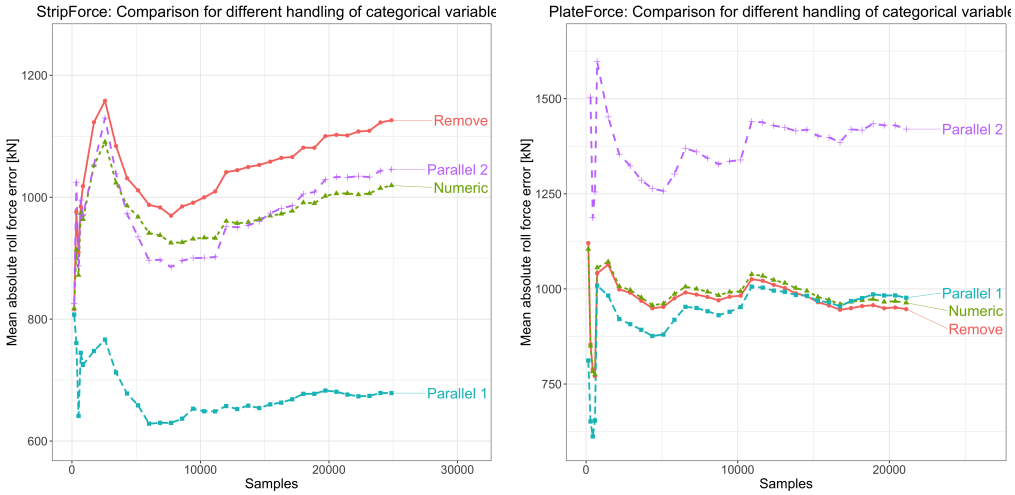
For real-world applications and especially for hot rolling it is important to have strategies for handling categorical variables. One of the major problems is the uncertainty regarding the number of levels for the categorical variable. Following strategies can be applied for the online SVR algorithm:

1. Removing categorical
2. Converting to a numeric value according to its level
3. Creating a separate algorithm instance for each unique combination (two variants)

The first two strategies can directly be implemented. The separate model instances are more complex to implement than the simpler algorithms PA and RLS. While the separate model instances in case of PA and RLS are initialized by copying the coefficient of the main model, the SVR algorithms lacks this feature. Copying of the whole stored matrices and vectors would be applicable but this will require too much time. The amount of memory which has to be copied is dependent on the number of training samples stored in the main model. Instead of initialization with parameters from the base model, the new algorithm instance will be initialized without any knowledge about the latest samples. Furthermore, the usage of a base model which is corrected by the special models will also be considered.

Figure 7.16 shows the MAE on the rolling datasets StripForce and PlateForce. All algorithms were using the RBF kernel and the oldest strategy for removing of samples and were using a storage size N_s of 200.

Especially the performance for the StripForce dataset in Figure 7.16(a) is highly dependent on the strategy how to handle categorical variables. As already pointed out in Section 6.2 the target value showed strong correlations to the material which was rolled. Therefore, it is not surprising that the strategy to use an own model instance for each categorical variable achieves the best performance. The PlateForce dataset shows a different behavior. The performance is depicted in Figure 7.16(b). At the initial learning the separate model instances show the best performance but the average performance over the whole dataset is almost the same



(a) StripForce (b) PlateForce

Figure 7.16.: Mean absolute error for datasets StripForce and PlateForce for different handling of the categorical variable. For comparison, only RBF kernel was used. The oldest samples have been removed. For both algorithms 200 samples have been stored ($N_s = 200$).

for all strategies. The worst performance is clearly achieved when a base model is corrected by the specialized model for each material (Parallel 2). The reason for this might be that the SVR parameters for the specialized model instances are not optimized. They are just copied from the base model. The base model parameter are optimized on the first samples as described in Section 7.3.4. Since the complexity and the memory requirements are higher if more algorithm instances are used the best choice would be to remove the categorical variable. Unfortunately, this behavior might change quickly, if new materials are rolled. This risk has to be considered when selecting the strategy for the treatment of categorical variables.

7.3.8. Training Effort

The main idea of the online SVR is to incrementally update parameters instead of solving the convex optimization problem directly. Therefore it is not surprising that the time for incorporation of new samples is much faster than calculating the solution for the complete dataset. One of the first papers which describes the online SVR method shows also a comparison between the libsvm package and their implementation of the online SVR algorithm[MTP03].
The exact training time will be a result of the complexity of the dataset, its

properties and the concrete implementation. Additionally the hardware, i.e. the processor, has a significant impact on the absolute time. A huge amount during training will be used for the handling of matrices and datasets. Comparing training times achieved with current state of the art machines with training times achieved in 2003 would be unfair. Therefore, this comparison is not made at this point. This section aims more for the discussion of the differences between various strategies and settings rather than the implementation itself. Some details about the implementation is described in Section 7.3.10.
The incremental update time for learning a new sample is of huge interest for online learning. The available time depends on the specific task and the application field of the algorithm. For reversing mills it is important to improve the prediction between two consecutive passes. This time should be as short as possible but has physical limitations due to the drives of the mill and some time for adjusting to the new pass. The complete incorporation of all measurements should not exceed one second. Only a small fragment of that time is available to the SVR algorithm. Obviously the time to incorporate a new sample will depend on the storage size of the SVR. A higher storage size leads to a higher chance of migration for already learned samples into other sets during the update procedure. So, more incremental steps to determine the weight of the new samples might be needed.
The chosen initial values for the SVR parameter have also a huge impact on the training time. A high value for the allowed deviation ϵ will force more samples into the remaining set. Samples in the remaining set however have a weight of zero and can directly be added without any migration. A too low value for ϵ will force more samples into the error set with maximum weight. When a new sample is added to the error set, chances for other samples to migrate into a different set is higher.
Figure 7.17 depicts the training time on the StripForce dataset with three different storage sizes. Parameter ϵ was chosen to be half of the standard deviation σ of the target value. It can be seen that the training time will continuously rise until the storage is completely filled. After this point, the training time will almost be constant. As already pointed out during the discussion of the storage size, the differences during training for each sample will be higher with higher storage size.
The choice of the kernel can have also a significant impact on the training time. The major part of this influence is founded in the problem description and their relationship. Some problems will require linear kernel and some may require the selection of RBF or polynomial kernel. The complexity for the calculation of the kernel itself is only one part. Another part with a major impact on the complexity is caused by the optimization problem itself, i.e., the dataset. This phenomena is depicted in Figure 7.18 where the training time for three different kernels are shown for datasets StripForce and PlateForce. For all experiments, a storage size of 1 000 and the strategy oldest was used. Further, ϵ was set to $0.5\sigma_t$ and the

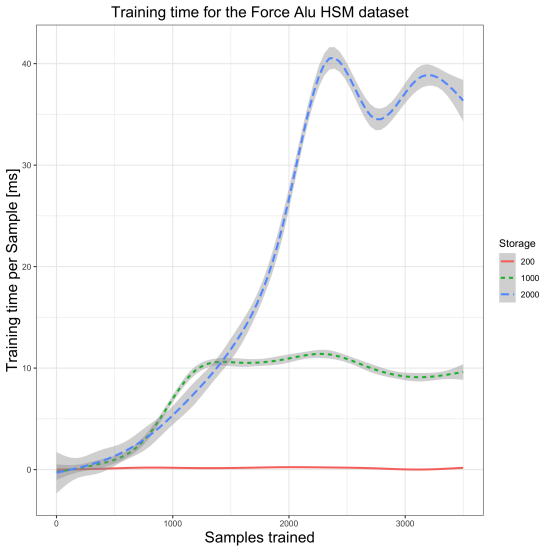


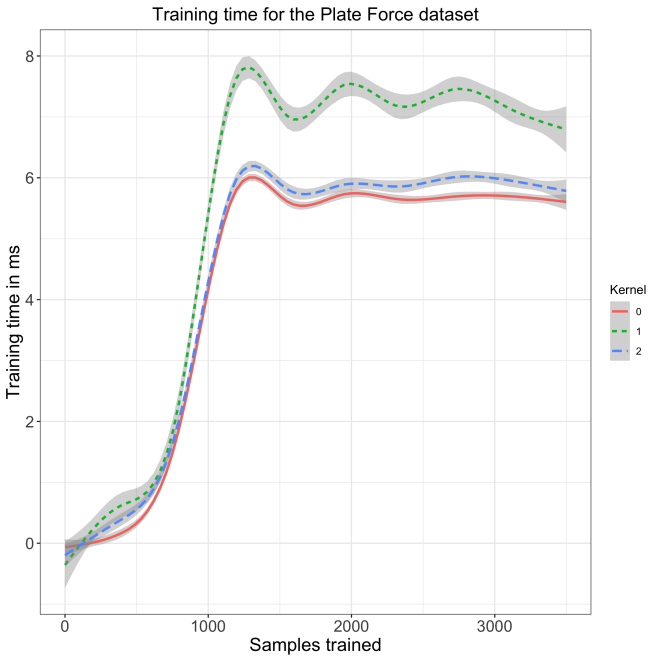
Figure 7.17.: Training time for one samples over the total number of samples already trained. Three different storage sizes N_s, i.e. 200, 1 000 and 2 000 are used. The parameter ϵ is chosen to be half of the standard deviation of the target value for the whole dataset.

penalty parameter C was set to 1.
It can be concluded that there is no significant difference during learning between the linear kernel and the RBF kernel. The polynomial kernel however has in general a much higher training time than the others. Another interesting effect can be seen during training on the StripForce dataset. The training time with the polynomial kernel is increasing until sample number 1 200. For the next 1 000 samples the training time is almost stable before it is decreasing continuously. This can only be explained with some structural changes within the dataset which is very hard to analyze. During training of the PlateForce dataset (Fig. 7.18) the training time continuously rises, until the storage reached the maximum capacity. Beside the training time, the selected parameter for the SVR will also determine the memory consumption which will briefly be discussed in the following.

7.3.9. Memory Requirement

The memory usage for the online SVR algorithm is an important property although it is not as critical as the training time. It will be more a limitation for the settings of the algorithm, i.e. a limitation to the storage size and the possibility of using the optimization. The total memory consumption will highly depend on

(a) StripForce



(b) PlateForce

Figure 7.18.: Training time per sample over the total number of samples already trained for different choice of the kernel function. The maximum storage size N_s has been set to 1 000 and the oldest sample will be removed for all variants. A parameter optimization was not used. The depicted kernels are linear (0), polynomial (1), or RBF (2) based kernels. On the top side the training time for the StripForce dataset is shown while on the bottom side the training time for the PlateForce dataset is sown.

Table 7.5.: Memory consumption on rolling datasets PlateForce, StripForce and Width for a storage size of 1 000 and 3 000. The amount of instances required for the parallel treatment of the categorical variable is also shown. Basically it is a multiplication of the levels for each individual categorical in the dataset. The maximum memory is a result based on a storage size N_s of 3 000 with every instance of the algorithm completely filled.

	Memory		Comb. of Levels	Max. Memory
Dataset	1 000	3 000		
PlateForce	15MB	80MB	12	960MB
StripForce	19MB	81MB	96 (24x4)	7.7 GB
Width	53MB	82MB	4	328 MB

the storage size and the number of features. The storage size is usually much larger than the number of features and thus it will be the dominating parameter for memory.
For hot rolling problems, the online SVR algorithm will be integrated into a prediction process for the mill. If the dataset contains categorical variables the major memory consumption will be defined by the choice of the strategy how the categorical variable is treated. When for each combination of the categorical variables a separate instance of the algorithm is used, then the memory consumption is much higher. Although not all of these data have to be kept in memory, also the total memory consumption has to be taken into account.
To get an idea about the consumption Table 7.5 lists the memory consumption with different storage sizes in dependence of the categorical treatment for the rolling datasets. It can easily be seen that the required memory allocation will quickly reach certain boundaries when using larger storage sizes and an improper handling of categorical variables. While all dataset will require less than 100 MB without special treatment of the categorical variables, the memory will tremendously increase when multiple instances are used. The StripForce dataset consists of 24 different materials and is using four stands which results in 96 different combination of the categorical. Consequently, the maximum storage size would be approximately 7.7 GB. When more material is rolled this value can easily increase to hundreds of GB or even TB. This has to be prevented, since this amount would require to store a large amount of data on hard drives. Instead, the amount of data which is used in the specialized algorithm instances has to be reduced or different strategies for handling of those data have to be chosen.

7.3.10. Implementation

The online SVR algorithm requires a huge amount of matrix and kernel calculations which would take too long in R. Because of speed, the whole implementation of the algorithm is done in C++. Although the time and memory measurements were done with OS specific implementations, the algorithm itself is implemented OS independent and was tested on Windows and MacOS. The C++ template library Eigen[GJ+10] was used to improve algebraic calculations. Different CPU profiler were used to detect bottlenecks and to optimize the matrix handling. The algorithm can be used completely within C++ but also provides an interface to R. This was created with the help of the package Rcpp [EF11]. The handling and managing of the different datasets, the matrix calculations and the different strategies for removal and optimization are implemented in C++. Every control parameter for the algorithm can either be passed through the R or C++ interface. Following functions are provided:

1. Training of one or more samples
2. Prediction of one or more samples
3. Controlling the size of the storage
4. Controlling the removal strategy
5. Control of the optimization strategy
6. Initial settings of the parameters
7. Removing of a sample

Removing of a specific sample can be used to update the target value of the corresponding sample. This is helpful if a second measurement is made and the reliability of this measurement is higher than the first one. Furthermore, sometimes problems with measurements are detected after learning already took place. Then this function can be used to remove those erroneous samples. The sample has to be passed by its dependable variables and will be detected through calculation of the euclidean distance. If the sample is found in one of the three sets it will be removed.
The function to control the optimization strategy will select a specific strategy how the SVR parameter may be updated online. This optimization is closely related to dynamic changes within the dataset which will be discussed in the following chapter.

8. Online Parameter Optimization

The previous chapters were describing the different online algorithms and their properties. Until now, all those algorithm parameters and changes have been done statically in dependence of the dataset. Parameters can been tuned when a warm-start problem has to be optimized. Without proper parameter tuning those online algorithms were achieving reasonable good performance. The idea is now to sequentially update the parameters during training in order to achieve an even better performance. This update has to be achieved without violating the time constraint. Furthermore, it is assumed that the requirement for these updates are bounded at least to some change within the data stream, i.e. some discrete events or drifts. Summarizing, answers to the following questions are analyzed:

Necessity:	Is online parameter optimization necessary?
Time:	How can parameter tuning with a time constraint be achieved?
Detection:	How to detect events and drifts?

The answer to these questions are the main topic for this chapter.

8.1. Necessity

If parameter optimization is required depends on many aspects. First and foremost the used algorithms define the possibility to optimize certain parameters. For the RLS algorithm the exponential decay can be modified and for the PA variants the regularization constraint can be used to optimize the predictions. The SVR algorithm is using the insensitivity loss, the kernel parameters and the violation constraint for prediction. The difference for those three algorithms is, that parameter changes can only be tested for the SVR algorithm. Here, different parameters can be used directly on the internally stored parameters for the online SVR algorithm. RLS and PA do not store any samples and additional implementations would be needed. So the indication if a parameter update should take place cannot be answered by the algorithm itself and an independent instance is required. The online SVR algorithm can test different parameter settings on the internal storage to calculate the performance.

Secondly, the requirement to update parameters is defined by the dynamic properties of the dataset. If the stream is almost static then clearly the necessity to perform an online parameter optimization is not given. But as soon as the data contains some time dependency the optimization can be beneficial. See e.g. Figure 7.5 which includes an event at a certain time. Clearly, a parameter update to adapt as fast as possible to the new situation will cause the performance to increase.
Finally, if the data contains categorical variables with differences in occurrence, chances are high to benefit from parameter optimization. Different occurrence of the categorical variables may show completely different relationships among the independent variables.

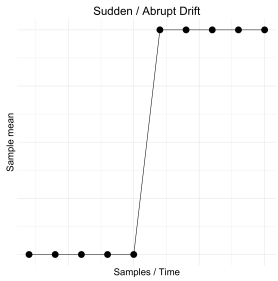
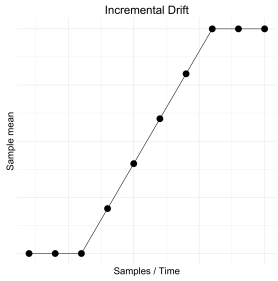
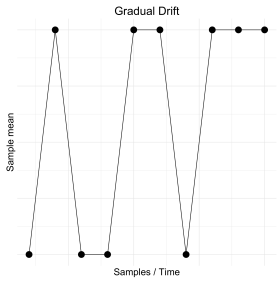
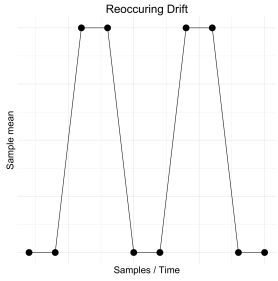
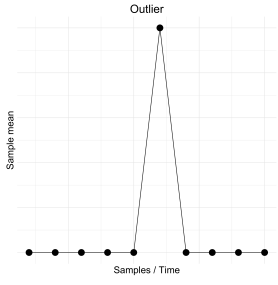
8.2. Time Constraint

The only algorithm where an optimization can be made without additional costs, is the online SVR algorithm. For all other described algorithms the parameters may be updated to external knowledge. The optimization of online SVR will take some time since the new parameters have to be validated on the internal storage. All common optimization algorithms have constraints to limit the number of iterations and some have also a time limit. During hot rolling with full production it may happen that the only spare time available is the transport time from a slab or ingot to the mill while the previous product has just finished rolling. The production is continuously trying to improve the capability of the mill and therefore minimizing those gaps. Usually, these gaps are below ten seconds. This will not be sufficient for many optimization procedures. If new parameters are tested they have to be added to the SVR algorithm with new parameters, i.e. the training of the whole storage has to be repeated. The time required for the parameter optimization of the SVR is therefore highly correlated to the storage size and the chosen kernel time. See Sections 7.3.8 where the training time is analyzed.
To be able to execute a parameter optimization, more than this time is needed. Each mill has some scheduled downtime for maintenance or roll change. If these downtimes can be detected, they can be used to trigger a more expensive optimization. The optimization will cause the algorithm to achieve a better performance in terms of RMSE on the internal stored samples. Since the parameter will also have an effect on the training time, the optimization also has to consider this aspect to ensure that the maximum training time is not violated.

8.3. Drift Detection

Dynamic changes are occurring inherently in real-world applications. Changes can affect the distribution over time of the dependent variables, the output variable or both. The change within the distribution of the dependent variables is referred in the literature as concept drift and the change of the conditional distribution of the output variable is called real concept drift [SG86, WK96]. Another type of drift is called virtual drift. According to [GvB+14] this term is not exactly specified but usually defines a situation where the distribution of the dependable variables changes but is not affecting the distribution of the target value.
For the hot rolling problems, the only drift where the algorithms need to be adjusted is the real concept drift. The regular concept drift, i.e., the change of the conditional distribution of input values may signal a change in production which cannot be avoided. The virtual concept drift is not affecting the output variable and therefore no action is required. In most real-world scenarios also hidden context drift can occur. Hidden context drift defines a situation where the drift is caused by variables not included in the dataset. Some of the strategies dealing with those kind of drift, e.g. to store concept descriptions, are described by Widmer et al. [WK96] for classification problems.
The real concept drift may manifest in different forms. Table 8.1 gives a summary of their time behavior analogues to [GvB+14] and lists possible reasons in the hot rolling context. Drift can occur in many different contexts and might seem to be random. Most of the time, the process state is not completely known and therefore analysis of the real-drift reasons can get very tedious. For the remainder of this thesis its assumed that the drift is visible in the input data which is available in the dataset.

Table 8.1.: Different type of drifts with corresponding time behavior and possible reasons.

Type of drift	Time behavior	Reasons
Sudden / Abrupt		Usually event based, e.g.: • Roll change • Calibration • Measurement problems. • Mechanical issues (clearance)
Incremental		Random or event-based start. End behavior may or may not be limited. • Sensor problems • deterioration • incremental predictive errors (roll wear, roll crown)
Gradual (over time)		Different reasons with complex analysis • Sensors • Mechanical
Reoccurring concepts		Usually also reoccuring process state, e.g. categoricals not used or monitored. • Hidden context change (variable not in dataset or not used for prediction). • Material supplier • Furnace • Material
Outlier		Various reasons • Sensors • Insufficient predictive capabilities • Wrong assumptions on process state / product state

8.4. Online Parameter Optimization for Support Vector Regression

Detection of drift and events is important when parameters can be optimized to improve the prediction accuracy. One of the first algorithms which were capable of drift detection were developed in the late eighties of the last century by Schlimmer and Granger[SG86]. They are using ensemble based descriptions most relevant for the current context. Other methods are using only a subset of the last instances [WK96]. Usually this subset is defined by a fixed or dynamically sized window, i.e. all of the recently received samples are used[Kli04].
Other methods for the detection of drift are usually monitoring the prediction accuracy. For classification problems methods are available which are using the error probability or the distance between two errors to detect changes [JAF+06].
The described online SVR method has already different methods for selecting or discarding samples since the storage size has to be limited to a certain amount of samples. A remaining issue is the optimization of parameters which is necessary on the occurrence of drift in the data.
Therefore the implementation was extended to be able to optimize the most relevant parameters. Dependent on the application, two different implementations can be used: discrete and continuous.
The discrete optimization is triggered through an event. Because the online parameter may be changed during optimization the algorithm is not able to receive any new training data in that time. This situation reflects typical real-world scenarios, e.g. if there is a scheduled maintenance or some predictable delay in the processes. This spare time can then be used to trigger the discrete optimization. A pseudo code of the discrete optimization is shown in Algorithm 7.
This discrete optimization will cause a simple grid search for the two main SVR parameter ϵ and C. Because the optimization time should be as short as possible, it has to finish as fast as possible. For each optimization call only four different parameter variations are tested and compared to the current performance. The new parameters are determined through multiplication of the current parameters. For sake of simplicity, the actual parameter are just multiplied by 0.9 and 1.1. This means that the parameter variation is 10% in both directions. Therefore only four new optimization threads are created. In principle also more threads and combinations may be tested. Table 8.2 shows the different settings, which are tested within one optimization call. The execution of each parameter variation is done in parallel in different threads (Step 2 in Algorithm 7). For reference also the original setting is shown.
Each thread is trained with all samples of the internal storage incrementally and before training of each sample, the prediction is stored and compared with the

Algorithm 7: Discrete online SVR optimization.

```
1 let N_o be the number of total optimizations loops; for n = 1 to N_o do
2     start 4 threads with different parameters and collect results ;
3     start optimization with parameters based on regression of all RMSE results;
4     let RMSE_i determine the RMSE of thread i and i = 0 reflect the RMSE
        with current settings;
5     if RMSE_i < 0.95RMSE_0, i = 1 ··· 5 then
6         start 2 threads with new parameter set and variation of kernel
            parameter;
7         Switch parameter of main algorithm to the best performing parameter
            combination
8     else
9         stop optimization
```

Table 8.2.: Parameter variation for optimization of C and ϵ. After the performance of all variants are determined a linear regression model is build and another variant is tested.

Thread	C	ϵ
Main	C_0	ϵ_0
Thread 1	$C_1 = 1.1C_0$	$\epsilon_1 = 1.1\epsilon_0$
Thread 2	$C_2 = 0.9C_0$	$\epsilon_2 = 1.1\epsilon_0$
Thread 3	$C_3 = 1.1C_0$	$\epsilon_3 = 0.9\epsilon_0$
Thread 4	$C_4 = 0.9C_0$	$\epsilon_4 = 0.9\epsilon_0$
Main	$(C_5, \epsilon_5) = f\left(C_{0..4}, \epsilon_{0..4}, RMSE_{0..4}\right)$	
Main	$\min RMSE\left\{\{\epsilon_0, C_0\}, \cdots, \{C_5, \epsilon_5, \}\right\}$	

true value.
For comparison of the different variants the performance in terms of RMSE is used. The MAE can also be used for the internal evaluation and will yield similar results. If a thread achieves a better result than the original one the algorithm creates a 2D linear regression based on both parameters and all individual thread results . Then the maximum gradient is used to calculate an optimum parameter set which is then used and tested again (Step 3 in Algorithm 7). If the best result out of all four threads and the last parameter variation achieves a significantly better performance than the original parameter set the kernel parameter variation is tested in two variants (Step 6 in Algorithm 7). Significance can be set in terms of RMSE improvement. The parameters of the best performing variant is transferred to the main algorithm and the optimization stops.
The background optimization procedure will continuously search for better pa-

rameters in background, i.e., in another thread. Therefore, the thread will first collect some samples and validate the performance against different parameters than used in the main thread. If the background optimizer found some better parameters it will trigger an event to switch to the new settings. This is simply realized with the usage of a mutex in C++. This mutex has to be used because, for a short time, the main algorithm will not be able to receive and handle any new data.
The background optimization can be used if information regarding scheduled downtimes or delay are not available and memory and computational power is sufficient for another algorithm continuously running in a separate thread.
The supply of training samples duplicated, i.e. the background thread will receive the same samples like the main algorithm. In principle any kind of parameter optimization algorithm can be used but for sake of simplicity the current implementation also uses a grid search as described in the case of discrete optimization. The main advantage of this optimization is that the main algorithm still can be used and is not affected during determination of the best parameter. Another advantage is that it will not rely on some delay detection of the process. The pseudo code of the procedure is shown in Algorithm 8. After a sufficient amount of samples have been collected, the background optimizer starts searching for better parameters. During this search, it will not be able to incorporate any new samples. If the optimizer found parameters with higher performance, they have to be verified against the latest data only received by the main algorithm. Only if the performance on the latest data has also improved, then the parameters of the main algorithm have to be changed. Due to the fact that already a duplicate version was trained in background the transfer is very fast and should not introduce a long disturbance of the process.
A problem when using the background optimization are the synchronization calls to the main thread. If better parameters have been found then the background process will retrieve all currently stored data from the main SVR algorithm instance and feed these samples to a new instance. At some point a lock has to be set for a short period of time in order to swap parameters from background and main thread. Clearly, the background optimization will require additional resources like memory and computational power. This will cause the incremental training to be slower than without optimization. Figure 8.1 shows a comparison of training time per sample for different storage sizes with and without the background optimization. Another reason for the high training times for the background optimization is the available CPU power. Multiple threads are requesting processor resources at the same time. This will influence the main process since no prioritization of the different thread is used. Another problem which might occur when using the background optimization is determined by the number of categorical variables and the strategy to handle those. When for each combination of categorical a

Algorithm 8: Background online SVR optimization.

```
start thread with parameters of main model;
start collecting samples;
let n be the number of samples received;
if n > 10 then
    store arriving samples in buffer (stop training);
    use grid search to check for better parameter settings;
    if better parameter found then
        train algorithm with buffered samples;
        check latest performance of improved variant against main model
          performance;
        if performance improved then
            lock main model and initialize switch to new parameters;
        else
            continue grid search for improved parameter settings;
    else
        train with buffered samples and start collecting new samples;
        continue after a certain time with parameter search;
else
    continue collecting samples;
```

separate model instance is used, then the required resources and memory will quickly be too high. Consider the example of the StripForce dataset with two categorical variables, i.e. material and stand. The material has 24 different values and the stand variable four different values. This would mean, despite already 96 different instances of the algorithm are created, that also 96 separate threads will be created and will consume processor resources. Therefore, both strategies can only be used together if the number of different levels of the categorical variables are low and are assumed to be known a priori.
An example of the impact on the optimization is given in Figure 8.2 where the Width dataset is used for the comparison on cold-start and warm start optimization problems. Shown is the performance when the SVR is used with default parameters without any optimization (cold-start) and with the two online optimization methods. For comparison, also the performance of the SVR algorithm with optimized parameters on the first 200 samples is shown (warm-start). It can be seen, that the online optimization yields significant improvements in comparison to the default parameter settings. Both optimization methods achieve almost the same performance throughout the complete dataset. When samples are known, the SVR parameters can be optimized. The achieved performance is therefore much

Figure 8.1.: Training time per sample for the StripForce dataset. Three different storage sizes N_s, i.e., 200, 1 000 and 2 000 with and without the background optimization is analyzed. For all variants the oldest sample was always deleted and the RBF kernel was used.

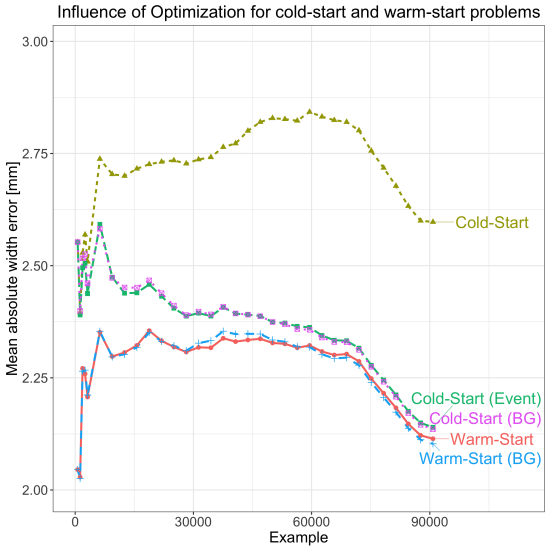


Figure 8.2.: Comparison of the MAE performance on the Width dataset for different usage of the optimization. Default SVR parameters (cold-start) without parameter optimization is compared with the performance when background optimization (BG) or optimization on event (Event) is used. Additionally, the performance for optimized parameters on the first 200 datasets is shown (warm-start).

better on the first 20 000 samples. It is interesting to see that the optimization can only marginally improve the overall performance for the width problem. In both cases, the abrupt drift and the incremental drift can be compensated. The drift is inherently compensated by integrating new samples into the SVR algorithm. The reason, why the optimization in case of the warm-start problem will only have a small impact, may lie in the bias-variance trade off which is described in detail in [JWHT14, HTF01]. Initial optimization will yield great improvement of the prediction error, but further optimization may only improve the prediction on the samples stored internally. For new samples, the prediction error might even increase.
As a consequence, the parameter optimization is highly recommended for cold-start problems. For warm-start problems, the optimization should be triggered if high dynamic changes are detected in the data.

9. Performance

After discussing the different algorithms and their parameter impact on various scenarios this chapter will now compare their performance. First, a comparison on the extrapolation behavior is made which is especially important on cold-start problems. Afterwards, the comparison on the default datasets is made before the chapter closes with the discussion about the performance achievements on the rolling dataset.

9.1. Extrapolation

An important criteria for online algorithm is their behavior for unseen areas, i.e., if they have to extrapolate. Offline algorithms will usually have enough data available and therefore, this behavior is not of great importance. If the online algorithm is used for cold-start problems, the prediction is always in regions where no prior information is available. Therefore, the algorithm has to extrapolate from regions, where initial data has been received. The question is, how the different algorithms are handling such scenarios? Extrapolation will also occur, if rare materials or rare constellations are considered. Figure 9.1 shows the prediction of all three online algorithms when they were trained in the region of $0 \leq x \leq 1$. The prediction over the whole area is very bad in all three cases. Clearly, the extrapolation with online SVR regression with RBF Kernel is not producing any prediction values much higher than the training region. The typical application scenario for the online algorithms in the hot rolling is to correct analytical models. Therefore, the assumption is that no huge errors are present. Therefore such extrapolation behavior as shown for PA-II and RLS are not desirable and have to be avoided. This might be accomplished with different strategies like a maximum correction term for regions without any data.
This chapter compares all discussed algorithms and their performance on the described datasets. Further, for the default dataset, a comparison with state-of-the-art models is made. To answer the question if online models are really needed, different offline algorithm performances are also shown.

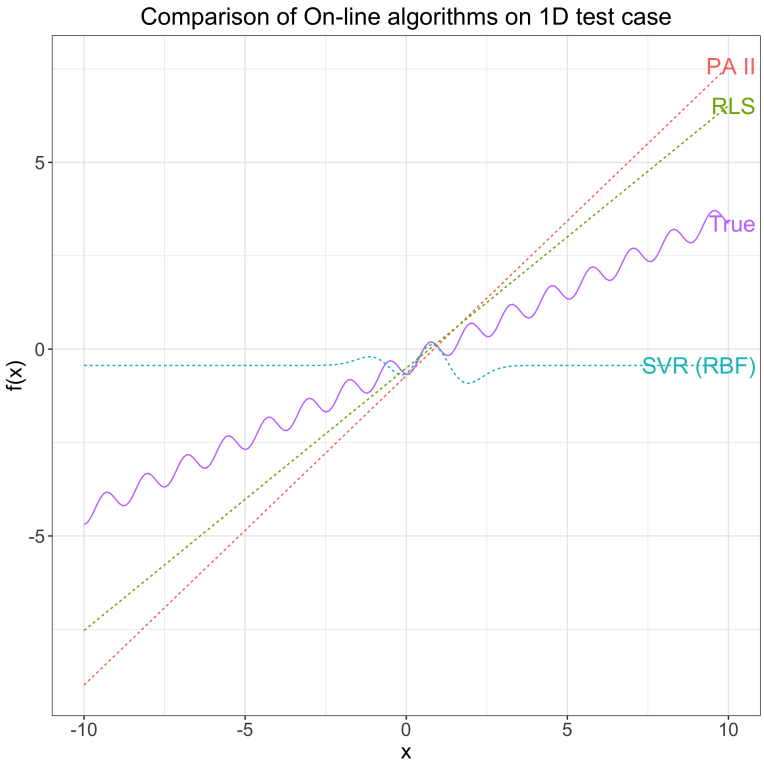


Figure 9.1.: Comparison of PA, RLS and SVR with RBF Kernel prediction. All algorithms were trained with training samples of a 1D test function $f(x) = 0.3 \sin(5 \times x - 2) + 0.4 \times (x - 1) + \sigma$ between $0 \leq x \leq 1$. The plot shows the prediction in the region of $-10 \leq x \leq 10$ and the true function (violet) for the RLS algorithm (green), PA-II (red) and SVR with RBF Kernel (cyan).

Table 9.1.: Mean squared error (MSE) for different algorithms on the default datasets. The first 3 columns contain offline algorithms for comparison which are using the first 10% of the data as training data. The following algorithms are used: Linear regression model (Lin), Random uniform forest(RUF), Support Vector Machine (SVM), Passive Aggressive in two variants (PA-I and PA-II), Recursive Least Squares (RLS), Support Vector Regression (SVR) and the best algorithm described in [Iko12] (IKON). Values marked bold achieve the best performance.

Problem	LIN	RUF	SVM	PA-I	PA-II	RLS	SVR	IKON
Abalone	8.4	7.9	**7.0**	4.9	**4.1**	4.6	4.9	5.7
Cal housing	7.5e9	**6.4e9**	9.7e9	2.6e9	**2.3e9**	3.0e9	5.6e10	5.1e9
Elevators	**1.0e-5**	2.2e-5	1.2e-5	5.3e-6	**4.9e-6**	8.9e-6	8.0e-6	2.2e-5
House 8L	1.8e9	**9.1e8**	1.2e9	2.5e9	2.3e9	1.8e9	1.5e9	**1.1e9**
House 16H	2.1e9	**1.2e8**	1.6e9	2.7e9	2.7e9	2.1e9	1.7e9	**1.6e9**
Mv delve	1.5	**2.6e-2**	0.5	8.6	6.6	1.9	**0.3**	1.7e1
Pol	9.9e2	**6.0e1**	2.7e2	2.4e3	2.1e3	1.9e3	2.5e3	**2.3e2**

9.2. Performance on Default Datasets

To compare the implemented online algorithms the default datasets described in Section 6.1 are used. In addition to the best result obtained from [Iko12] other simple algorithms are used as baseline results. To have also a comparison to offline models three simple offline algorithms are used which are trained by the first 10% of the data. The first one is a simple linear regression model (LIN) , the second one is a random uniform forest model (RUF) and the third one uses the support vector machine implemented in the e1071 package which is based on the popular libsvm (SVM) [CL11]. The tested online algorithms are the two variants of the PA algorithms, i.e. PA-I and PA-II, the RLS algorithm and the Online SVR algorithm. The performance value is taken from the mean squared error (MSE) of the complete data. This is used to directly compare the results with the results obtained in [Iko12]. Only dummy coding strategies which are suitable for online algorithms are used here. This means, that traditional dummy coding schemes, which causes the feature vector to increase, are not considered.

9.3. Performance on Rolling Datasets

The main application of the presented algorithms are analyzed in the context of the hot rolling process. The performance of the best algorithms on the corresponding rolling problem is analyzed in the following part.

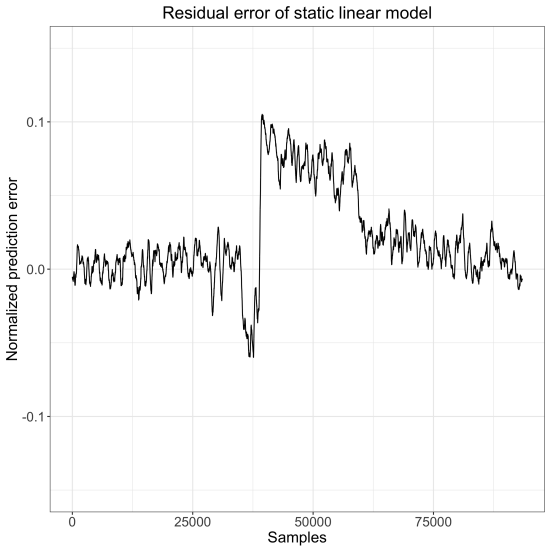


Figure 9.2.: The plot over the residuals of a static liner model is used to detect the dynamic changes like discrete events or drifts in the dataset.

Width Problem

The Width dataset is an excellent example to test algorithms on their capability to deal with drift in the data. At some point of the dataset which occurs approximately at sample 40 000 a drift in the dataset is visible. Using the different drift types presented in Table 8.1 it can easily be identified as a sudden or abrupt drift followed by an incremental drift. The later one is not easily visible on the target data itself but on the residuals of an offline linear regression model. The residuals of a linear model which was trained on the first 5 000 samples is shown in Figure 9.2. Before training, the dataset was scaled to values between 0 and 1. The abrupt drift and the incremental drift is clearly visible.
The impact of the algorithm parameters has already been shown in the corresponding sections of the algorithm. The interesting question is now, which algorithm achieves overall the best performance. To get an idea about how good the predictions are the first comparison is done based on the MAE. Figure 9.3 shows the performance of the different algorithms. For all online algorithms, only the best variant is shown. The corresponding offline variants, i.e. a linear model and a SVM model, are also shown for comparison. It clearly turns out, that there is a high dynamic within the datasets since both offline variants are the two worst algorithms. Especially around at approximately half of the dataset the performance of the linear model drops significantly. All online variants are almost

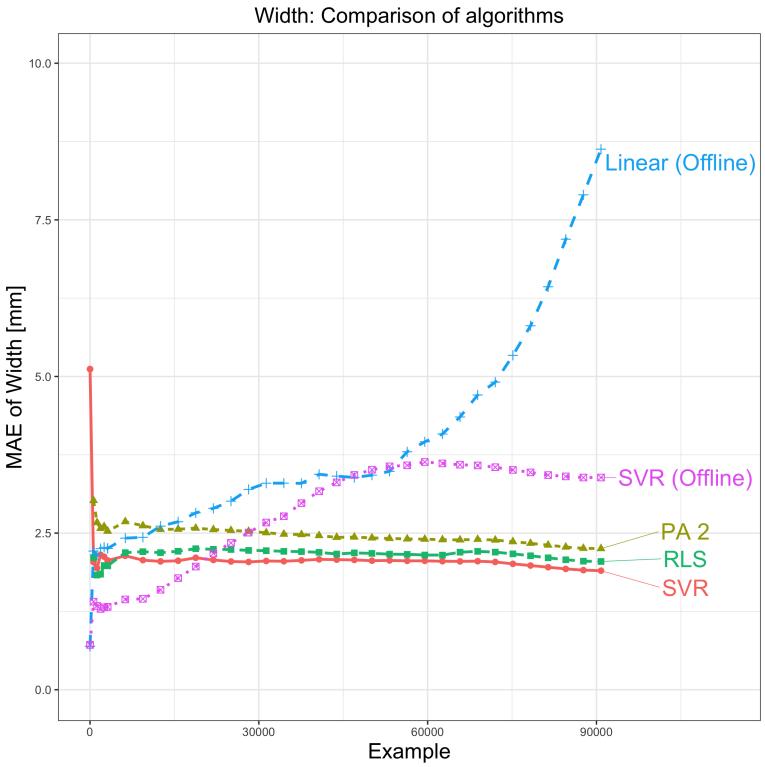


Figure 9.3.: Comparison on the best algorithm variants for the Width dataset. Shown is the mean absolute error over the whole dataset. For each algorithm variant only the best one is shown. Beside the three online algorithms PA, RLS and SVR also a linear regression model and an offline SVR algorithm is shown.

Table 9.2.: Summary of the mean absolute error (MAE) for the best algorithms over the whole dataset.

Rank	Algorithm	Details	Coding	MAE
1	SVR online	Size N_s: 1 000, Mgmt. M_s: Reserve Kernel: RBF	Numeric	1.89
2	RLS	Decay: 0.98	Remove	2.04
3	PA2	Aggressive Parameter: 0.1	Remove	2.27
4	SVR Offline	Libsvm trained on first 10%	Numeric	3.40
5	Linear Offline	Trained on first 10%	Numeric	9.50

Table 9.3.: Top 10 online SVR variants for the Width dataset.

Rank	Size N_s	Kernel	Storage Mgmt.	Coding	MAE
1	1 000	RBF	Reserve	Numeric	1.89
2	1 000	RBF	Random	Remove	1.89
3	1 000	RBF	Distance	Remove	1.92
4	1 000	RBF	Oldest	Numeric	1.93
5	1 000	RBF	Oldest	Numeric	1.94
6	500	RBF	Distance	Numeric	1.94
7	1 000	RBF	Distance	Numeric	1.95
8	1 000	RBF	Distance	Remove	1.95
9	1 000	RBF	Oldest	Remove	1.95
10	1 000	RBF	Oldest	Remove	1.96

stable over the whole dataset with SVR achieving the best performance followed by the RLS algorithm. The PA 2 variant is the best among the algorithms of class Passive Aggressive but are the worst online algorithm. Nevertheless, the result is only slightly worse than the RLS variant.
The achieved performance over the whole dataset is summarized in Table 9.2. An interesting observation is, that the dummy coding scheme is either the conversion to numerical or the removal. The categorical factor has almost no impact on the performance.
For the online SVR algorithm the best performance was achieved with minimum storage size which clearly indicates the high dynamic of the dataset. It is not necessary to store any old information. The fact that the random removal of samples achieves the best performance is also nothing special and was already discussed in [JZBBR17]. Tables 9.3 shows the top ten performance of the different online SVR variants.

It can be seen from Table 9.3 that all online SVR variants are achieving a better

performance than any of the other algorithms. Dominating kernel was the RBF kernel which was used in all of those algorithms. The best linear kernel algorithm achieves a MAE performance of 1.96 and the best polynomial kernel has a performance of 2.19, which is still better than all offline variants and also all PA variants. Clearly, the storage size of 1 000 is also dominating the best performing algorithms. Only one algorithm with smaller storage size has a similar performance. The best algorithm with storage size of 200 achieves a MAE performance of 2.04.

PlateForce Dataset

The PlateForce dataset is the only dataset for hot steel rolling. The flow curves for steel are much more established than those for aluminum. Additionally, the grouping for steel is very fine coarse in comparison to aluminum. Therefore it is assumed, that there are no huge differences between the material groups. Still the questions arises, if online algorithms can significantly improve those kind of problem. The handling of the material group as categorical variable with a potential high number of levels is further of great importance.
To answer these questions the best variants for all online algorithms and two corresponding offline algorithm are used and the MAE is plotted over the whole dataset. The result can be seen in Figure 9.4. The figure shows, that the performance of the offline SVR and the linear model can significantly be improved with the usage of their online versions. It is interesting to see, that the best PA and the best RLS algorithm have almost the same performance. After sample number 10 000, some drift in the data occurs. At this point, both offline algorithms have a tremendous decrease in performance. Either some new material was rolled at this point or there is some event-based drift occurring. Somehow, this drift can be better compensated by the RLS algorithm than the PA algorithm.
The best performance can be achieved by the online SVR algorithm which has far the best performance in terms of MAE. While the total MAE for the online SVR is $\approx$ 850 kN, the other online algorithms achieve a performance of approximately $\approx$ 2 700 kN (RLS) and $\approx$ 3 000 kN (PA-II).
Surprisingly, the offline SVR which was only trained on the first 10% of the data achieve a better performance than PA and RLS. The online algorithms achieve a more or less stable performance, whereby the performance of the offline SVR algorithms is almost continuously decreasing and is expected to further decrease. It seems that the properties of the SVM are ideally suited to deal with this kind of problem.
In general, a force deviation of 1 000 kN can be seen as a very good prediction which is a perfect prerequisite for a robust stable production with highest quality demands.

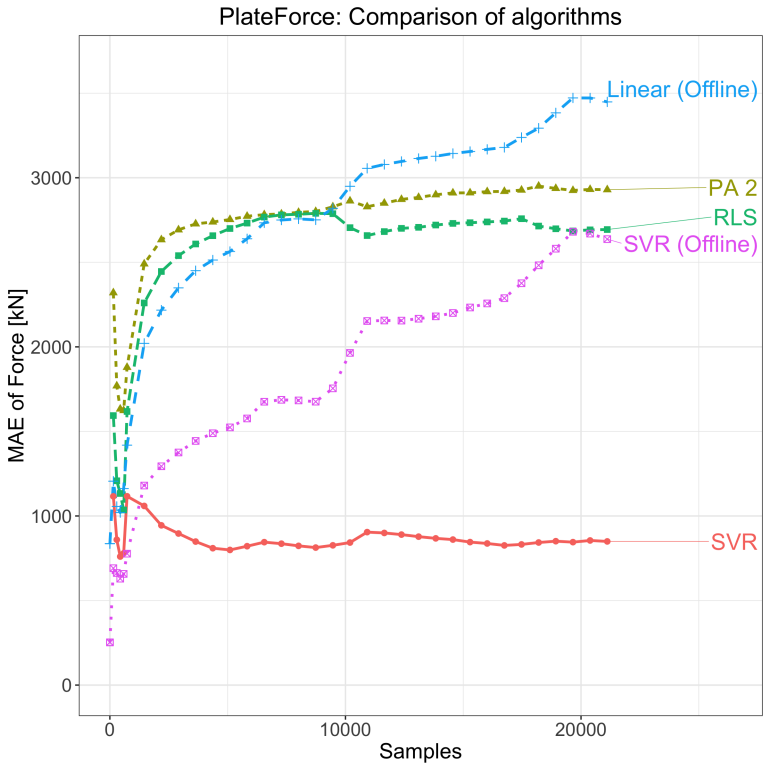


Figure 9.4.: Mean absolute error (MAE) for the PlateForce dataset. Shown is the MAE over the whole dataset for the best variants of all analyzed online algorithms, i.e. Passive Aggressive (PA), Recursive Least Squares (RLS) and online SVR. For reference and also to see whether an online algorithm is necessary two offline algorithms are also shown which were trained on the first 10% of the data.

Table 9.4.: Top 10 online SVR variants for the PlateForce dataset.

Rank	Size N_s	Kernel	Storage Mgmt. M_s	Coding	MAE
1	500	RBF	Oldest	Numeric	842
2	1 000	RBF	Reserve	Remove	851
3	1 000	RBF	Oldest	Remove	854
4	1 000	RBF	Oldest	Numeric	854
5	500	RBF	Oldest	Remove	855
6	500	RBF	Oldest	Numeric	855
7	1 000	RBF	Reserve	Numeric	862
8	1 000	RBF	Oldest	Remove	871
9	500	RBF	Oldest	Remove	874
10	1 000	RBF	Random	Remove	878

A further analysis is summarized in Table 9.4 where the top ten online SVR variants are listed with the corresponding parameter settings. The dominating kernel was again the RBF kernel. The dummy variable should be removed or converted to a numerical value and the oldest samples should be removed.

StripForce Dataset

The StripForce dataset contains data from aluminum mills where the flow curve is very sensitive to variation in the chemical composition. As already discussed, the target value is strongly correlated to the categorical variable. The comparison of the MAE performance is shown in Figure 9.5 where the best variants of all three online algorithms are compared to two offline variants, which were trained on the first 10% of the data.

It can be seen, that the performance can be significantly improved with online algorithms and the best performance was achieved with the online SVR algorithm, followed by the RLS and PA algorithm. Table 9.5 summarizes the top ten SVR algorithm variants. The best results were achieved when the categorical variables were treated separately and the RBF kernel was used. It is also interesting to see, that despite both other rolling datasets, the distance based strategy to select samples achieves the best performance.

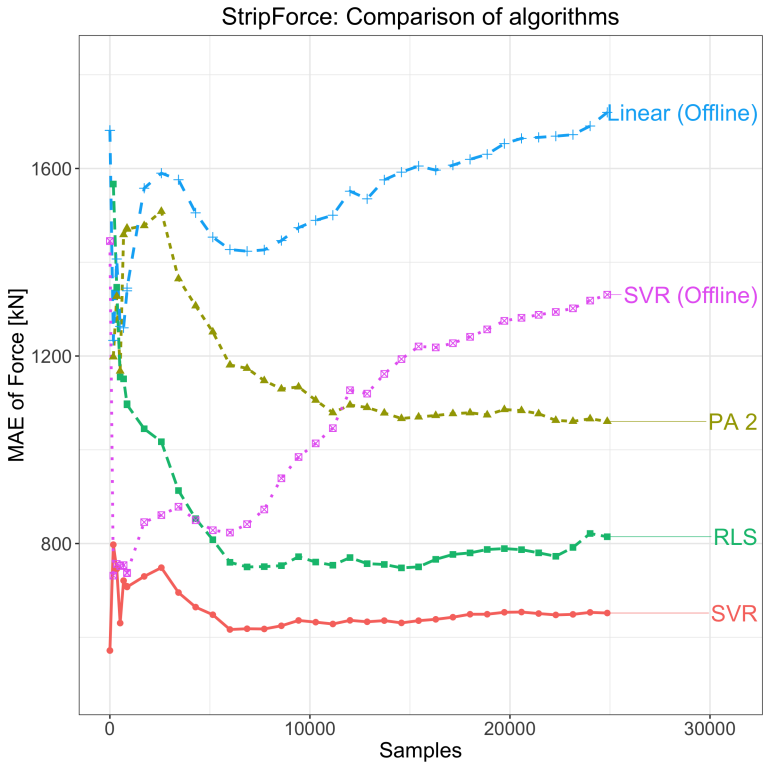


Figure 9.5.: Mean absolute error (MAE) of the StripForce over the whole dataset. The two offline algorithms, i.e. a linear model and the offline SVR were trained on the first 10% of the data for comparison. For the online algorithms only the best performance is shown.

Table 9.5.: Top 10 online SVR variants for the StripForce dataset.

Rank	Size N_s	Kernel	Storage Mgmt. M_s	Coding	MAE
1	1 000	RBF	Distance	Parallel 1	647
2	500	RBF	Distance	Parallel 1	647
3	1 000	RBF	Distance	Parallel 1	647
4	500	RBF	Distance	Parallel 1	648
5	200	RBF	Distance	Parallel 1	650
6	200	RBF	Distance	Parallel 1	653
7	1 000	RBF	Random	Parallel 1	654
8	200	RBF	Random	Parallel 1	656
9	1 000	RBF	Oldest	Parallel 1	658
10	500	RBF	Reserve	Parallel 1	658

10. Conclusion

Chapters 5-9 introduced online learning for real-world problems, focusing on hot rolling problems. It was shown that classical, offline optimization is not suitable for this kind of optimization problem. For dynamic real-world problems, online algorithms should be used. The section showed that all presented online algorithms outperform their offline variants and may improve the predictive accuracy of the process. This will ensure that the process will be more robust and is able to produce products of high quality. For all considered real-world examples the online SVR has proven to be the best among the three algorithms.
It was further shown that the algorithms are also applicable for cold-start optimization problems with some limitations. Parameters should be guessed or set prior to known values of similar problems. Since no information is available, the categorical variables should be neglected for cold-start problems. This will allow the model to learn the base problem constellations and the main effects on the error variable. With online parameter optimization a significant improvement could have been achieved and depending on the exact application the background optimization process or the event-based trigger can be used for prediction improvements.
The usage of categorical variables has a significant effect on the achievable performance of the problems. However, it is wise to use feature selection techniques to determine, if the categorical variables have an impact on the predictive accuracy at all. In general, if the categorical has an impact on the predictions using parallel model instances, i.e. either parallel model instances or just corrections are beneficial. However, if there is a high dynamic in the dataset with or without the presence of drift, categoricals may cause the learning of that drift to be much slower. In such cases, especially if an abrupt drift is present or cannot be excluded, the usage of categoricals has to be rethought.
The size of the stored samples is highly dependent on the properties of the dataset. High dynamic problems will benefit from low storage sizes. Otherwise the size should be as high as possible while keeping the learning time constraint in mind.

10.1. Recommendations for Other Real-World Problems

The algorithms presented and discussed in previous chapters are not restricted to hot rolling. They can be used on any kind of real-world problem. The following recommendations can be derived from the previous results.

- For cold start problems: Start with PA or even simpler algorithms. Also SVR with only few dependable variables can lead to reasonable results. Do not consider categorical variables directly. Use online optimization of parameters for SVR.
- Feature selection with various algorithms to reveal the most important variables. Do not use standard methods for this screening since most of them are not considering dynamic effects. Instead, use online algorithms and adequate performance measurements.
- If the problem is to reduce the residual prediction errors of other prediction models, the usage of categorical variables is highly dependent on the internal structure of those models. Using the categorical as ordinal is most often not desirable and will usually not improve the performance.
- If the problem is known to be highly dynamic, a low storage size is beneficial. Otherwise use as much storage as suitable without violating the time constraint.
- For highly dynamic data the best strategy for selecting the samples is to remove the oldest one. Otherwise the distance based strategy achieves the best result.
- If processing power, memory and time is not violated use either background optimization or the discrete events for optimization of the SVR parameter.
- The RBF kernel seems to be a very robust kernel for most of the problems. However if some data are available other kernels should also be tested and compared to the RBF kernel.

10.2. Outlook

There are still some items which could not fully been analyzed in this thesis. The optimization of kernel parameter is one of them. It might be, that polynomial of different degree and offsets might be more suitable than the polynomial used here. Further, also other kernel like the sigmoid kernel should be tested and compared to other kernels.
A problem for categorical variables is still the optimization of all parallel instances of the algorithm. In the currently presented approach only the base models were

optimized but surely also the specific models can benefit from online parameter optimization. Due to the high CPU usage, the background optimization of multiple instances is currently not recommended. A possibility could be to instantiate one background optimization process which sequentially optimizes the parameters of all instances. Then, only one additional thread would be used for optimization and computation.
Another more general problem is the variable selection which still has to be made prior to the usage of these online algorithms. A solution might be to integrate the feature selection into the background optimization process. Then, the full samples would always be stored but the algorithms would work on a subset selected by a separate process.

Part III.

Summary

11. Summary

The previous chapters described optimization in context of real-world applications with focus on hot rolling of aluminum and steel. The complex rolling process, which was introduced in Chapter 2, can highly benefit from the presented ideas and algorithms if it is combined with knowledge from process experts. Without an in-depth understanding of the process on the one-hand side and the analytical models on the other hand side the potential improvement is limited.
Data-driven surrogate based optimization, introduced in Chapter 4 is an excellent method for improving even complex processes. The method for optimization of individual analytical models was generated especially for the hot rolling context but can be transferred to any kind of complex process. The prediction process itself was included as black box. Knowledge about the internal design of the process models however was determining the requirements for the optimization framework.
The first optimization problem was using data of an aluminum roughing mill to optimize flow curve parameters for an unknown material. It was shown in Section 4.4, that a tremendous improvement can be achieved. These improvements were validated with real process data. The optimization consisted of two consecutive optimizations. After an initial parameter screening, the five most promising points were selected and used for a local optimization with a Nelder-Mead algorithm.
The second optimization example, which was discussed in Section 4.5, illustrated, that it is sufficient to use only the local optimization if the region for the initial flow curve parameters are known in advance.
Afterwards it was shown in Section 4.6, that the concept is also applicable to other mill types. To transfer the result from one mill to another, certain conditions and limitations have to be made. In general it would be good to extend the simulation to be able to cover multiple mill types with wide ranges of different product geometries to achieve the best performance.
Although the concept was applied only for flow curve parameter optimization it is suitable for any kind of prediction for the rolling process. The only restriction is, that parameters can be modified by the optimization framework. The simulation was already used to optimize the friction calculation in aluminum mills. The friction will also influence the roll force calculation and therefore first the flow curves have to be optimized. It can be seen as a logical next step if the flow curve

parameters have been optimized. As mentioned already during the introduction of the rolling process, the friction is dependent on the emulsion which is almost unique to every rolling mill.
Prediction residuals will always occur for various reasons. Most often these reasons are not easy to detect and will have a high dynamic behavior. These residuals can be reduced with help of online algorithms as demonstrated in Chapter 5. Three different online algorithms were compared with state-of-the-art algorithms on default datasets but also on data coming from different rolling mills. It was demonstrated, that all three algorithms can significantly reduce the residuals for the rolling datasets. The online algorithms were used here to improve the predictions coming from various analytical models. The most promising candidate for all rolling datasets was the online SVR algorithm, which was introduced in Section 7.3. The online SVR algorithm was extended to fit the requirements for real-world processes. The storage size of the SVR has to chosen in a way to balance between robustness and speed. It was shown in Section 7.3.5 and Chapter 9, that for high dynamic optimization problems the strategy to remove the oldest data is the best choice. Only for static problems, other distance-based strategies seems to be beneficial. The handling of categorical variables is of great importance for online processes since most of the common strategies will only work properly for offline algorithms. It turned out, that depending on the optimization problem, the usage of separate models for each unique combination of categorical variables can highly improve the performance. This was shown in Section 7.3.7. For cold-start optimization problems, i.e., when no information about the residuals is available, the categorical has to be neglected since it would only slow down the initial learning process. Another extension for the online SVR was made to enable online parameter optimization. Online parameter optimization as introduced in Chapter 8 has to be used in case of high dynamic processes and if no information about the dataset is known. Both presented optimization strategies will ensure that the performance will significantly improve. If initial samples are available then the optimization will be only beneficial for highly dynamic datasets.
Online algorithms represent a model class which are of great importance for all kind of real-world application. The presented extension to the online SVR algorithm will enable multiple future application possibilities across industrial branches. The online optimization of SVR parameters will ensure that the model will adapt as fast as possible to any kind of problems regardless of the dynamic behavior of the dataset.

List of Figures

List of Tables

Bibliography

[Agr07] A. Agresti. *An Introduction to Categorical Data Analysis*. Wiley Series in Probability and Statistics. Wiley, 2007.

[AW94] Karl Johan Astrom and Bjorn Wittenmark. *Adaptive Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1994.

[AY01] Kazuya Asano and Kazuhiro Yamamoto. Parameter estimation for the tension model in hot strip mills. *IFAC Proceedings Volumes*, 34(18):345 – 350, 2001. 10th IFAC Symposium on Automation in Mining, Mineral and Metal Processing (MMM 2001), Tokyo, Japan, 4-6 September, 2001.

[BB03] Thomas Bartz-Beielstein. Experimental Analysis of Evolution Strategies—Overview and Comprehensive Introduction. Reihe CI. SFB 531 157/03, University Dortmund, November 2003.

[BBLP05] Thomas Bartz-Beielstein, Christian Lasarczyk, and Mike Preuss. Sequential Parameter Optimization. In B McKay et al., editors, *Proceedings 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, pages 773–780, Piscataway NJ, 2005. IEEE Press.

[BBPV04] Thomas Bartz-Beielstein, Konstantinos E Parsopoulos, and Michael N Vrahatis. Design and analysis of optimization algorithms using computational statistics. *Applied Numerical Analysis and Computational Mathematics (ANACM)*, 1(2):413–433, 2004.

[BBZ12] T Bartz-Beielstein and M. Zaefferer. A gentle introduction to sequential parameter optimization. Technical Report TR 01/2012, CIplus, 2012.

[BHKP10] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604, August 2010.

[BYHB10] Salah Bouhouche, Laib Laksir Yazid, Sissaoui Hocine, and Jürgen Bast. Evaluation using online support-vector-machines and fuzzy reasoning. Application to condition monitoring of speeds rolling process. *Control Engineering Practice*, 18(9):1060 – 1068, 2010.

[CCWA03] J. Cohen, P. Cohen, S.G. West, and L.S. Aiken. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Applied Multiple Regression/ Correlation Analysis for the Behavioral Sciences. L. Erlbaum Associates, 2003.

[CDK+06] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585, December 2006.

[CDSSS03] Koby Crammer, Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pages 1229–1236, Cambridge, MA, USA, 2003. MIT Press.

[CL01] Chih-Chung Chang and Chih-Jen Lin. Training ν-support vector classifiers: Theory and algorithms. *Neural Comput.*, 13(9):2119–2147, September 2001.

[CL11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[CP01] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems (NIPS*2000)*, volume 13, 2001.

[DSP+19] Jifei Deng, Jie Sun, Wen Peng, Yaohui Hu, and Dianhua Zhang. Application of neural networks for predicting hot-rolled strip crown. *Applied Soft Computing*, 78:119 – 131, 2019.

[DT97] J. Dennis and Virginia Torczon. Managing approximation models in optimization. In N.M. Alexandrov, M.Y. Hussaini, I.C.A.S. Engineering, and L.R. Center, editors, *Multidisciplinary Design Optimization: State of the Art*, Proceedings in Applied Mathematics Series, pages 330–347. SIAM, 1997.

[EF11] Dirk Eddelbuettel and Romain François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011.

[Enc08] *The Concise Encyclopedia of Statistics*. Springer New York, New York, NY, 2008.

[FF09] Andreas Vollmer Frank Feldmann, Mark Gerdau. Adaptive setup models for cold rolling mills. `https://library.e.abb.com/public/77efb4c9677e46c5c125759100274942/43-48%201M924_ENG72dpi.pdf`, 2009. Accessed: 2019-02-18.

[FSK07] Alexander Forrester, András Sóbester, and Andy Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 463(2088):3251–3269, 2007.

[FSK08] Alexander Forrester, András Sóbester, and Andy Keane. *Engineering Design via Surrogate Modelling*. Wiley, 2008.

[GB00] V.B. Ginzburg and R. Ballas. *Flat Rolling Fundamentals*. Manufacturing Engineering and Materials Processing. Taylor & Francis, 2000.

[GE03] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[GG01] Jacek Gondzio and Andreas Grothey. Reoptimization with the primal-dual interior point method. *SIAM Journal on Optimization*, 13:842–864, 01 2001.

[GJ+10] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. `http://eigen.tuxfamily.org`, 2010.

[GKR+01] Martin Grötschel, Sven O. Krumke, Jörg Rambau, Thomas Winter, and Uwe T. Zimmermann. *Combinatorial Online Optimization in Real Time*, pages 679–704. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[Gon98] Jacek Gondzio. Warm start of the primal-dual method applied in the cutting-plane scheme. *Mathematical Programming*, 83(1):125–143, Jan 1998.

[GvB+14] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, 2014.

[H+72] Milan Hajduk et al. Effect of improper selection of the rpm of vertical and horizontal drives on balanced rolling force distribution in a universal rolling mill. *Hutnicke Listy*, 27(8):259, 1972.

[Haf16] Raphael T. Haftka. Requirements for papers focusing on new or improved global optimization algorithms. *Structural and Multidisciplinary Optimization*, 54(1):1–1, 2016.

[HCL16] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2016.

[HELR08] Manu Hietaniemi, Ulla Elsilä, Perttu Laurinen, and Juha Röning. Defect prediction in hot strip rolling using ANN and SVM. In *Tenth Scandinavian Conference on Artificial Intelligence, SCAI 2008, Stockholm, Sweden, May 26-28, 2008*, pages 44–51, 2008.

[Hin03] Rolf Hinkfoth. *Massivumformung*. Wissenschaftsverlag, Aachen, 2003.

[HMR96] C.A. Hernandez, S.F. Medina, and J. Ruiz. Modelling austenite flow curves in low alloy and microalloyed steels. *Acta Materialia*, 44(1):155 – 163, 1996.

[HNK14] U. K. Hassan, N. M. Nawi, and S. Kasim. Classify a protein domain using sigmoid support vector machine. In *2014 International Conference on Information Science Applications (ICISA)*, pages 1–4, 2014.

[HS78] Arno Hensel and Thilo Spittel. *Kraft- und Arbeitsbedarf bildsamer Formgebungsverfahren*. Verlag Grundstoffindustrie, 1978.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[HZEK72] Milan Hajduk, Milan Zidek, Jiri Elfmark, and Svatopluk Kopec. Derivation of mean values of inherent deformation resistance in hot rolling of tonnage steel. *Hutnicke Listy*, 27(8):567, 1972.

[Iko12] Elena Ikonomovska. *Algorithms for Learning Regression Trees and Ensembles on Evolving Data Streams*. PhD thesis, Jožef Stefan International Postgraduate School Ljubljana, Slovenia, 2012. `http://kt.ijs.si/elena_ikonomovska/00-disertation.pdf`.

[JAF+06] Manuel Baena-Garca Jose, Jose Del Campo Avila, Raul Fidalgo, Albert Bifet, Ricard Gavalda, and Rafael Morales bueno. Early drift detection method. `http://www.lsi.upc.edu/~abifet/EDDM.pdf`, 2006.

[JBBR18] C Jung, T Bartz-Beielstein, and G Rudolph. Extending support vector regression for on-line learning methods on real-world data. *(To appear)*, 2018.

[Jin03] Y Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, October 2003.

[Joh14] Steven G. Johnson. The nlopt nonlinear-optimization package. 2014. R package version 1.0.4.

[Joh18] Steven G. Johnson. The NLopt nonlinear-optimization package. `hhttp://ab-initio.mit.edu/nlopt`, 2018. Accessed: 2018-10-20.

[Jon01] Donald R Jones. A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization*, 21:345–383, 2001.

[JSW98] D R Jones, M Schonlau, and W J Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13:455–492, 1998.

[JWHT14] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer New York, 2014.

[JZBBR17] Christian Jung, Martin Zaefferer, Thomas Bartz-Beielstein, and Günter Rudolph. Metamodel-based optimization of hot rolling processes in the metal industry. *The International Journal of Advanced Manufacturing Technology*, 90(1):421–435, 2017.

[Kar25] T.V. Karman. Beitrag zur Theorie des Walzvorganges. *Zeitschrift für Angewandte Mathematik und Mechanik*, 5:139–141, 1925.

[Kar92] Richard M. Karp. On-line algorithms versus off-line algorithms: How much is it worth to know the future? In *Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing '92, Volume 1 - Volume I*, pages 416–429, Amsterdam, The Netherlands, The Netherlands, 1992. North-Holland Publishing Co.

[Kat17] Gauri Katiyar. Off-line handwritten character recognition system using support vector machine. *American Journal of Neural Networks and Applications*, 3:22, 01 2017.

[Kli04] Ralf Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intell. Data Anal.*, 8(3):281–300, 2004.

[Kru02] Sven O. Krumke. Online optimization: Competitive analysis and beyond. (02-25), 2002.

[KW99] R. Kopp and H. Wiegels. *Einführung in die Umformtechnik*. Verlagsgruppe Mainz, 1999.

[LGK+06] Pavel Laskov, Christian Gehl, Stefan Krüger, Klaus-Robert Müller, Kristin Bennett, and Emilio Parrado-Hern. Incremental support vector learning: Analysis, implementation and applications. *Journal of Machine Learning Research*, 7:1909–1936, 2006.

[LMZZ15] Yuan-Ming Liu, Geng-Sheng Ma, Dian-Hua Zhang, and De-Wen Zhao. Upper bound analysis of rolling force and dog-bone shape via sine function model in vertical rolling. *Journal of Materials Processing Technology*, 223:91 – 97, 2015.

[LRT01] Perttu Laurinen, Juha Röning, and Harri Tuomela. Steel slab temperature modelling using neural and bayesian networks. *Soft computing and intelligent systems for industry (SOCO/ISFI 2001). Paisley, Scotland, UK*, 2001.

[LZHM12] Jega Laxmi, Jie Zhang, Mohd Hussain, and Julian Morris. Batch-to-batch iterative learning control using updated models based on a moving window of historical data. *Procedia Engineering*, 42:206–213, 09 2012.

[Mar02] Mario Martin. On-line support vector machine regression. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Machine Learning: ECML 2002: 13th European Conference on Machine Learning Helsinki, Finland, August 19–23, 2002 Proceedings*, pages 282–294, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[Mis18] M. Miskowicz. *Event-Based Control and Signal Processing*. Embedded Systems. CRC Press, 2018.

[MRS+09] Sumantra Mandal, V. Rakesh, P.V. Sivaprasad, S. Venugopal, and K.V. Kasiviswanathan. Constitutive equations to predict high temperature flow stress in a ti-modified austenitic stainless steel. *Materials Science and Engineering: A*, 500(1):114 – 121, 2009.

[MTP03] Junshui Ma, James Theiler, and Simon Perkins. Accurate on-line support vector regression. *Neural computation*, 15:2683–2703, 2003.

[NM65] J A Nelder and R Mead. A Simplex Method for Function Minimization. *Computer Journal*, 7:308–313, 1965.

[OANY80] M Okada, T Ariizumi, Y Noma, and Y Yamazaki. On the behavior of edge rolling in hot strip mills. In *International Conference on Steel Rolling*, volume 1, pages 275–286, 1980.

[OJB11] Olufemi Omitaomu, Myong Jeong, and Adedeji Badiru. Online support vector regression with varying parameters for time-dependent data. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41:191 – 197, 2011.

[OMBH07] Olufemi Omitaomu, Jeong Myong, Adedeji Badiru, and J. Hines. Online support vector regression approach for the monitoring of motor shaft misalignment and feedwater flow rate. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37:962 – 970, 2007.

[ÖÖ00] Can Özsoy and Ertan Öznergiz. A self-tuning thickness control in plate hot-rolling. *IFAC Proceedings Volumes*, 33(22):233 – 238, 2000. IFAC Workshop on Future Trends in Automation of the Mineral and Metal Processing (MM 2000), Finland.

[Pow88] M J D Powell. A Review of Algorithms for Nonlinear Equations and Unconstrained Optimization. In *Proceedings ICIAM*, pages 220–232, 1988.

[Pow94] M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In Susana Gomez and Jean-Pierre Hennart, editors, *Advances in Optimization and Numerical Analysis*, pages 51–67, Dordrecht, 1994. Springer Netherlands.

[PPP17] Kedar Potdar, Taher Pardawala, and Chinmay Pai. A comparative study of categorical variable encoding techniques for neural network classifiers. *International Journal of Computer Applications*, 175:7–9, 2017.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C (2nd Ed.): The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[R C17] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017.

[RH95] Brian D. Ripley and N. L. Hjort. *Pattern Recognition and Neural*

Networks. Cambridge University Press, New York, NY, USA, 1st edition, 1995.

[Row90] Thomas Harvey Rowan. *Functional Stability Analysis of Numerical Algorithms*. PhD thesis, Austin, TX, USA, 1990. UMI Order No. GAX90-31702.

[RPS17] Matruprasad Rout, Surjya Kanta Pal, and Shiv Brat Singh. Finite element analysis of cross rolling on aisi 304 stainless steel: Prediction of stress and strain fields. *Journal of The Institution of Engineers (India): Series C*, 98(1):27–35, 2017.

[RZD12] R. Ren, X. P. Zhang, and X. Z. Du. FEM Simulation of the Slab Edge Rolling in Unsteady Rolling Stage. *Applied Mechanics and Materials*, 152:297–300, January 2012.

[SG86] Jeffrey C. Schlimmer and Richard H. Granger. Incremental learning from noisy data. *Machine Learning*, 1(3):317–354, 1986.

[SK96] Dirk FJ Staalman and Arend Kusters. On-line slab temperature calculation and control. *Manufacturing Science and Engineering*, 4:307–314, 1996.

[SK08] Sudipta Sikdar and Sabita Kumari. Neural network model of the profile of hot-rolled strip. *International Journal of Advanced Manufacturing Technology*, 42:450–462, 2008.

[SLSJ02] Amnon Shirizly, J. Lenard, J. Sauer, and K. Januszkiewicz. Lubricant capture during hot rolling of an aluminum alloy. *Tribology Transactions - TRIBOL TRANS*, 45:205–210, 04 2002.

[Søn03] Jacob Søndergaard. *Optimization Using Surrogate Models—by the Space Mapping Technique*. PhD thesis, Technical University of Denmark, Richard Petersens Plads, DK-2800 Kgs. Lyngby, 2003.

[SS04] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[SS12] Shai Shalev-Shwartz. Online learning and online convex optimization. *Found. Trends Mach. Learn.*, 4(2):107–194, 2012.

[SS16] Marlene Spittel and Thilo Spittel. Influence of chemical composition and forming conditions on flow stress. In Hans Warlimont, editor, *Part 3: Non-ferrous Alloys - Heavy Metals: Subvolume C: Metal Forming Data - Volume 2: Materials - Group VIII:Advanced Materi-*

als and Technologies - Landolt-Börnstein New Series, pages 18–64, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[SSWB00] Bernhard Schölkopf, Alex J. Smola, Robert C. Williamson, and Peter L. Bartlett. New support vector algorithms. *Neural Comput.*, 12(5):1207–1245, 2000.

[ST17] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017.

[Str10] Gilbert Strang. *Wissenschaftliches Rechnen*. Springer, Berlin, Heidelberg, 2010.

[SWMW89] J Sacks, W J Welch, T J Mitchell, and H P Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 1989.

[TC14] R. R. Draxler T. Chai. Root mean square error (rmse) or mean absolute error (mae)? – arguments against avoiding rmse in the literature. hhttps://www.geosci-model-dev.net/7/1247/2014/gmd-7-1247-2014.pdf, 2014.

[THW+83] Masahiro Takeuchi, M Hoshiya, K Watanabe, O Hirata, T Kikuma, and S Sadahiro. Heavy width reduction rolling of slabs. *Nippon steel technical report. Overseas*, (21):235–246, 1983.

[TL03] David M. J. Tax and Pavel Laskov. Online SVM learning: From classification to data description and back. In *Proc. Neural Network and Signal Processing*, pages 499–508, 2003.

[TLWM13] S. Tang, Z.Y. Liu, G.D. Wang, and R.D.K. Misra. Microstructural evolution and mechanical properties of high strength microalloyed steels: Ultra fast cooling (ufc) versus accelerated cooling (acc). *Materials Science and Engineering: A*, 580:257 – 265, 2013.

[TNR81] A.I. Tselikov, G.S. Nikitin, and S.E. Rokotyan. *The Theory of Lengthwise Rolling*. Mir Publishers, 1981.

[TOY+82] Hiromitsu Takei, Yoshihiro Onishi, Yoshimasa Yamasaki, Atsuhisa Takekoshi, Masaharu Yamamoto, and Masaru Okado. Automatic width control of rougher in hot strip mill. Nippon Kokan Technical Report 34, Computer Systems Development Department Fukuyama Works, 1982.

[TWW+18] Y. Tian, H. T. Wang, Z. D. Wang, R. D. K. Misra, and G. D.

Wang. Microstructural evolution and the precipitation behavior in x90 linepipe steel during isothermal processing. *Journal of Materials Engineering and Performance*, 27(4):1494–1504, 2018.

[Vap95] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[VGS97] Vladimir Vapnik, Steven E. Golowich, and Alex J. Smola. Support vector method for function approximation, regression estimation and signal processing. In Michael Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems 9 — Proceedings of the 1996 Neural Information Processing Systems Conference (NIPS 1996)*, pages 281–287, Dever, CO, USA, December 1997. MIT Press, Cambridge, MA, USA.

[VL63] V Vapnik and A Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.

[Web73] Karheinz Weber. *Grundlagen des Bandwalzens*. VEB Deutscher Verlag fuer Grundstoffindustrie, Leipzig, 1973.

[Wen04] Craig Wendorf. Primer on multiple regression coding: Common forms and the additional case of repeated contrasts. *Understanding Statistics*, 3:47–57, 2004.

[WK96] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

[WM97] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[WSDQS98] Harald Wehage, Ulrich Skoda-Dopp, Uwe Quitmann, and Wolfgang Sauer. Beispiele für die Simulation von Stichplänen für das Warmflachwalzen. *Stahl und Eisen*, 11:103–111, 1998.

[WZZ16] Xiaoshu Wang, Zhijun Zhang, and Peng Zhang. Development and technology of large thickness tmcp steel plate with 390mpa grade used for engineering machinery. In *HSLA Steels 2015, Microalloying 2015 & Offshore Engineering Steels 2015*, pages 961–966, Cham, 2016. Springer International Publishing.

[YL08] Jue Zhong Y.C. Lin, Ming-Song Chen. Prediction of 42crmo steel

flow stress at high temperature and strain rate. *Mechanics Research Communications*, 35:142–150, 2008.

[You14] Peter C. Young. *Recursive Estimation and Time-Series Analysis: An Introduction for the Student and Practitioner*. Springer Publishing Company, Incorporated, 2nd edition, 2014.

[YS16] Hirohisa Kawamura Hiroyuki Yoshino Ryuji Yamamoto Shigeru Ogawa Keiji Oogushi Yoshihiro Serizawa, Yasuyuki Takamachi. Heat transfer technology for steel rolling process. `http://www.nssmc.com/en/tech/report/nssmc/pdf/111-14.pdf`, 2016. Accessed: 2019-02-13.

[Zei12] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[Zhi10] Leonid V. Zhigilei. Introduction to the science and engineering of materials - chapter 6. `http://people.virginia.edu/~lz2n/mse209/Chapter6.pdf`, 2010. Accessed: 2019-02-13.