Incremental Support Vector Regression for Steering Hot Rolling Mills

Diplomarbeit
im Rahmen des Diplomstudienganges Informatik
eingereicht von

Sascha Klement

Ausgegeben und betreut von

Prof. Dr. rer. nat. Thomas Martinetz
Institut für Neuro- und Bioinformatik, Universität zu Lübeck

Lübeck, November 2006

Zusammenfassung

Maschinelle Lernkonzepte, wie die Support Vector Machine (SVM), sind mittlerweile theoretisch sehr gut verstanden und mathematisch fundiert; in industriellen Anwendungen werden jedoch häufig eher konservative Verfahren bevorzugt. Um zu zeigen, dass maschinelle Lernkonzepte Planung und Steuerung von industriellen Abläufen signifikant verbessern und vereinfachen können, wurde eine Kooperation mit der stahlverarbeitenden Industrie geschlossen. Ziel war die Konstruktion eines Stichplanrechners für Warmwalzstraßen. Der Kern des Stichplanrechners besteht aus dem MinOver-Algorithmus für Regression, einem Trainingsverfahren für die SVM, um physikalische Abhängigkeiten zu beschreiben, für die keine exakten Formeln bekannt sind. So sollen Vorhersagen über die zu erwartenden Walzkräfte und Dickenabnahmen während eines Walzvorganges getroffen werden.

Um verschiedene Arten von Vorwissen zu integrieren, wurden mehrere Approximationsebenen eingeführt und MinOver erweitert, um gewichtete Datenpunkte verwenden zu können. Desweiteren wurde die Laufzeit von MinOver für Regression durch Kernel Caching und eine optimierte Auswertung der Regressionsfunktion entscheidend verbessert. Durch die Verwendung bekannter Verfahren zur Parameterselektion und Validierung werden Fehlerraten für die Vorhersagen erzielt, die nur gering von der zu erwartenden Genauigkeit, bedingt durch die Sensorik, abweichen.

Ein Prototyp des Stichplanrechners wurde implementiert, getestet und bei der Buderus Edelstahl GmbH als dauerhafter Ersatz für eine vorhandene Stichplandatenbank installiert. Erste Betriebserfahrungen sind vielversprechend und belegen die angekündigten Verbesserungen und Vereinfachungen durch die Anwendung maschinellen Lernens.

Abstract

Machine learning concepts, such as the support vector machine (SVM), have been theoretically discussed in great detail, but in industrial applications still rather old-fashioned solutions are preferred. To show that learning concepts improve industrial planning, scheduling, and steering processes, a cooperation with a steel manufacturer was initiated to build a prototype pass schedule calculator for a hot rolling mill. The core of this scheduler consists of MinOver for regression estimation, an SVM training algorithm to learn physical dependencies for which no exact equations exist — such as the computation of rolling force or thickness reduction.

To incorporate different types of prior knowledge, MinOver is enhanced to deal with weighted data points and two approximation layers are introduced. Furthermore, the runtime of MinOver for regression is improved by kernel caching and an optimsed evaluation of the regression function. Using common parameter selection and validation methods results in error rates close to the theoretically expectable minimum.

The prototype was implemented, tested and installed at a hot rolling mill of Buderus Edelstahl GmbH as a permanent substitution for an old pass schedule database. First experiences document the promised improvements and simplifications due to the usage of machine learning concepts.

Statement of Originality

The work presented in this thesis is, to the best of my knowledge and belief, original, except as acknowledged in the text. The material has not been submitted, either in whole or in part, for a degree at this or any other university.

Ich versichere an Eides statt durch meine Unterschrift, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe. Darüber hinaus habe ich mich keiner anderen als der angegebenen Literatur oder sonstiger Hilfsmittel bedient. Die Arbeit hat in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Lübeck, November 2006

Acknowledgements

First, I would like to thank my supervisor, Prof. Thomas Martinetz, for his encouragement, faith in my work, and for his famous phone call during a lecture — the beginning of this project.

I would like to thank the iba AG, Fürth, Germany for supporting this work in many ways, especially Horst Anhaus for initiating and promoting the project, Günter Sörgel for his various advice and ideas concerning neural networks, as well as Thomas George for developing the pass schedule framework.

I would also like to thank the staff at Buderus Edelstahl Band GmbH, Wetzlar, Germany for kind provision of data, metallurgical expertise, and their objective to apply the results of this thesis in practice.

Thanks go to Michael Dorr for proofreading, especially for commas, and for his eloquent suggestions and advice to clarify and polish the contents of this thesis.

Thanks to Fabian Timm for our numerous discussions on machine learning, validation methods, and shortcomings of LaTeX, Matlab, and Microsoft Windows.

Finally, my biggest thanks go to Johanna for her continuous support, motivation, patience, and love — even in times when I do not deserve it.

Contents

Chapter 1

Introduction

Neural networks have been used for more than 15 years in steel manufacturing, especially to control electric arc furnaces or hot rolling mills [18, 13]. Experience with these neural networks showed that machine learning concepts significantly improve the accuracy of control systems. All these systems have in common that a set of sensory inputs is used to predict certain other values. Since the physical dependencies are not fully understood, the system learns its behaviour by a number of examples. Some systems only get trained once by *offline-training*, others are able to continuously adapt by *online-training*.

Another application is not only to control the system from one step to the next, but to plan a longer sequence of tasks. The idea of using neural networks for scheduling hot rolling mills to reduce the number of processing steps was formulated at the beginning of a cooperation between Buderus Edelstahl Band GmbH, iba AG and the Institute for Neuro- and Bioinformatics, Lübeck (INB).

Buderus, one of the major steel producing companies in Germany, aims to simplify and automate the generation of pass schedules for their blooming mill — a special kind of hot rolling mill.

The iba AG designs individual solutions for automating steel works and wants to improve their systems by the ability to learn from recorded data.

Finally, the INB intends to verify the advantages of the support vec-

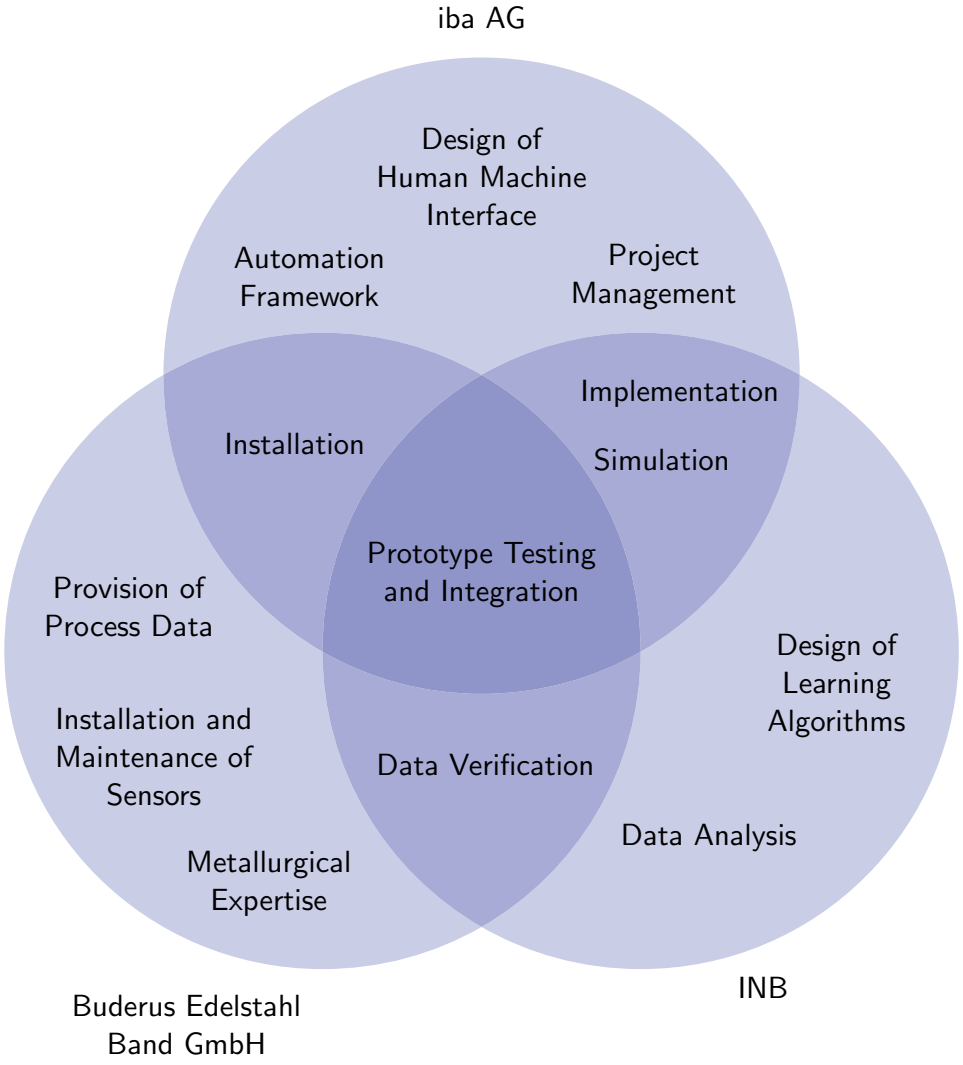


Figure 1.1: Responsibility assignment.

tor machine for learning tasks and to develop new training methods to be used in practice. This includes an in-depth analysis of the available training data provided by Buderus, as well as the incorporation of prior knowledge to the machine learning algorithm. The various sources of defect influence the sensors at the hot rolling mill — very high temperature, unpredictable vibrations and various kinds of contamination — and require extremely robust methods for separating valid from invalid data and for predicting values.

The different responsibilities and tasks (see fig. 1.1) were scheduled so that within six months after the project kickoff a prototype of the system could be installed at Buderus.

The applied methods, techniques and algorithms, as well as the main findings and suggestions for further work will be presented in this thesis. A thorough problem description will follow — starting with an introduction to longitudinal rolling and to the individual processing steps at the hot rolling mill. Then, the concept of learning from examples with its mathematical foundations in optimisation and regularisation theory will be presented. Afterwards, the support vector machine — a state-of-the-art approach to solve various learning tasks — and the numerous ways to train this machine for regression estimation will be described in detail. This includes methods for parameter selection, validation, and runtime improvements. The presented techniques will be applied to process data acquired at the hot rolling mill of Buderus to evaluate the prediction capabilities of the support vector machine in this special implementation.

Finally, the results and improvements that were achieved at the rolling mill will be discussed and some ideas to further enhance the support vector machine will be mentioned.

Chapter 2

Problem Description

Steel is a general term for a large number of metal alloys that consist primarily of iron and various types of alloying materials — such as carbon, chromium, or vanadium. These alloying materials are chosen to increase hardness, elasticity, ductility, or tensile strength of the finished product to apply it in such diverse fields such as cable-stayed bridges, engine blocks, or surgical instruments. On the way from molten steel to the finished product, various semi-finished products are manufactured.

In this chapter, the process of rolling a solid steel *slab* into a long *bloom* by a *hot rolling mill* will be covered in detail. First, the hot rolling mill and its role in the steel production process will be explained. Then, an introduction to longitudinal rolling in general and its scheduling in particular will follow. Finally, the idea of using adaptive systems for this scheduling process will be elucidated.

2.1 The Hot Rolling Mill

The molten steel, heated by an electric arc furnace, is cast into conical moulds of different geometries (see fig. 2.1). The solidified ingots will be called *slabs* throughout this thesis but other terms are equally used in practice, many of them limited to one single steel processing company. The slabs are temporarily stored in a slab yard and will be reheated to around 1250 °C before further processing. Within a few minutes after the reheating, they are

Format	Head (H) [mm]	Foot (F) [mm]	Length (L) [mm]
A28	475	376	1980
A34	516	434	2000
A41	535	415	2350

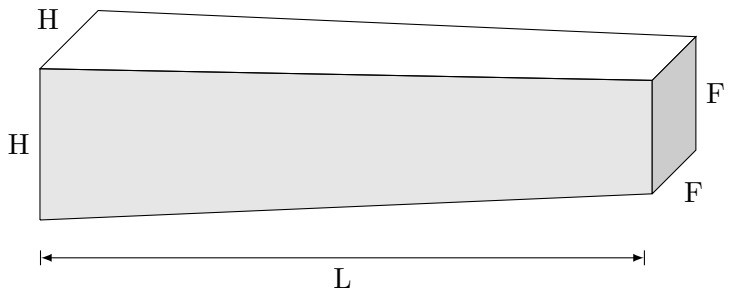


Figure 2.1: Workpiece dimensions.

transported to the blooming train — a hot rolling mill that is used for very large steel dimensions. There, thickness and width of the slab are reduced by a series of *passes* forth and back between the rolls. As volume stays nearly the same, the length will increase simultaneously with each pass.

Since the ends of the slab may contain rolling defects or contaminations from the casting process, they are cut off by a *crop shear* after the rolling process is terminated. This *semi-finished* steel — now called a *bloom* — is further processed by a finishing train or directly shipped to the customer (see fig. 2.2). These blooms have a length of up to 20 m, whereas thickness and width range between 100 mm and 400 mm.

The blooming train itself houses two reversing rolls within a stand. The rolls are cylinders with a number of *grooves* — different in shape, width, and depth — to stabilise the slab and limit its spread during a pass (see fig. 2.3). The distance between the two rolls — the *roll clearance* — determines the exit thickness of the slab within each pass. In front of and behind the rolls, *side guides* move the slab laterally between the grooves. Additionally, a *tilting device* in front of the stand tilts the slab after a certain number of passes to ensure a proper ratio of width and thickness. Side guides, tilting

device, roll clearance and the engine of the rolls are controlled and surveyed via the *human-machine interface (HMI)*.

2.2 The Pass Schedule

Whenever a slab is pulled from the reheat furnace, the monitoring system identifies the slab, determines its geometry, goal dimensions, and material, selects a *pass schedule* from a database and displays the information to the operator via the HMI. The pass schedule exactly defines the intended intermediate geometries after each single pass, as well as the groove number, roll clearance and rolling speed to achieve these geometries (see fig. 2.4).

As the slab spreads, it may not fit into the current groove anymore. If this happens behind the rolls an *empty pass* is necessary to transport the slab

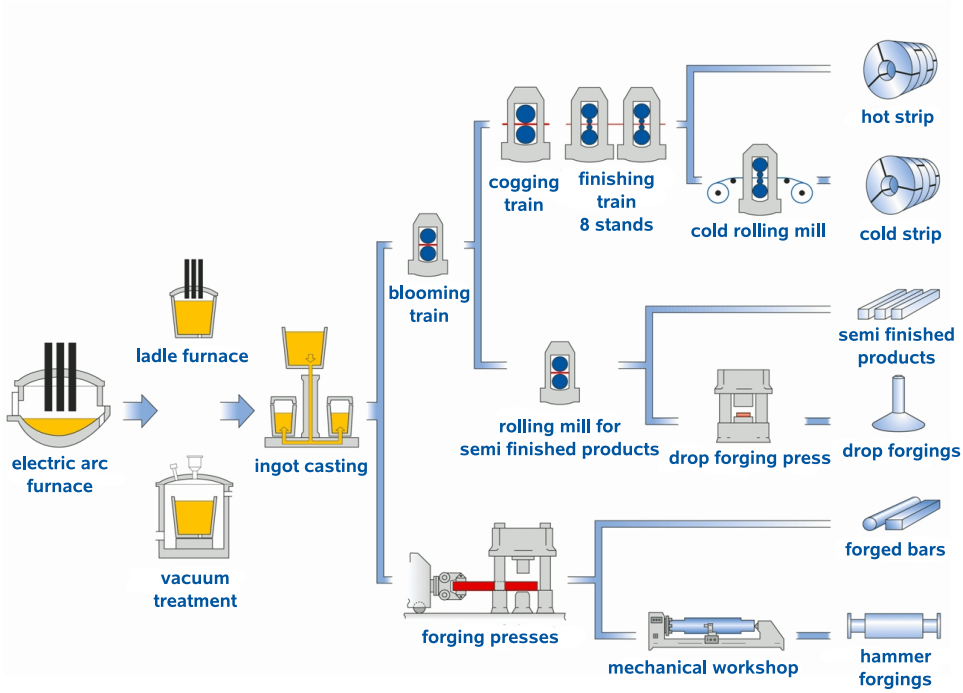


Figure 2.2: Production steps and finished products. The blooming train is the first processing step after the ingot casting, necessary for a wide range of products (Source: Buderus, slightly modified).

to the front of the rolls, where it is tilted or moved to a different groove. By installation of a second tilting device behind the rolls, empty passes would be avoided and processing would decrease.

The pass schedule database contains hundreds of such individually conceptualised pass schedules — one for each designated slab geometry. To ensure a proper rolling process without damaging the rolls, the pass schedules have to satisfy numerous more or less specific constraints. A short introduction to the physics of longitudinal rolling is given to understand these constraints.

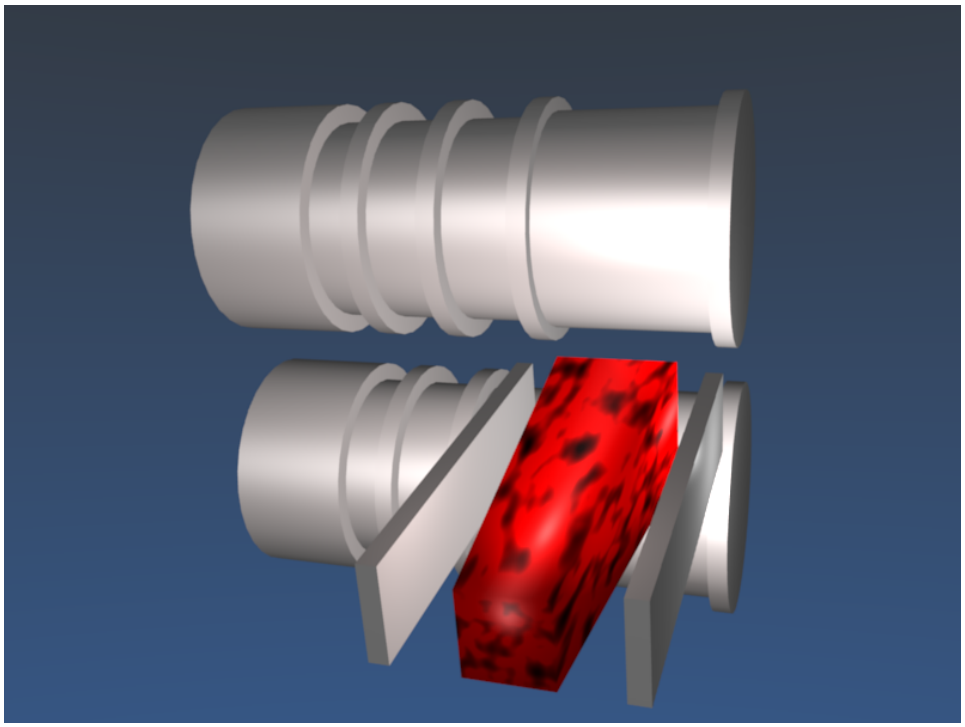


Figure 2.3: The reversing rolls before the first pass. The rolls contain five grooves numbered from right to left — four box-shaped grooves and one flat groove at the left end of the rolls. The slab gets rolled by a series of passes starting in the first groove. Small rolls (not depicted) push the slab forward while the rolls drag it through the roll gap. As the slap looses its conical shape and gets longer and longer, the smaller grooves are used for rolling. Side guides move the slab laterally and stabilise it.

2.3 Basics of Longitudinal Rolling

Rolling with two cylindrical rolls pivoted in parallel within a certain distance is the most general case of longitudinal rolling and will be used for a simplified explanation of the forming operation and the occuring forces in the blooming train. The basic terms needed for explanation are displayed

Pass schedule identifier: 475475.360135
Entry geometry: 475 x 475
Exit geometry: 360 x 135

Pass	Groove	Roll clearance [mm]	Exit thickness [mm]		Exit width [mm]	Tilting	Velocity [m/s]
1	1	323	415	x	488		4.0
2	1	263	355	x	502	1	4.0
3	1	333	425	x	371		3.8
4	1	273	365	x	385	1	4.0
5	1	213	305	x	384		3.8
6	1	178	270	x	393	1	4.3
7	2	215	350	x	280		4.0
8	2	265	350	x	280	1	4.5
9	1	118	210	x	370		3.9
10	1	63	155	x	388	1	4.1
11	4	215	350	x	164		4.0
12	4	265	350	x	164	1	4.5
13	5	125	135	x	360		4.9
14	5	175	135	x	360		5.3

Figure 2.4: Example of a pass schedule. The schedule identifier contains information about the slab's geometry before the first and after the last pass. As each groove — except the last one — has a certain depth, the exit thickness is usually larger than the roll clearance. The passes 8, 12, and 14 are empty passes as thickness is not reduced.

in fig. 2.5.

In order to pull a workpiece through the *roll gap*, the rolls must first bite the workpiece and then drag it through the gap. The *biting conditions* are fulfilled if the *roll bite angle* remains below a threshold depending on temperature, material and roughness of rolls and workpiece, and the rolling speed.

During each pass the geometry of the workpiece changes, as it is forced to move through the roll gap. The continuity theorem states that the product

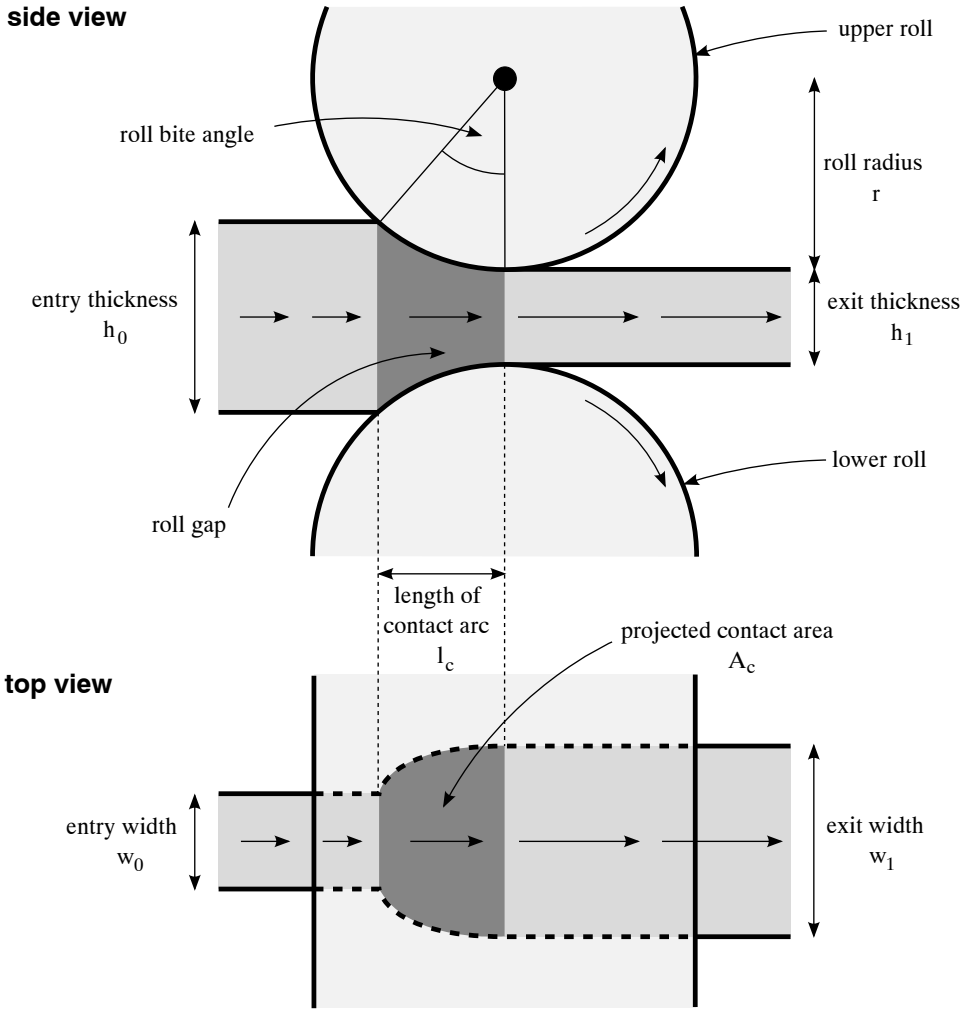


Figure 2.5: Terms around the deformation zone.

of cross section and material velocity remains constant:

$$\begin{aligned} & \text{cross section} \times \text{velocity} \\ = \; & \text{width} \times \text{height} \times \text{velocity} \\ = \; & w \cdot h \cdot v \\ = \; & \text{const} \end{aligned}$$

Thus, as thickness decreases across the *length of contact arc* l_c, material velocity will increase (depicted by horizontal arrows of different length in fig. 2.5) and the workpiece will spread, i.e. the exit width w_1 will be larger than the entry width w_0. The *spread* depends on numerous geometric factors such as thickness reduction, roll diameter, aspect ratio of the workpiece, and ratio of width to length of contact arc. Also, physical parameters as temperature, velocity, material and roughness of the workpiece affect the spread. Rolling with *bounded spread* means limiting the spread by using a groove that is not much wider than the workpiece, so that the sides of the groove constrain the geometry. The opposite of bounded spread, when the groove does not get totally filled, is called *free spread*. In practice, the transition from free to bounded spread is smooth and depends on many material parameters.

The *rolling force* F, needed for shaping within the roll gap, depends on the *deformation resistance* k_r and the projected contact area A_c:

$$F = k_r \cdot A_c \tag{2.1}$$

The deformation resistance changes across the length of contact arc, so that the *mean deformation resistance* k_{rm} is used instead of k_r — again depending on material, temperature, and speed of the workpiece as well as its geometry. Exceeding a certain rolling force would increase the wear of the rolls and, in the worst case, would lead to roll fracture.

To drag the workpiece through the roll gap, sufficient *rolling torque* τ must be available. The total torque needed for the deformation can be calculated by

$$\tau = 2 \cdot F \cdot a \cdot l_c \tag{2.2}$$

where the *lever-arm coefficient* a can be approximated to 0.5. A more accurate approximation can be found in [6]:

$$a = \begin{cases} 0.385 & \text{if } \frac{r}{h_1} > 25 \\ 0.78 + 0.017 \cdot \frac{r}{h_1} - 0.163 \cdot \sqrt{\frac{r}{h_1}} & \text{otherwise} \end{cases}$$

2.4 The Optimisation Problem

Physical reasons derived from the previous section and practical issues lead to the following constraints when generating a pass schedule:

1. The rolling force F has to be kept below F_{max} in order to limit the wear of the rolls and to prevent roll fractures.

2. The rolling torque τ must be limited by τ_{max}, otherwise the engine of the rolls could not afford enough torque to drag the workpiece through the roll gap.

3. The thickness reduction or draught per pass Δh is limited to Δh_{max} as for larger geometries the biting conditions could no longer be satisfied and the rolls would not bite the workpiece.

4. In order to fit into the groove dimensions, the entry width w_0 has to stay below the groove width w_{groove} of the current groove. Experiences show that keeping w_0/w_{groove} around 95% leads to best results.

5. Also the ratio of h_0/w_0 plays a major role. If it were too large the workpiece might unintentionally tilt — if it were too small later tilting would not be feasible any more. Thus, this ratio must be kept within a certain range ($c_{min} < h_0/w_0 < c_{max}$).

6. The shape of the workpiece also depends on the ratios w_0/h_0 and l_c/h_0 when getting rolled with free spread. If the ratios are too large, bulges along the sides are shaped; constrictions occur, if they are too small. Both artifacts lower the quality of the product and have to be avoided by constraining the ratios to an optimal range ($(w_0/h_0, l_c/h_0) \in S_{opt}$).

Identifier	Usage
$g(p_i)$	groove number
$m_0(p_i) = (h_0(p_i), w_0(p_i))$	entry geometry (entry thickness and width)
$m_1(p_i) = (h_1(p_i), w_1(p_i))$	exit geometry (exit thickness and width)
$t(p_i)$	indicates a tilting operation after the pass
$w_{groove}(p_i)$	groove width
$d_{groove}(p_i)$	groove depth
$d(p_i)$	roll clearance
d_{spring}	roll spring
$\Delta h(p_i) = h_0(p_i) - h_1(p_i)$	expected thickness reduction
$F(p_i)$	expected rolling force
$\tau(p_i)$	expected rolling torque
$w_1(p_i)$	expected exit width

Figure 2.6: Used function identifiers. Each function depends on the pass number p_i.

Now, the aim is to design a feasible pass schedule for transforming a certain slab geometry into another with the least number of passes while fulfilling all constraints. More formally, the pass schedule is a series of vectors $P = \{p_1, p_2, \ldots, p_{n(P)}\}$ with one vector for each pass. Given the entry geometry $m_{entry} = m_0(p_1)$ and the exit geometry $m_{exit} = m_1(p_{n(P)})$ the task is to solve the following optimisation problem (the function identifiers were chosen according to figures 2.5 and 2.6):

$$
\begin{array}{ll}
\text{Minimise} & n(P) \\
\text{subject to} & \left\{ \begin{array}{lcll}
m_0(p_1) = m_{entry} & & & \text{entry geometry} \\
m_1(p_{n(P)}) = m_{exit} & & & \text{exit geometry} \\
m_0(p_{i+1}) & = & m_1(p_i) & \\
i & \in & \{j | t(p_j) = 0\} & \text{succeeding geometries} \\
m_0(p_{i+1}) & = & (w_1(p_i), h_1(p_i)) & \\
i & \in & \{j | t(p_j) = 1\} &
\end{array} \right.
\end{array}
$$

$$
\text{subject to} \quad \begin{cases}
t(p_i) = 0 \quad \forall i \in \{1, 3, 5, \ldots\} & \text{tilting} \\
h_1(p_i) = d(p_i) \quad +2 \cdot d_{groove}(p_i) \quad +d_{spring}(p_i) \quad \forall i & \text{exit thickness} \\
0 \leq F(p_i) \leq F_{max} \quad \forall i & \text{force constraints} \\
0 \leq \tau(p_i) \leq \tau_{max} \quad \forall i & \text{torque constraints} \\
0 \leq \Delta h(p_i) \leq \Delta h_{max} \quad \forall i & \text{biting conditions} \\
w_0(p_i) \leq w_{groove}(p_i) \quad \forall i & \text{groove limits} \\
c_{min} < \frac{h_0(p_i)}{w_0(p_i)} < c_{max} \quad \forall i & \text{ratio constraints} \\
(\frac{w_0(p_i)}{h_0(p_i)}, \frac{l_c(p_i)}{h_0(p_i)}) \in S_{opt} \quad \forall i & \text{shape constraints}
\end{cases}
$$

The first five side conditions make the pass schedule feasible, i.e. it starts with the given entry geometry and ends with the desired exit geometry. As changes in geometry happen only during a pass, the exit dimensions of the previous pass must match those of the next pass. When a tilting operation is executed, thickness and width simply are swapped. The fifth condition allows tilting only in front of the stand, because only one tilting device is installed. The roll spring is a term that increases the exit thickness due to the deformation of rolls and stand. The remaining constraints have already been mentioned above.

Trying to solve the optimisation problem leads to a series of major problems:

1. The solution may not exist or may not be definite.

2. The search space is capacious, and using standard methods to find a solution would likely lead to combinatorical or numerical issues. It is also not clear how to formulate the problem in a normal form to apply such standard methods.

3. The estimators for torque, force, thickness reduction and spread cannot be stated as simple formulas. Either their input parameters are not exactly measurable, not to mention the core temperature of the slab, or they can only be derived via complicated characteristics. The rolling force for example depends on more than 40 material factors — not mentioned the influence of the particular rolling mill. Even rolling mills identical in construction will show slightly different behaviour. Thus, robust estimators are needed, taking all these issues into account to predict a confident value.

4. It would not be possible to generate pass schedules online within a few seconds, because the standard methods are too time-consuming. However, online pass schedule calculation and even recalculations after the start of a schedule should be possible. These recalculations are necessary whenever an intermediate geometry is not reached, e.g. due to variations in temperature or a material permutation.

   The computing time is probably the most challenging constraint when using standard methods.

2.5 Architecture of a Pass Schedule Calculator

Dividing the optimisation problem into a number of subproblems leads to a novel architecture for pass schedule calculation that handles all these issues.

First, the most frequently used pass schedules at the blooming train of Buderus were identified and analysed with regard to their common features to derive a number of *rolling strategies*. Each strategy defines a groove series, geometries to achieve when entering and leaving a groove, and tilting operations. Thus, only the total thickness reduction across a series of passes is defined, but not yet the number of passes to be used for this reduction. Such a series of passes will be called a *sequence*. Rolling strategy and schedule as well as the distinction of passes and sequences are shown in fig. 2.7.

As the overall rolling process is documented by values taken from numerous sensors — such as rolling force and rolling torque sensors — sufficient

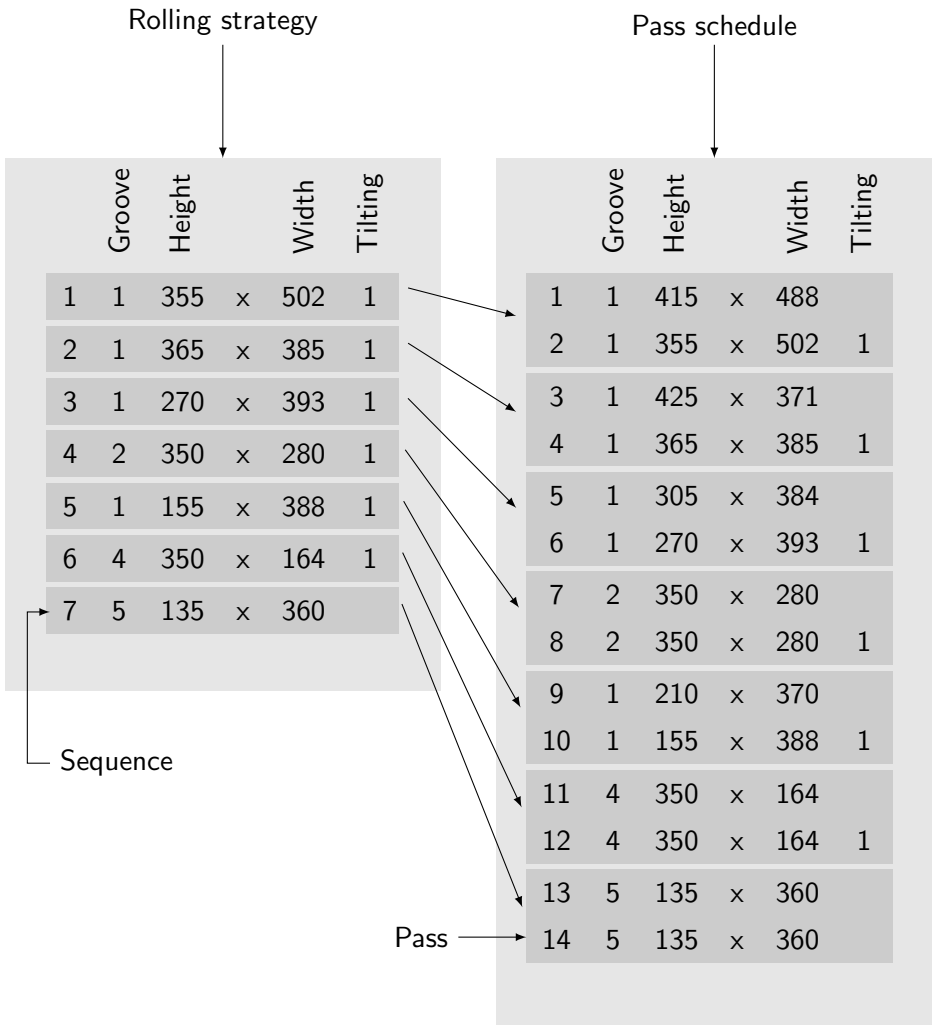


Figure 2.7: Rolling strategy vs. pass schedule. In this example the rolling strategy consists of seven sequences, which are split into 14 passes in the pass schedule.

data should be available to construct predictors for these values. The architecture, construction, and robustness of these predictors will be covered by the following chapters. Finally, with these predictors it will be possible to split each sequence into a couple of passes. Therefore, the thickness reduction is estimated stepwise, by permitting the maximum rolling force and rolling torque for each pass. So, the total thickness reduction is distributed across a number of passes. The shape constraints may further reduce the

feasible thickness reduction. Once the reduction is fixed, the effective rolling force, torque, and spread values are verified by the corresponding prediction modules.

This novel architecture of a pass schedule calculator uses two major components — rolling strategy and predictors in form of neural networks — to generate a pass schedule. Although the resulting schedule may not be the optimal one in terms of the original optimisation problem, it is revolutionary in many ways. An overview of the intended architecture is shown in fig. 2.8.

The current pass schedule databases are static. Extensive manual calculations and years of experience are necessary to add schedules for new slab geometries. Furtermore, substantial modifications to the rolling mill, e.g. the installation of a second tilting device, would require the complete recalculation of all hand-crafted pass schedules. So, necessary improvements to the rolling mill are avoided due to the great effort and complexity of the consequential recalculations.

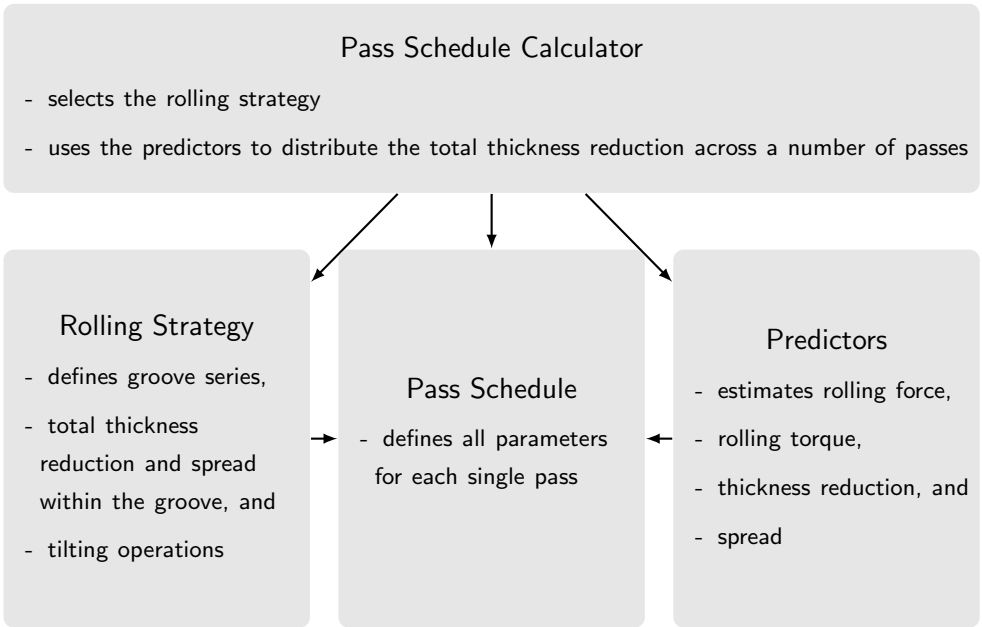


Figure 2.8: Architecture of pass schedule calculator.

In contrast, the new concepts will automate the schedule calculation and will allow the simple introduction of new slab geometries as well as substantial modifications to the rolling mill. Not only the precalculation of pass schedules before the rolling process but also changes between two passes are possible. Thus, the schedule calculator is able to react on permutations of material or variations in temperature. In addition, the pass schedule calculator will not only be applicable to a particular blooming train, but — with slight customisations — to a wide range of hot rolling mills.

Chapter 3

Learning from Examples

Learning from examples is, among reasoning and planning, one of the basic capabilities that constitute intelligence. Given some observations, it describes the ability to formulate universal rules that allow reliable prediction of observations in the future. *Machine learning*, a major branch of artificial intelligence, is engaged in constructing machines with this ability to learn.

Statistical learning theory explains under which circumstances the process of learning will be successful, how fast learning can be done and suggests good learning strategies.

The *support vector machine* (SVM) fulfils these requirements and can be applied to a variety of learning tasks such as classification and regression. *Optimisation theory* provides the necessary tools to transform and solve the mathematical representation of the SVM.

Finally, a framework taken from *regularisation theory* shows the commonalities of neural networks, least-squares approximation, and the support vector machine.

3.1 Statistical Learning Theory

A universal framework to describe the process of learning from examples was first presented by Vapnik [23]. His function estimation model consists of three components - a *generator*, a *supervisor*, and the *learning machine*. The generator produces independently random vectors $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ drawn

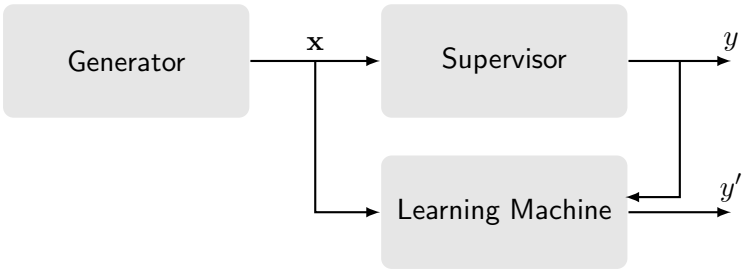


Figure 3.1: Vapnik's model of learning from examples. Generator and supervisor supply the learning machine with the training data pairs $\mathbf{x}$ and y. The learning machine adapts its output behaviour so that the difference between the supervisor's answer y and the learning machine's answer y' becomes less and less.

from an unknown but fixed probability distribution function $P(\mathbf{x})$. The supervisor returns for each $\mathbf{x}$ an output value $y \in \mathcal{Y}$ according to the unknown conditional distribution function $P(y|\mathbf{x})$. The learning machine implements a set of functions $f(\mathbf{x}, \alpha) \in \mathcal{F}$ with parameters $\alpha \in \Lambda$. The input vectors combined with the corresponding output values represent the *training data set*

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^{l},$$

derived by taking l samples according to the *joint probability density* function $P(\mathbf{x}, y)$. The joint density $P(\mathbf{x}, y)$ can be expressed in terms of the *marginal density* $P(\mathbf{x})$ and the *conditional density* $P(y|\mathbf{x})$ by

$$P(\mathbf{x}, y) = P(y|\mathbf{x}) \cdot P(\mathbf{x}).$$

Given the set $\mathcal{D}$, the learning machine has to find an *estimator*

$$f : \mathcal{X} \times \Lambda \rightarrow \mathcal{Y}, f \in \mathcal{F}$$

that predicts for each feature vector $\mathbf{x}$ the output value y. Selecting a proper function family $\mathcal{F}$ as the basis for the learning machine is a crucial step and will directly control the generalisation capabilities of the machine, i.e. it will determine whether the machine is able to learn well or not.

3.1.1 Risk Minimisation

In order to measure the quality of estimation, a *loss function* $L(y, f(\mathbf{x}, \alpha))$ needs to be defined. This function describes the discrepancy between the supervisor's answer y and the estimated answer $y' = f(\mathbf{x}, \alpha)$, derived by the learning machine. Looking for the best estimator according to the predefined loss function is the same as to minimise the expected error, i.e. the *risk functional*

$$R(\alpha) = \int L(y, f(\mathbf{x}, \alpha))\, dP(\mathbf{x}, y). \tag{3.1}$$

Thus, the *target function* or *ideal estimator* f^* fulfils

$$f^*(\mathbf{x}) = f(\mathbf{x}, \alpha^*) \quad \text{with} \quad \alpha^* = \arg\min_{\alpha \in \Lambda} R(\alpha).$$

Unfortunately, f^* cannot be determined, because the probability distribution $P(\mathbf{x}, y)$ is unknown. In order to approximate α^*, a task-specific loss function needs to be defined. Vapnik discriminates three major machine learning tasks by their loss function [23]:

Pattern Recognition Suppose two classes of samples are given. The task is to assign each new sample correctly to one of the two classes. Now, the supervisor's answer can only take two values — 0 and 1 — and the estimated functions $f(\mathbf{x}, \alpha)$ have to be indicator functions, i.e. they are also limited to $\{0, 1\}$. Then, the loss function

$$L(y, f(\mathbf{x}, \alpha)) = \begin{cases} 0 & \text{if } y = f(\mathbf{x}, \alpha) \\ 1 & \text{if } y \neq f(\mathbf{x}, \alpha) \end{cases}$$

indicates whether a pattern was correctly classified by the estimated function or not.

Regression Estimation If an arbitrary function has to be learned, the supervisor's answer can take all real values. The common least-squares approach uses the loss function

$$L(y, f(\mathbf{x}, \alpha)) = (y - f(\mathbf{x}, \alpha))^2$$

and results in the regression function

$$f(\mathbf{x}, \alpha) = \int y\, dF(y|x).$$

Regression estimation will be the main concept to construct estimators for describing the rolling process, as the output values are taken from a continuous scale.

Density Estimation The third type of learning task is density estimation, where a set of densities $f(\mathbf{x}, \alpha)$ is given. The desired density function is one that minimises the risk functional 3.1 with the loss function

$$L(f(\mathbf{x}, \alpha)) = -\log f(\mathbf{x}, \alpha).$$

3.1.2 Empirical Risk Minimisation

As the distribution function $P(\mathbf{x})$ is not known in an explicit sense but only by a limited number of samples, the risk functional $R(\alpha)$ is replaced by the *empirical risk functional*

$$R_{emp}(\alpha) = \frac{1}{P} \sum_{i=1}^{l} L(y_i, f(\mathbf{x}_i, \alpha)). \tag{3.2}$$

According to the law of large numbers, the *arithmetic mean* — in this case R_{emp} — will converge against the *expectation* R with increasing sample size l. However, the arguments that minimise R and R_{emp} are not necessarily the same. In order to find α^* only by minimising R_{emp}, the principle of empirical risk minimisation must be *consistent*, i.e. R and R_{emp} must uniformly converge:

$$\lim_{l \to \infty} P(\sup_{\alpha \in \Lambda} |R(\alpha) - R_{emp}(\alpha)| < \epsilon) = 0 \tag{3.3}$$

Necessary and sufficient conditions for consistency were defined by Vapnik and Chervonenkis [25].

3.1.3 Vapnik-Chervonenkis Dimension

Vapnik and Chervonenkis defined a measure to describe the expressive power of a family of classification functions. The set $\mathcal{D}$ of l training patterns can be labelled with 0 and 1 in 2^l different ways. Let $N(\mathcal{D}, \mathcal{F})$ be the number of *dichotomies* for the set $\mathcal{D}$ that can be realized by a family $\mathcal{F}$ of classification functions. Then, the *growth function*

$$G(l) = \ln \left(\max_{\mathcal{D}} N(\mathcal{D}, \mathcal{F}) \right) \leq \ln 2^l = l \cdot \ln 2$$

is a measure of the maximum number of different labellings for an arbitrary set of length l. The *VC dimension* of a function family is the maximum number h of patterns, so that these patterns can be separated correctly for each arbitrary labelling. Either the VC dimension is infinite or bounded according to the type of growth function:

1. $$G(l) = l \cdot \ln 2 \quad \forall\, l$$

   Thus, for all arbitrary l there exists a dataset $\mathcal{D}$ so that the function family can discriminate all different labellings of $\mathcal{D}$. The growth function is linear and the *VC dimension* infinite.

2. $$G(l) \begin{cases} = l \cdot \ln 2 & \text{if } l \leq h \\ \leq h \cdot \left(\ln \frac{l}{h} + 1\right) & \text{if } l > h \end{cases}$$

   When the number of patterns exceeds the threshold h — the VC dimension — the growth function is bounded by a logarithmic function. Now, the function family does not contain a discriminating function for each possible labelling anymore.

A finite VC dimension is necessary and sufficient for uniform convergence, as defined in (3.3), and will guarantee fast convergence [24, 25]. Thus, learning by minimising the empirical risk will be successful, as the empirical risk converges to the expected risk.

In order to generalise the VC dimension for regression, the loss function $L(y, f(\mathbf{x}, \alpha))$ must be bounded:

$$l_{min} \leq L(y, f(\mathbf{x}, \alpha)) \leq l_{max}.$$

Then, the VC dimension of the binary indicator function

$$\Theta(L(y, f(\mathbf{x}, \alpha)) - \beta) \quad \text{with} \quad \Theta(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases}$$

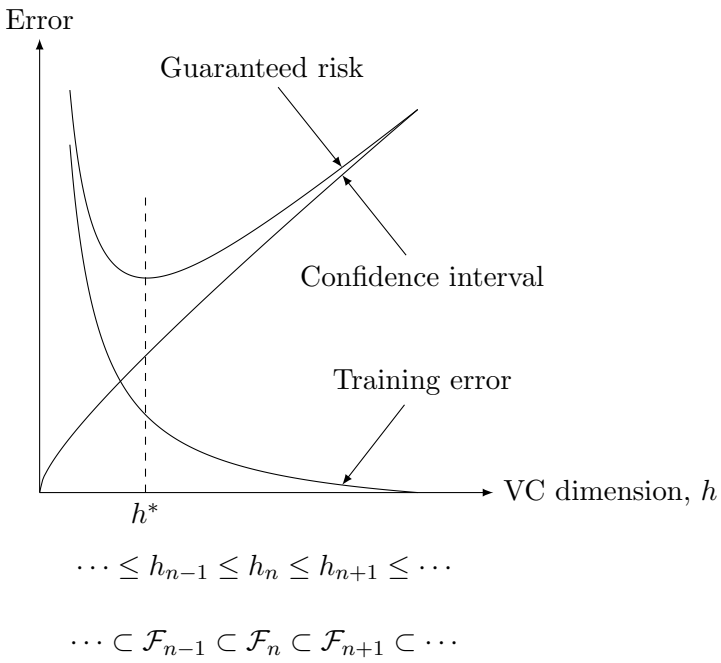


Figure 3.2: Tradeoff between training error and confidence interval. The guaranteed risk is an upper bound for the sum of both learning performance measures. Choosing a function class with VC dimension h^* will yield the smallest guaranteed risk ([5], slightly modified)

with parameter $\beta \in [l_{min}, l_{max}]$ is used to characterise the continuous function class $\mathcal{F}$ with elements $f(\mathbf{x}, \alpha)$. Now, the finiteness of the VC dimension is only a sufficient condition but no longer necessary for uniform convergence. Alon et al. introduced the V_γ dimension to describe necessary conditions as well [1].

3.1.4 Structural Risk Minimisation

When constructing a learning machine, the challenge is to define a proper function family that is both limited to achieve a low VC dimension and large enough to contain the function that best describes the data. Vapnik proved the generalisation error to be upper bounded by the *guaranteed risk*, which is the sum of the training error and the *confidence interval*. The confidence interval is a measure for the probability that a function, taken from the

given function family, with small generalisation error can be found at all. The confidence interval will increase with increasing VC dimension while the training error will decrease. This trade-off between *training error* and *confidence interval* is illustrated in fig. 3.2.

Now, the question is how to determine the function family that yields the least guaranteed risk. The idea of *structural risk minimisation* [23] is to define a series of nested *hypothesis spaces*

$$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \cdots \subset \mathcal{F}_n$$

with increasing VC dimension

$$h_1 \leq h_2 \leq \cdots \leq h_n.$$

The machine with the smallest guaranteed risk h^* is chosen during the learning process. In practice this might be implemented in various ways, e.g. by increasing h until the training error does not decrease significantly anymore. A major feature of the *support vector machine* — which will be used for training in the scope of the pass schedule calculator – is the simultaneous minimisation of training error and VC dimension.

3.2 Optimisation Theory — Constrained Minimisation

When dealing with the mathematical representation of the support vector machine, some tools from optimisation theory are needed.

All types of support vector machines can be expressed in a representation similar to the following constrained minimisation problem — the so-called *primal problem*:

$$\begin{aligned} &\text{Minimise} \quad f(\mathbf{w}), \quad \mathbf{w} \in \Omega \subseteq \mathbb{R}^n, \\ &\text{subject to} \quad \begin{cases} g_i(\mathbf{w}) \leq 0 \quad i = 1, \ldots, k, \\ h_i(\mathbf{w}) = 0 \quad i = 1, \ldots, m \end{cases} \end{aligned} \tag{3.4}$$

The so-called *Lagrangian function* combines both the objective function f and the constraints within one equation.

$$L(\mathbf{w}, \alpha, \beta) = f(\mathbf{w}) + \sum_{i=1}^{k} \alpha_i g_i(\mathbf{w}) + \sum_{i=1}^{m} \beta_i h_i(\mathbf{w}). \tag{3.5}$$

Each constraint is weighted with a positive *Lagrangian multiplier* α_i or β_i. Kuhn and Tucker stated conditions for the optimum of (3.4) to exist depending on the partial derivatives of the Lagrangian function [8]:

Theorem 3.1 *Given the optimisation problem 3.4 where f is convex and h_i, g_i are affine mappings. Necessary and sufficient conditions for an optimum w^* is the existence of α^* and β^* with*

$$\begin{aligned}
\frac{\partial L(\mathbf{w}^*, \alpha^*, \beta^*)}{\partial \mathbf{w}} &= \mathbf{0}, \\
\frac{\partial L(\mathbf{w}^*, \alpha^*, \beta^*)}{\partial \beta} &= 0, \\
\alpha_i^* g_i(\mathbf{w}^*) &= 0, \quad i = 1, \ldots, k, \\
g_i(\mathbf{w}^*) &\leq 0, \quad i = 1, \ldots, k, \\
\alpha_i^* &\geq 0, \quad i = 1, \ldots, k.
\end{aligned}$$

If $f(\mathbf{w})$ is convex and g_i and h_i are both affine functions, the solution of the primal problems equals the solution of the dual problem, which is to

$$\begin{aligned}
&\text{Maximise} \quad \min_{\mathbf{w} \in \Omega} L(\mathbf{w}, \alpha, \beta) \\
&\text{subject to} \quad \alpha \geq 0.
\end{aligned} \tag{3.6}$$

So the task is no longer to find the minimum of the objective function, but to find a saddle point of the Lagrangian function, which implicates a number of significant advantages when dealing with the support vector machine.

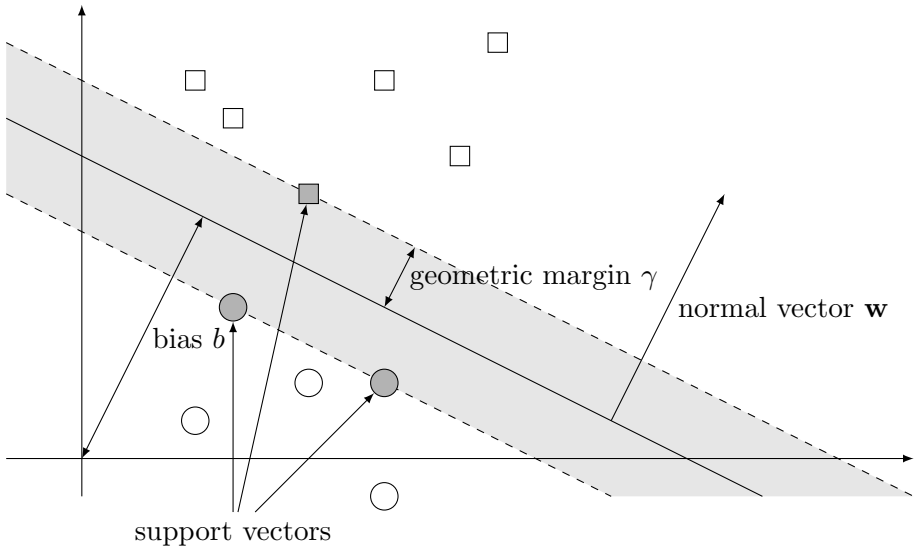


Figure 3.3: Maximal margin classifier. The classifier is well-defined by a normal vector $\mathbf{w}$ and the bias b. Points with geometric margin γ are called support vectors.

3.3 Support Vector Machines

3.3.1 Maximal Margin Classifier

The simplest support vector machine is the one that implements a maximal margin classifier.

Given the training data set

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \{-1, +1\}\}_{i=1}^{l},$$

the classification task is to find a hyperplane that separates the patterns with $y = -1$ from those with $y = +1$. In terms of statistical learning theory, we want to determine the linear hypothesis that leads to the smallest generalisation error. This hyperplane is represented by a normal vector w and a bias b that describes the distance of the hyperplane from the origin. The minimal distance from the hyperplane to a pattern is called *geometric margin* (see fig. 3.3). The maximal margin classifier selects that hyperplane among the set of all separating hyperplanes with the largest margin.

Suppose the maximal margin classifier is given by $(\mathbf{w}, b)$. Then, for all points $f(\mathbf{x}_i) = y_i \cdot \left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 0$ is satisfied. Furthermore, the length of w is chosen so that the points with the least distance to the hyperplane from both sides — x^+ and x^- — have a *functional margin* of $+1$ and -1 respectively:

$$\begin{aligned}\mathbf{w}^T\mathbf{x}^+ + b &= +1\\ \mathbf{w}^T\mathbf{x}^- + b &= -1\end{aligned}$$

The geometric margin is derived by normalising $\mathbf{w}$:

$$\begin{aligned}\gamma &= \frac{1}{2}\left(\frac{1}{||\mathbf{w}||_2}\cdot\mathbf{w}^T\mathbf{x}^+ - \frac{1}{||\mathbf{w}||_2}\cdot\mathbf{w}^T\mathbf{x}^-\right)\\ &= \frac{1}{2||\mathbf{w}||_2}\left(\mathbf{w}^T\mathbf{x}^+ - \mathbf{w}^T\mathbf{x}^-\right)\\ &= \frac{1}{||\mathbf{w}||_2}.\end{aligned}$$

So, in order to maximise the margin, the 2-norm of the normal vector has to be minimised, provided that all patterns are classified correctly:

$$\begin{aligned}&\text{Minimise} && \mathbf{w}^T\mathbf{w}\\ &\text{subject to} && y_i\left(\mathbf{w}^T\cdot\mathbf{x}+b\right)\geq 1, \quad i=1,\ldots,l\end{aligned}$$

Thus the Lagrangian, according to (3.4) and (3.5) is

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{l}\alpha_i\left(y_i\left(\mathbf{w}^T\cdot\mathbf{x}_i+b\right)-1\right).$$

The factor $\frac{1}{2}$ is used for convenience, as it gets eliminated when differentiating to find the minimum of $L(\mathbf{w}, \alpha, \beta)$:

$$\begin{aligned}\frac{\partial L(\mathbf{w},b,\alpha)}{\partial\mathbf{w}} &= \mathbf{w} - \sum_{i=1}^{l}\alpha_i y_i\mathbf{x}_i = \mathbf{0} \Leftrightarrow \mathbf{w} = \sum_{i=1}^{l}\alpha_i y_i\mathbf{x}_i\\ \frac{\partial L(\mathbf{w},b,\alpha)}{\partial b} &= \sum_{i=1}^{l}\alpha_i y_i = 0\end{aligned}$$

Resubstitution delivers

$$\begin{aligned} L(\mathbf{w}, b, \alpha) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{l} \alpha_i \left(y_i \left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1\right) \\ &= \frac{1}{2}\sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j - \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j + \sum_{i=1}^{l} \alpha_i \\ &= \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j. \end{aligned}$$

In total, the dual problem is to

$$\begin{array}{ll} \text{maximise} & \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to} & \begin{cases} \sum_{i=1}^{l} \alpha_i y_i & = 0 \\ \alpha_i & \geq 0 \quad i = 1, \ldots, l \end{cases} \end{array} \tag{3.7}$$

Suppose the optimum is found for $\alpha = \alpha^*$. Then the weight vector $\mathbf{w}^*$ and the bias b^* of the maximal margin hyperplane are

$$\begin{aligned} \mathbf{w}^* &= \sum_{i=1}^{l} y_i \alpha_i^* \mathbf{x}_i \\ b^* &= -\frac{\max_{i,y_i=-1}\left(\mathbf{x}_i^T\mathbf{w}^*\right) + \min_{i,y_i=+1}\left(\mathbf{x}_i^T\mathbf{w}^*\right)}{2}. \end{aligned}$$

The advantages of this dual representation are:

1. The optimisation problem is convex, so that a unique solution exists, which can be obtained by quadratic programming.

2. The third Kuhn-Tucker condition in theorem 3.1 is satisfied if

$$\alpha_i^* \cdot \left(y_i \left(\mathbf{x}_i^T\mathbf{w}^* + b^*\right) - 1\right) = 0, \quad i = 1, \ldots, l.$$

This implies that only those α_i differ from 0 that have functional margin $+1$ or -1. Hence, the corresponding patterns $\mathbf{x}_i$ are called *support vectors*, and all other patterns can be omitted as their Lagrangian

multipliers are 0

$$\begin{aligned} f(\mathbf{x}) &= \sum_{i=1}^{l} y_i \alpha_i^* \mathbf{x}_i^T \mathbf{x} + b^* \\ &= \sum_{i \in S=\{s|\alpha_s>0\}} y_i \alpha_i^* \mathbf{x}_i^T \mathbf{x} + b^* \end{aligned} \tag{3.8}$$

3. The concept of kernels, which allows more complicated decision borders to overcome the limitation to linear separable classes, requires the dual problem.

This *hard margin classifier* is strongly affected by outliers — one single outlier may avoid linear separation. Thus, a *soft margin* with so-called *slack variables* is introduced. Different soft margin approaches will be discussed in detail for support vector regression (see section 3.3.3).

3.3.2 Introducing Nonlinearity — the Idea of Kernels

Linear hypothesis spaces will occur only in very few real-world applications, so a concept to deal with nonlinearity must be introduced. The idea of kernels is to transform the low dimensional input space $\mathcal{X}$ into a high dimensional feature space $\mathcal{X}'$. With increasing dimension the probability of linear separability rises so that a linear learning machine would be able to determine a correct decision border.

A linear hypothesis is a function of the type

$$f(\mathbf{x}) = \sum_{i=1}^{l'} w_i \Phi_i(\mathbf{x}) + b$$

where $\mathbf{\Phi}$ is a (non-linear) mapping from the input space to the feature space with

$$\begin{aligned} \mathbf{\Phi} : \mathcal{X} &\mapsto \mathcal{X}' \\ \mathbf{x} = (x_1, \ldots, x_l) &\mapsto \mathbf{\Phi}(\mathbf{x}) = (\Phi_1(\mathbf{x}), \ldots, \Phi_{l'}(\mathbf{x})), \quad l < l'. \end{aligned}$$

According to 3.8 the separating hyperplane or linear hypothesis can be expressed in dual representation by

$$f(\mathbf{x}) = \sum_{i=1}^{l'} \alpha_i y_i \mathbf{\Phi}(\mathbf{x}_i)^T \mathbf{\Phi}(\mathbf{x}) + b.$$

The mapping $\mathbf{\Phi}$ only occurs in the inner product

$$K(\mathbf{x}_i, \mathbf{x}) = \mathbf{\Phi}(\mathbf{x}_i)^T \mathbf{\Phi}(\mathbf{x}).$$

If this inner product can be expressed directly in terms of the input vectors, then the mapping into the high dimensional feature space need not be done explicitly for each vector. Such a function K is called a *kernel* and has some interesting properties:

1. The kernel can be expressed by a *kernel matrix*

$$\mathbf{K} = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^{l}.$$

2. If the kernel is known, the decision rule can be evaluated efficiently by

$$f(\mathbf{x}) = \sum_{i=1}^{l'} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b.$$

   Neither the mapping $\mathbf{\Phi}$ nor the structure of the underlying feature space must be known.

Some of the most widely used kernel functions are:

Polynomial kernels

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + c)^p, \quad p \in \mathbb{N},\ c \geq 0$$

Radial basis function kernels (RBF or Gaussian kernels)

$$K(\mathbf{x}, \mathbf{y}) = e^{-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2}}, \quad \sigma > 0$$

Linear combinations of kernels

$$K(\mathbf{x}, \mathbf{y}) = c_1 \cdot K_1(\mathbf{x}, \mathbf{y}) + c_2 \cdot K_2(\mathbf{x}, \mathbf{y}), \quad c_1, c_2 \geq 0$$

3.3.3 Support Vector Regression

The support vector machine can easily be extended to the regression estimation task. Suppose the training data is given by

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathbb{R}\}_{i=1}^{l}.$$

In ϵ-support vector regression (ϵ-SVR), according to Vapnik [23], the aim is to find a linear function

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b, \quad \mathbf{w} \in \mathcal{X}, b \in \mathbb{R},$$

which deviates no more than ϵ from y_i for each $\mathbf{x}_i$. Among the set of linear hypotheses that fit all points into an ϵ-tube volume, the flattest one is preferred. This can be achieved by minimising the 2-norm of the weight vector $\mathbf{w}$. Thus, on the assumption that such a function exists, the problem can be stated as a convex minimisation problem:

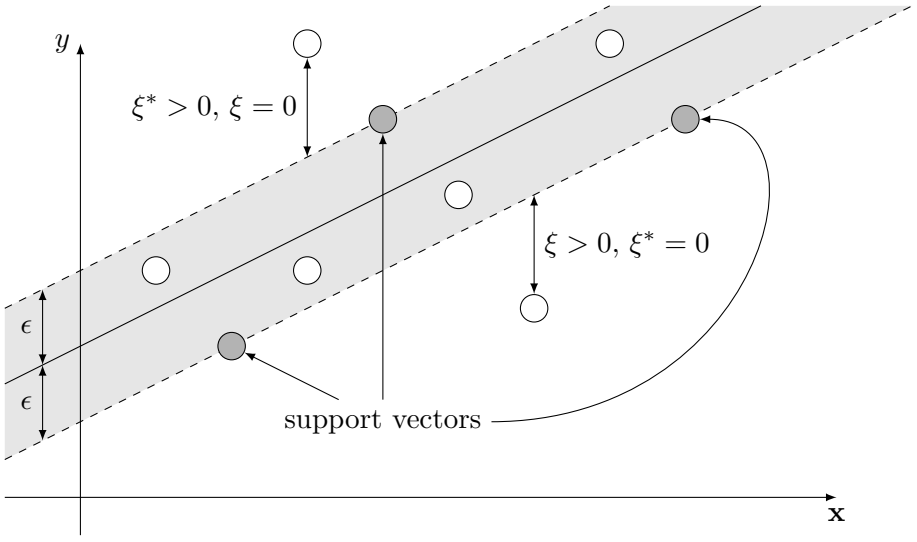


Figure 3.4: Support vector regression. Within ϵ-support vector regression the flattest regression function is found that allows at most an error of ϵ. As normally outliers exist, the margin is softened by introducing slack variables ξ and ξ^* for each data point.

$$\begin{aligned} &\text{Minimise} \quad \tfrac{1}{2}||\mathbf{w}||^2 \\ &\text{subject to} \quad \begin{cases} y_i - \mathbf{w}^T\mathbf{x}_i - b \leq \epsilon \\ \mathbf{w}^T\mathbf{x}_i + b - y_i \leq \epsilon \end{cases} \end{aligned} \tag{3.9}$$

Supposing all points fit into an ϵ-tube is often too restrictive, so a mechanism to *soften* the tube is required. Cortes and Vapnik [2] introduced *slack variables* to allow *soft margin* classification, so that some training patterns may be within the margin or even misclassified. Vapnik extended this concept for regression by declaring two slack variables ξ_i and ξ_i^* for each input point x_i. If a data point is situated within the tube, both slack variables are 0. The slack variable ξ_i measures positive deviation of more than ϵ, while ξ_i^* indicates negative deviations. Since no point can be on both sides simultaneously, at least one of the two slack variables is always 0. Mathematically, this corresponds to the so-called ϵ-insensitive loss function:

$$|\xi|_\epsilon = \begin{cases} 0 & \text{if } |\xi| \leq \epsilon \\ |\xi| - \epsilon & \text{otherwise} \end{cases}$$

The amount of outliers and their influence on the regression function is adjusted by incorporating the weighted norm of the slack variables into the objective function. The two most common approaches add the weighted value of the 1-norm or the 2-norm.

The *1-Norm Soft Margin* approach adds the regularisation term

$$C \cdot \left(\sum_i \xi_i + \sum_i \xi_i^* \right)$$

to the objective function, resulting in the following optimisation problem:

$$\begin{aligned} &\text{Minimise} \quad \tfrac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{l} (\xi_i + \xi_i^*) \\ &\text{subject to} \quad \begin{cases} y_i - \mathbf{w}^T\mathbf{x}_i - b & \leq \epsilon + \xi_i \\ \mathbf{w}^T\mathbf{x}_i + b - y_i & \leq \epsilon + \xi_i^* \\ \xi_i,\, \xi_i^* & \geq 0 \end{cases} . \end{aligned} \tag{3.10}$$

The regularisation parameter C determines the trade-off between flatness and toleration of errors beyond ϵ. Parameter selection techniques such as *grid search* to pick an appropriate C will be presented in chapter 4.4.4. Applying the Lagrangian theory and substituting the inner product by the kernel function K results in the dual representation:

$$\begin{array}{ll} \text{Maximise} & \begin{cases} -\frac{1}{2}\sum\limits_{i,j=1}^{l}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)K(\mathbf{x}_i,\mathbf{x}_j) \\ -\epsilon\sum\limits_{i=1}^{l}(\alpha_i+\alpha_i^*)+\sum\limits_{i=1}^{l}y_i(\alpha_i-\alpha_i^*) \end{cases} \\ \text{subject to} & \begin{cases} \sum\limits_{i=1}^{l}(\alpha_i-\alpha_i^*) & =0 \\ \alpha_i,\alpha_i^* & \in[0,C] \end{cases} \end{array} \tag{3.11}$$

The second constraint ensures that all Lagrangian multipliers are upper bounded by C. Thus, the influence of outliers on the regression function

$$f(\mathbf{x})=\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)K(\mathbf{x}_i,\mathbf{x})+b \tag{3.12}$$

is limited.

A dual representation with fewer constraints is derived by using the 2-norm instead of the 1-norm for regularisation. The primal problem is then as follows:

$$\begin{array}{ll} \text{Minimise} & \frac{1}{2}\|\mathbf{w}\|^2+C\sum\limits_{i=1}^{l}(\xi_i^2+\xi_i^{*2}) \\ \text{subject to} & \begin{cases} y_i-\mathbf{w}^T\mathbf{x}_i-b & \leq\epsilon+\xi_i \\ \mathbf{w}^T\mathbf{x}_i+b-y_i & \leq\epsilon+\xi_i^* \\ \xi_i,\,\xi_i^* & \geq 0 \end{cases} . \end{array} \tag{3.13}$$

Suppose the third constraint is not satisfied, i.e. $\xi_i^{(*)}<0$. Setting $\xi_i^{(*)}=0$ does not violate the first two constraints but decreases the value of the objective function. Therefore, the third constraint can be omitted. Converting

this reduced optimisation problem into the dual representation results in:

$$\begin{array}{ll} \text{Maximise} & \begin{cases} -\frac{1}{2}\sum\limits_{i,j=1}^{l}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\left(K(\mathbf{x}_i,\mathbf{x}_j)+\frac{\delta_{ij}}{C}\right) \\ -\epsilon\sum\limits_{i=1}^{l}(\alpha_i+\alpha_i^*)+\sum\limits_{i=1}^{l}y_i(\alpha_i-\alpha_i^*) \end{cases} \\ \text{subject to} & \begin{cases} \sum\limits_{i=1}^{l}(\alpha_i-\alpha_i^*)=0 \\ \alpha_i,\alpha_i^*\geq 0 \end{cases} \end{array} \tag{3.14}$$

The usage of Kronecker's delta with

$$\delta_{ij}=\begin{cases} 1 & \text{if i = j} \\ 0 & \text{otherwise} \end{cases}$$

can be interpreted as a modification of the kernel matrix. The support vector machine based on this soft margin representation can be trained by the same algorithms as in the hard margin case, but with a modified kernel

$$\hat{K}(\mathbf{x}_i,\mathbf{x}_j)=K(\mathbf{x}_i,\mathbf{x}_j)+\frac{\delta_{ij}}{C}.$$

3.3.4 Incorporating Prior Knowledge

In many practical applications of machine learning some amount of prior knowledge is available. Depending on its formulation there are different techniques to incorporate this knowledge into the SVM to improve the generalisation capabilities.

Le, Smola, and Gärtner [10] introduced prior knowledge in binary classification by ensuring that for a certain subset of the training patterns the classification must be correct. For scalar regression they defined prior knowledge in form of upper and lower bounds, and for multiclass classification they restricted the possible classes for a subset of the patterns.

Wu and Srihari [26] proposed the *weighted margin support vector machine*, where a confidence value is assigned to each training pattern. Patterns with high confidence values affect the decision plane by a larger amount than

low-confident patterns. Thus, the training set can be extended by pseudo-training patterns to incorporate human prior knowledge.

3.4 Regularisation Theory

In general, approximating a function from data is an ill-posed problem, in contrast to well-posed problems that have a unique solution for any input and where the solution depends continuously on the input. The classical way to solve such ill-posed problems is known as *regularisation*.

3.4.1 A Generalised Framework for Function Estimation

Expanding the empirical risk functional (3.2) by a *stabiliser term* to add certain constraints to the regression function f leads to

$$H(f) = \frac{1}{l} \sum_{i=1}^{l} L(y_i, f(\mathbf{x}_i)) + \lambda \Phi. \tag{3.15}$$

The first term minimises the distance between the regression function and the set of examples, the second term adds certain flatness conditions. The importance of both terms is weighted by the regularisation parameter λ. This general formulation was used by Evgeniou et al. [4] to cover different approximation problems by the same framework. Four major regularisation problems can be distinguished by the choice of the loss function L, the regularisation parameter λ, and the stabiliser Φ:

Least Squares Approximation

$$\begin{aligned} L(y_i, f(\mathbf{x}_i)) &= (y_i - f(\mathbf{x}_i))^2 \\ \lambda &= 0 \end{aligned}$$

The common least squares approximation does not use any type of stabiliser. So, the parameters of the regression function f are not constrained and will be prone to overfitting.

Classical Regularisation

$$\begin{aligned} L(y_i, f(\mathbf{x}_i)) &= (y_i - f(\mathbf{x}_i))^2 \\ \lambda &> 0 \\ \Phi &= \|f\|_K^2, \end{aligned}$$

The stabiliser term $\|f\|_K^2$ is a norm in a Reproducing Kernel Hilbert Space (RKHS) defined by the positive definite function K. Classical regularisation is also known as *Ridge Regression* or *Tikhonov Regularisation*.

Soft Margin Support Vector Regression

$$\begin{aligned} L(y_i, f(\mathbf{x}_i)) &= |y_i - f(\mathbf{x}_i)|_\epsilon = \xi_i + \xi_i^* \\ \lambda &= \frac{1}{2C'} \\ \Phi &= \|\mathbf{w}\|^2 \end{aligned}$$

Substitution in (3.15) delivers the optimisation problem

$$\begin{aligned} \min_{\mathbf{w},\xi,\xi^*} H(\mathbf{w}, \xi, \xi^*) &= \frac{1}{l}\sum_{i=1}^{l}(\xi_i + \xi_i^*) + \frac{1}{2C'}\|\mathbf{w}\|^2 \\ &= \frac{1}{C'}\left(\underbrace{\frac{C'}{l}}_{C}\sum_{i=1}^{l}(\xi_i + \xi_i^*) + \frac{1}{2}\|\mathbf{w}\|^2\right) \end{aligned}$$

with the same constraints as in (3.10). After elimination of the factor $\frac{1}{C'}$ the solution of both optimisation problems will be equal.

Soft Margin Support Vector Classification

$$\begin{aligned} L(y_i, f(\mathbf{x}_i)) &= |1 - y_i f(\mathbf{x}_i)|_+ = \begin{cases} 1 - y_i f(\mathbf{x}_i) & \text{if } 1 - y_i f(\mathbf{x}_i) > 0 \\ 0 & \text{otherwise} \end{cases} \\ &= \xi_i \\ \lambda &= \frac{1}{2C'} \\ \Phi &= \|\mathbf{w}\|^2 \end{aligned}$$

Even for classification this framework is applicable with the assumption that $y_i \in \{-1, +1\}$. This results in the following quadratic programme:

$$\begin{aligned} \min_{\mathbf{w},\xi} H(\mathbf{w},\xi) &= \frac{1}{l}\sum_{i=1}^{l}\xi_i + \frac{1}{2C'}\|\mathbf{w}\|^2 \\ &= \frac{1}{C'}\left(\underbrace{\frac{C'}{l}}_{C}\sum_{i=1}^{l}\xi_i + \frac{1}{2}\|\mathbf{w}\|^2\right) \\ \text{subject to} \quad & \begin{cases} y_i f(\mathbf{x}_i) & \geq 1-\xi_i \\ \xi_i & \geq 0 \end{cases} \end{aligned}$$

So, regularisation and support vector theory are very closely related. Both concepts use stabilisers or smoothness factors to limit the hypothesis space and to ensure that no chaotic hypotheses may be the result of the learning process. This correlates with most observations from the real world. Normally, within systems above a certain complexity small changes of the input parameters will result in small changes of the output variable — so smoothness can be seen a a basic concept of real-world dependencies.

3.4.2 Neural Network, Regularisation, and the SVM

The architecture of the support vector machine can also be described graphically similar to neural networks with one hidden layer of neurons. The input layer consists of n neurons as the dimension of each training pattern is n. Each input neuron is connected to l hidden neurons — one for each training pattern — to derive the inner product kernels. Each hidden layer implements the kernel function K. The hidden layer and an additional neuron to incorporate the bias b are connected to the output neuron (see fig. 3.5).

In contrast to conventional neural networks, where the complexity is directly controlled by the number of hidden neurons *before* training starts, the support vector machine is self-organizing. Once training is finished, only the support vectors and their Lagrangian multipliers affect the output — the remaining hidden neurons are omitted. Thus, the architecture of the neural

network — especially the number of hidden neurons — has not to be known in advance. On the other hand, the hidden neurons are strictly bound to the training vectors, so that the hidden neurons have no ability to adapt. Within classical multilayer networks the basis functions are normally not fixed.

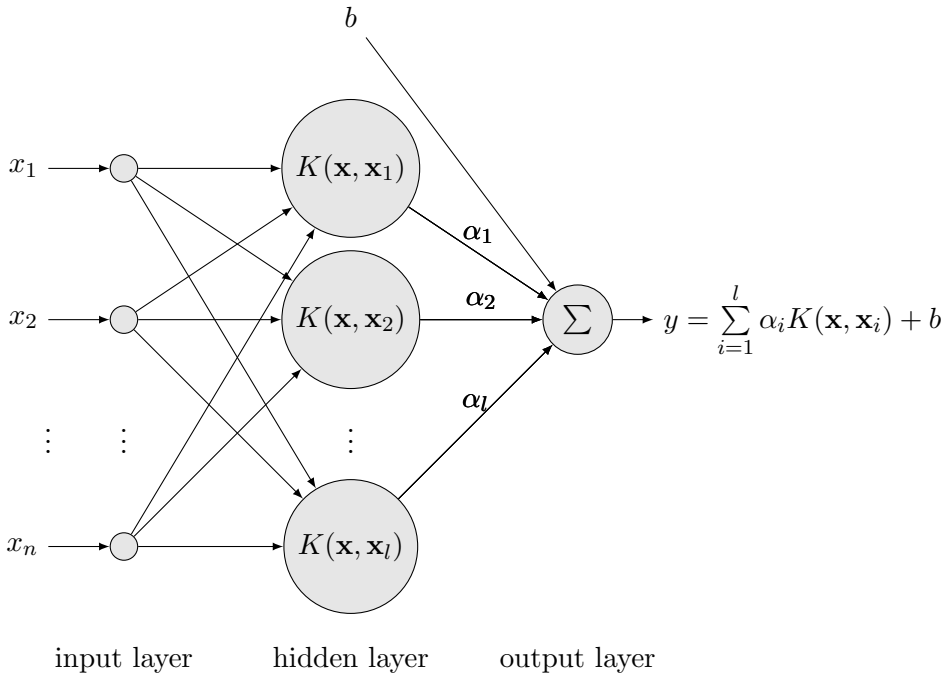


Figure 3.5: Interpretation of the SVM as a Neural Network. The hidden layer determines the kernel function values, and the output neuron calculates the weighted sum of the kernel values and the bias. Note: The input values x_i are the particular entries of the input vector $\mathbf{x}$, whereas the vectors $\mathbf{x}_i$ are the training patterns. After training, only the hidden neurons with $\alpha_i > 0$ have an influence on the output y. The corresponding $\mathbf{x}_i$ are the support vectors.

Chapter 4

Training of Support Vector Machines

Training of learning machines to derive a decision or regression function can be done in many different ways. So, the aim is to find algorithms that are simple to understand, easy to implement, and which have a runtime that scales linearly within the number of training patterns.

Most implementation techniques to solve optimisation problems are based on gradient descent. The straight-forward approach to use existing *optimisation toolboxes* may often not be feasible as these toolboxes are expensive and seldom error-free.

The *Sequential Minimal Optimisation* algorithm, proposed by Platt, is probably the most widely used algorithm for support vector learning. This algorithm breaks the entire problem into smaller ones based on some heuristics to solve them analytically.

In contrast, the *MinOver* algorithm — an extension to Rosenblatt's *Perceptron* — processes the training set pattern-by-pattern and is very easy to understand. Since MinOver works for classification as well as for regression, it can be applied to a wide range of tasks.

Apart from the core algorithm, there are a number of implementation issues that have to be considered, in order to construct an efficient algorithm to be used in practice.

4.1 Quadratic Programming Toolboxes

Dual representations of support vector machines as in (3.7), (3.11), and (3.14) are special cases of the quadratic programme

$$\begin{aligned} &\text{Maximise} && \mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{f}^T\mathbf{x} && \left(\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n,\, \mathbf{H} \in \mathbb{R}^{n\times n},\, \mathbf{f} \in \mathbb{R}^n\right) \\ &\text{subject to} && \begin{cases} \mathbf{A}\mathbf{x} & = \mathbf{b} \\ \mathbf{x} & \geq \mathbf{0}. \end{cases} && \left(\mathbf{A} \in \mathbb{R}^{m\times n},\, \mathbf{b} \in \mathbb{R}^m\right) \end{aligned}$$

Most available algorithms are based on sophisticated gradient ascent and Newton iteration methods to find the optimum. As these techniques become extremely complicated and numerically challenging for large dimensions, one has to mistrust the correctness of many — also commercially available — toolboxes.

Besides, the matrix $\mathbf{H}$, which contains the kernel matrix with l^2 entries, would not even fit into the working memory when dealing with many thousands of training patterns. *Chunking* and *working set selection* are methods to successively choose small subsets of the training patterns for optimisation. However, these techniques make the algorithms even more complicated.

So, one would like not to have algorithms that require such expensive commercial optimisation toolboxes, but to use simple and fast techniques that are adjusted for usage with support vector machines.

Probably the first incremental learning algorithm that implements a linear classifier — but not yet the maximal margin classifier — was the *Perceptron*.

4.2 Iterative Learning Algorithms

4.2.1 Rosenblatt's Perceptron

In the late 1950's Rosenblatt proposed the first pattern-by-pattern learning algorithm for binary classification [17]. The decision function or hypothesis

$f(\mathbf{x})$ to predict the class of an unlabelled pattern $\mathbf{x}$ is defined by

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w}^T\mathbf{x} + b) = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} + b \geq 0 \\ -1 & \text{otherwise,} \end{cases} \tag{4.1}$$

taking two parameters — the *weight vector* $\mathbf{w}$ and the *bias* b.

Starting with an initial weight vector and bias, the algorithm successively updates them by a trial-and-error method until all patterns are classified correctly according to (4.1). If the patterns are not linearly separable, the algorithm does not terminate, since correct classification of all patterns is impossible.

It can be shown that for linear separable training sets, having margin γ and a maximum distance to the origin of $R = \max_i ||\mathbf{x}_i||$, the algorithm will make at most $\left(\frac{2R}{\gamma}\right)^2$ mistakes. Thus, after a finite number of iterations it *will* terminate, returning a separating hyperplane indicated by the weight vector $\mathbf{w}$ and the bias b. If such a hyperplane does not exist, the algorithm

Input:
$(\mathbf{x}_i, y_i)$, $i \in \{1, \ldots, l\}$ — training patterns
Output:
$\mathbf{w}$, b — normal vector and bias of separating hyperplane

$\mathbf{w} \leftarrow \mathbf{0}$
$b \leftarrow 0$
$R \leftarrow \max_i ||\mathbf{x}_i||$
repeat
  forall i **do**
    if $y_i(\mathbf{w}^T\mathbf{x} + b) \leq 0$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + y_i\,\mathbf{x}_i$
      $b \leftarrow b + y_i\,R^2$
    end
  end
until *until no mistakes made within the forall loop*

Figure 4.1: Perceptron learning algorithm in primal representation for linear classification.

Input:
$(\mathbf{x}_i, y_i)$, $i \in \{1, \ldots, l\}$ — training patterns
Output:
α, b — dual coordinates and bias of separating hyperplane

$\alpha \leftarrow \mathbf{0}$
$b \leftarrow 0$
$R \leftarrow \max_i ||\mathbf{x}_i||$
repeat
  forall i **do**
    if $y_i \left(\sum_{j=1}^{l} \alpha_j \, y_j \, \mathbf{x}_j^T \mathbf{x}_i + b \right) \leq 0$ **then**
      $\alpha_i \leftarrow \alpha_i + 1$
      $b \leftarrow b + y_i \, R^2$
    end
  end
until *until no mistakes made within the forall loop*

Figure 4.2: Perceptron learning algorithm in dual representation for linear classification.

will oscillate forever between different hypotheses.

The Perceptron algorithm can be formulated either in primal or dual representation (see figs. 4.1 and 4.2), due to the notation of the decision plane as a linear combination

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i \, y_i \, \mathbf{x}_i \tag{4.2}$$

of training patterns. Now the decision function is

$$\begin{aligned} f(\mathbf{x}) &= \operatorname{sgn}\left(\mathbf{w}^T \mathbf{x} + b \right) \\ &= \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i \, y_i \, \mathbf{x}_i^T \mathbf{x} + b \right). \end{aligned}$$

Since the algorithm terminates as soon as a separating hyperplane is found, the maximal margin solution is not found in general.

4.2.2 Sequential Minimal Optimisation

The idea behind Platt's *Sequential Minimal Optimisation* algorithm (SMO) is to optimise only two Lagrangian variables simultaneously [15, 16]. Consider the dual representation of the soft margin SVM for classification:

$$\begin{aligned} &\text{Maximise} \quad -\tfrac{1}{2}\sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i^T \mathbf{x}_j) - \textstyle\sum_{i=1}^{l} \alpha_i \\ &\text{subject to} \quad \begin{cases} \sum_{i=1}^{l} \alpha_i = 0 \\ 0 \leq \alpha_i \leq C \end{cases} \end{aligned}$$

First, two Lagrangian multipliers, α_p and α_q, are selected according to some heuristics, favouring those multipliers that violate the Kuhn-Tucker conditions in theorem 3.1. Now the objective function gets maximised only with respect to these two variables — all other variables remain constant. The domain of (α_p, α_q) is limited to a diagonal line, due to the equality constraint $\sum_{i=1}^{l} \alpha_i = 0$ and bounded by the inequality constraints $0 \leq \alpha_p, \alpha_q \leq C$. This allows an analytical maximisation without the various problems arising from numerical iteration methods. Selection and constrained maximisation are repeated until the Lagrangian multipliers do not change anymore.

Besides sophisticated selection heuristics, there exist numerous speedup methods that improve the performance of SMO. The algorithm does not depend on complicated quadratic programming libraries, but still fails to have the simplicity of a pattern-by-pattern algorithm such as the Perceptron.

4.3 MinOver

The following modification to the Perceptron learning rule, which preserves the simple structure of the algorithm but achieves convergence to the maximal margin classifier, was presented by Krauth and Mézard [7].

4.3.1 Classification

The perceptron learning algorithm terminates as soon as a separating hyperplane is found and it uses the whole training set during each iteration step — usually not resulting in the maximal margin solution.

Input:
$(\mathbf{x}_i, y_i)$, $i = 1, \ldots, l$ — training patterns
t_{max} — number of learning steps
Output:
α, b — dual representation of separating hyperplane

$\alpha \leftarrow \mathbf{0}$
$b \leftarrow 0$
foreach $t \in \{1, \ldots, t_{max}\}$ **do**
  foreach i **do**
    $r_i \leftarrow y_i \left(\sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i) + \sum_{j=1}^{l} \alpha_j y_j \right)$
  end
  $k \leftarrow \arg\min_i \{r_i\}$
  $\alpha_k \leftarrow \alpha_k + 1$
  $b \leftarrow b + y_k$
end

Figure 4.3: MinOver algorithm for classification in dual representation. In contrast to the Perceptron algorithm, the pattern with the worst margin is chosen and learning is continued even if there are no mistakes anymore.

In contrast, the MinOver algorithm (see fig. 4.3) selects, during each learning step, the pattern with the minimal residual margin or overlap — hence the name MinOver. Either, this is the pattern that is strongest misclassified or, if all patterns are correctly classified, the pattern that is closest to the decision plane. Mathematically, a pattern k with

$$k = \arg\min_i y_i \left(\mathbf{w}^T \mathbf{x}_i - b \right)$$

is added to the normal vector $\mathbf{w}$. The bias b is updated by adding the corresponding class label.

Furthermore, the algorithm is no longer terminated as soon as correct classification for all training patterns is achieved, but after a predefined number of learning steps t_{max}. The decision function remains the same as

for the Perceptron:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w}^T\mathbf{x} + b)$$

Again, the usage of the dual representation and kernel functions will enable nonlinear decision borders. With transformations equivalent to (4.2) the dual representation of the MinOver algorithm is derived:

$$\begin{aligned} f(\mathbf{x}) &= \text{sgn}\left(\sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j^T \mathbf{x}_i + \sum_{j=1}^{l} \alpha_j y_j\right) \\ k &= \arg\min_i y_i \left(\left(\sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j\right)^T \mathbf{x}_i + \sum_{j=1}^{l} \alpha_j y_j\right) \end{aligned}$$

A first convergence proof was given by Krauth and Mezard, who showed that MinOver converges at least as $\mathcal{O}(t^{-\frac{1}{2}})$ to the maximal margin classifier. Martinetz showed that MinOver even converges as $\mathcal{O}(t^{-1})$ with increasing number of training steps [12]. Both simplicity and fast convergence qualify MinOver for many classification tasks and there are many further extensions that improve performance.

However, the solution does not only contain support vectors, but also some other training patterns that were selected at an earlier stage of the algorithm.

4.3.2 Regression

By modifying the pattern selection strategy and the update rule, MinOver can easily be extended for regression tasks.

The starting point is the dual representation of the 2-norm soft margin approach (3.14). For convenience, the Lagrangian multipliers are substituted so that $(\alpha_i - \alpha_i^*)$ becomes α_i. Either α_i or α_i^* are 0 because a support vector cannot be simultaenously above and below the regression function. So, $(\alpha_i + \alpha_i)$ are substituted by $|\alpha_i|$. In total, we get the following optimisation problem:

$$\begin{aligned} &\text{Maximise} \quad L(\alpha) = -\tfrac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j \left(K(\mathbf{x}_i, \mathbf{x}_j) + \tfrac{\delta_{ij}}{C}\right) - \epsilon \sum_{i=1}^{l} |\alpha_i| + \sum_{i=1}^{l} y_i \alpha_i \\ &\text{subject to} \quad \sum_{i=1}^{l} \alpha_i = 0 \end{aligned}$$

In order to meet the constraint, we have to select two Lagrangian multipliers within each learning step and increment one of them, while the other gets decremented. The partial derivative gives an idea which multipliers one should select:

$$\frac{\partial L(\alpha)}{\partial \alpha_i} = y_i - \epsilon \operatorname{sgn}(\alpha_i) - f(\mathbf{x}_i) - \frac{\alpha_i}{C}$$

In the optimum the partial derivatives vanish. Thus, a good choice for the next learning step is to increase $\alpha_{i_{min}}$ and to decrease $\alpha_{i_{max}}$ with

$$\begin{aligned} i_{min} &= \arg\min_i \left(y_i - f(\mathbf{x}_i) - \frac{\alpha_i}{C} \right) \\ i_{max} &= \arg\max_i \left(y_i - f(\mathbf{x}_i) - \frac{\alpha_i}{C} \right) \end{aligned}$$

In the classification task only the sign of f was important, but for regression the norm of the weight vector w is significant. So, the learning rate, i.e. the amount of changes within one learning step, must be chosen with care. The convergence proof for MaxMinOver [19], an extension to MinOver with forgetting rule, showed that a learning rate of $\frac{1}{t}$ is a good choice.

Some remarks about the MinOver algorithm 4.4 should be made:

1. The MinOver algorithm for regression does *not* converge to the ϵ-SVR solution in the sense that the discrepancy between exact and approximated solution will become smaller and smaller. Within a certain learning step t' the algorithm may observe that the residuals of all training points are less than ϵ. Hence, the algorithm terminates, allthough not all support vectors may have been identified. The regression function, derived from this MinOver variant, is situated somewhere between a least squares solution for the given training patterns and the ϵ-SVR solution.

2. A number of prefixes for MinOver is used to emphasise different features. The prefix *Soft* indicates that a soft margin approach — no matter which type of soft margin — is used.

   When MinOver for classification was introduced, the class labels were

integrated implicitly into the training patterns with the mapping

$$\mathbf{x}_i \rightarrow y_i \cdot (\mathbf{x}_i \quad 1)^T .$$

Thus, the bias b became part of the weight vector $\mathbf{w}$. A more intuitive way — indicated by the prefix *Double* — is to train with the original patterns and update the bias afterwards. Additionally, not one but two patterns with closest distance to the decision surface are updated — one from each side. However, the Double version does not change the convergence rate.

Input:
$(\mathbf{x}_i, y_i)$, $i \in \{1, \ldots, l\}$ — training patterns
ϵ, C, t_{max} — allowed error, softness, number of learning steps
Output:
α, b — weights of the support vectors and bias

$\alpha \leftarrow \mathbf{0}$
foreach $t \in \{1, \ldots, t_{max}\}$ **do**
  foreach i **do**
    $r_i \leftarrow \sum_{j=1}^{l} (\alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\alpha_i}{C}) - y_i$
  end
  $i_{min} \leftarrow \arg\min_i(r_i)$
  $i_{max} \leftarrow \arg\max_i(r_i)$
  if $r_{i_{min}} - r_{i_{max}} > 2\epsilon$ **then**
    $\alpha_{i_{min}} \leftarrow \alpha_{i_{min}} + \frac{1}{t}$
    $\alpha_{i_{max}} \leftarrow \alpha_{i_{max}} - \frac{1}{t}$
  else
    break
  end
end
$b \leftarrow \frac{1}{2}(r_{i_{min}} + r_{i_{max}})$

Figure 4.4: SoftDoubleMinOver algorithm for regression in dual representation with the 2-norm soft margin approach.

Finally, the prefix *Max* indicates the use of a *forgetting* rule, as described in the next section.

3. The domains of the features, i.e. the single entries of the training vectors, may vary by many orders of magnitude. Using these patterns directly for classification or regression may lead to strange behaviour, as the features with great numeric range dominate those with small range. Thus, all training patterns should have normalised entries, i.e. the range of each feature should be within $[-1, 1]$ or $[0, 1]$. The normalisation to zero mean and unit variance would be another possibility.

4.3.3 Forgetting with MaxMinOver

The main disadvantage of MinOver for regression is the lack of convergence to the support vector solution.

The MaxMinOver algorithm for classification [9, 11] ensures that the solution only contains support vectors and it improves the convergence to the maximal margin solution. This is achieved by introducing a *forgetting rule*. In addition to the pattern with minimal margin, also the pattern with maximal margin $\mathbf{x}_{i_{max}}$ is determined. This pattern is the one that is best classified, so this one should normally not be a support vector and can be *dememorised*. It can be shown that the convergence improves, if the norm of the weight vector is kept small [11]. This is the case if

$$\begin{aligned} \mathbf{w}^T \left(y_{i_{max}} (\mathbf{x}_{i_{max}} - b) - y_{i_{min}} (\mathbf{x}_{i_{min}} - b) \right) &> 4 \cdot R^2 \\ &= 4 \cdot \max_i ||\mathbf{x}_i||. \end{aligned}$$

Thus, whenever the above inequality is satisfied, $\alpha_{i_{max}}$ is decreased by 1, while $\alpha_{i_{min}}$ is increased by 2. With this extension MinOver is still very simple, but will surely converge to the support vector machine solution.

In the regression case, convergence to the ϵ-SVR solution is guaranteed [19, 20]. In the following, the main parts of the algorithm (see fig. 4.5) will be explained in detail:

Input: $(\mathbf{x}_i, y_i), i \in \{1, \ldots, l\}$ — training patterns
ϵ, C, t_{max} — allowed error, softness, learning steps
Output: α, b — Lagrangian multipliers and bias

$\alpha \leftarrow \mathbf{0}$
$R \leftarrow$ radius of minimal enclosing sphere in kernel induced feature space
foreach $t \in \{1, \ldots, t_{max}\}$ **do**
  foreach i **do**
    $r_i \leftarrow \sum_{j=1}^{l}(\alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\alpha_i}{C}) - y_i$
  end
  $i_{min} \leftarrow \arg\min_i(r_i)$
  $i_{max} \leftarrow \arg\max_i(r_i)$
  $i_{minNSV} \leftarrow \arg\min_{i,\,\alpha_i>0}(r_i)$
  $i_{maxNSV} \leftarrow \arg\max_{i,\,\alpha_i<0}(r_i)$
  if $(r_{i_{min}} - r_{i_{max}}) > 2 \cdot \epsilon$ **then**
    if $r_{i_{max}} - r_{i_{maxNSV}} > \frac{g(R)}{t}$ **then**
      $\alpha_{i_{min}} \leftarrow \alpha_{i_{min}} + \frac{1}{t} + \min\left(\frac{1}{t}, \alpha_{i_{minNSV}}\right)$
      $\alpha_{i_{minNSV}} \leftarrow \alpha_{i_{minNSV}} - \min\left(\frac{1}{t}, \alpha_{i_{minNSV}}\right)$
      $\alpha_{i_{max}} \leftarrow \alpha_{i_{max}} - \frac{1}{t} - \min\left(\frac{1}{t}, -\alpha_{i_{maxNSV}}\right)$
      $\alpha_{i_{maxNSV}} \leftarrow \alpha_{i_{maxNSV}} + \min\left(\frac{1}{t}, -\alpha_{i_{maxNSV}}\right)$
    else
      $\alpha_{i_{min}} \leftarrow \alpha_{i_{min}} + \frac{1}{t}$
      $\alpha_{i_{max}} \leftarrow \alpha_{i_{max}} - \frac{1}{t}$
    end
  else if $r_{i_{max}} - r_{i_{maxNSV}} > \frac{g(R)}{t}$ **then**
    $m \leftarrow \min\left(\frac{1}{t}, \alpha_{i_{minNSV}}, -\alpha_{i_{maxNSV}}\right)$
    $\alpha_{i_{minNSV}} \leftarrow \alpha_{i_{minNSV}} - m$
    $\alpha_{i_{maxNSV}} \leftarrow \alpha_{i_{maxNSV}} + m$
  end
end
$b \leftarrow \frac{1}{2}(r_{i_{min}} + r_{i_{max}})$

Figure 4.5: SoftDoubleMaxMinOver for regression.

Multiplier selection is done as in the MinOver algorithm, but additionally the support vectors with largest and smallest residuals are determined. Thus, up to four different multipliers are selected during each learning step.

Dememorisation must be done for two different types of patterns. As support vectors have residuals of exactly ϵ, all points with other residuals must be dememorised. The threshold for this procedure is determined to be

$$g(R) = 4 \cdot R^2 + 16 \cdot (3 \cdot \epsilon^2 + 4 \cdot \epsilon + 1)$$

where R is the minimal enclosing ball containing all training patterns in the kernel-induced feature space [19, 20].

Learning and forgetting rate are set to $\frac{1}{t}$, but as the sum of all dual coordinates must be 0, some effort has to be made to mutually increase and decrease the α_i. When dememorisation takes place, the forgetting rate is upper bounded so that the corresponding α_i vanish. When choosing these rates the algorithm will converge to the support vector solution as $\mathcal{O}(\frac{1}{t})$ [20].

The usage of a forgetting rule is not essential for all regression tasks. Often, the data is very noisy. Then, MinOver will not terminate before the maximum number of learning steps t_{max} is reached, because the selected multipliers can never have a distance of less than ϵ. In this case, the regression function will converge against the support vector solution, but not necessarily with the same support vectors.

4.3.4 SoftDoubleMinOver with Prior Knowledge

Within many applications one would like to incorporate prior knowledge into the regression function. This knowledge is normally not available via exact formulas but rather by significant points. At these points the residuals must vanish or at least be much smaller than the allowed error ϵ at other points.

In the following, a novel extension to MinOver is presented, that incorporates prior knowledge by pseudo-data points and by the usage of a

data-dependent ϵ. Suppose training data is given by the two sets

$$\begin{aligned}\mathcal{D} &= \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathbb{R}\}_{i=1}^{l} \\ \mathcal{P} &= \left\{\left(\mathbf{x}_i^{(P)}, y_i^{(P)}\right) \in \mathcal{X} \times \mathbb{R}\right\}_{i=1}^{m},\end{aligned}$$

where $\mathcal{D}$ is derived by observations and $\mathcal{P}$ is a pseudo-training set describing the prior knowledge. Now, both sets are combined and an additional entry for each pattern indicates how confident this pattern is.

In a first approach, low confidence is indicated by 1 and high confidence by $H \in \mathbb{R}^+$, resulting in the new training set

$$\mathcal{D}' = \left\{\left(\mathbf{x}_1, y_1, 1\right), \ldots, \left(\mathbf{x}_l, y_l, 1\right), \left(\mathbf{x}_1^{(P)}, y_1^{(P)}, H\right), \ldots, \left(\mathbf{x}_m^{(P)}, y_m^{(P)}, H\right)\right\}.$$

The confidence value is meant to directly adjust the allowed residuals and weight the margin. In the hard margin case, all points having confidence value 1 must have residuals less than or equal to ϵ. For prior knowledge points the residuals are upper bounded by $\epsilon' = \frac{\epsilon}{H}$. Thus, the margin of the regression function varies (see fig. 4.6).

In this binary representation of the confidence the separation between prior knowledge and observations is obvious, but this concept can be further extended to arbitrary confidence values. A value $c \in \mathbb{R}^+$ is assigned to each pattern to characterise its importance in advance. There are many ways to derive the confidence value:

1. Whenever the feature space is derived by subjective human observations, it also makes sense to use an additional manual label as a confidence value.

2. If the output values y_i are derived by different measuring instruments, then for each observation the accuracy of the used instrument may be used as a confidence value.

3. In the case of clustered feature spaces, where the values for some clusters are more accurate than for others, it may be useful to partition the space and assign a different confidence value to each cluster.

4. Prior knowledge can be incorporated by high confidence values. But the number and distribution of prior knowledge points should be selected with care. If the restrictions to the learning machine get too

strong, the regression function would only depend on the prior knowledge and not on the remaining training data, which in general is not intended.

4.4 Implementation Issues

Besides the choice of the training algorithm there exist numerous ways to improve machine learning algorithms. The following concepts are not restricted to support vector machines and MinOver, but can be applied to various other algorithms to improve performance.

4.4.1 Chunking

The concept of chunking — again a technique described by Vapnik [22] — helps to solve large quadratic programmes such as the dual representation of the support vector machine. Because for comprehensive training sets the kernel matrix would not fit into the working memory, one needs to split

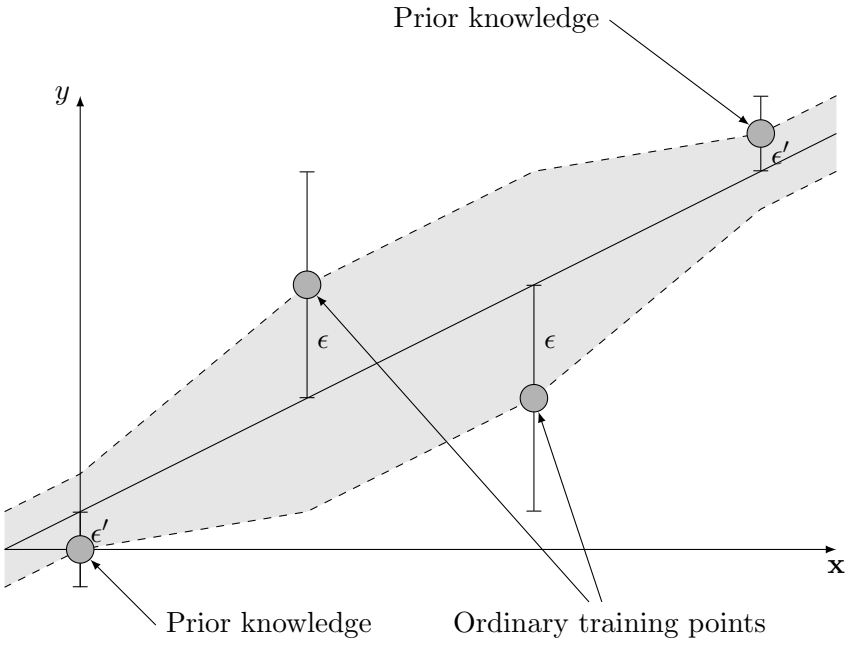


Figure 4.6: Incorporating prior knowledge by weighted margin.

Input:

$(\mathbf{x}_i, y_i, v_i), i \in \{1, \ldots, l\}$ — training patterns and margin weights

ϵ, C, t_{max} — allowed error, softness, number of learning steps

Output:

α, b — Lagrangian multipliers and bias

$\alpha \leftarrow \mathbf{0}$

foreach $t \in \{1, \ldots, t_{max}\}$ **do**

  foreach i **do**

    $r_i \leftarrow v_i \cdot \left(\sum_{j=1}^{l} (\alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\alpha_i}{C}) - y_i - b \right)$

  end

  $i_{min} \leftarrow \arg\min_i(r_i)$

  $i_{max} \leftarrow \arg\max_i(r_i)$

  if $(r_{i_{min}} - r_{i_{max}}) > 2\epsilon$ **then**

    $\alpha_{i_{min}} \leftarrow \alpha_{i_{min}} + \frac{1}{t}$

    $\alpha_{i_{max}} \leftarrow \alpha_{i_{max}} - \frac{1}{t}$

  else

    break

  end

  foreach i **do**

    $r_i \leftarrow \sum_{j=1}^{l} (\alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\alpha_i}{C}) - y_i$

  end

  $i_{min} \leftarrow \arg\min_i(r_i)$

  $i_{max} \leftarrow \arg\max_i(r_i)$

  $b \leftarrow \frac{1}{2}(r_{i_{min}} + r_{i_{max}})$

end

Figure 4.7: SoftDoubleMinOver algorithm with prior knowledge.

the training set into small *chunks*. Normally, only very few support vectors are among the large set of all training patterns. If we knew the support vectors in advance we could train only on this subset and would get the same solution as if training with the whole set. Unfortunately, this is not the case, so by chunking we try to limit the number of non-support vectors in the training set by some heuristics.

Chunking is an incremental procedure that utilises a modified training set within each phase. This set contains all support vectors of the previous phase and additionally N patterns that violate the Kuhn-Tucker conditions most. After the quadratic programme is solved, all non-support vectors are discarded and the next iteration is started. The algorithm terminates when the training set only contains support vectors and no other pattern violates any of the constraints. Thus, even large training sets can be used — assuming that the number of support vectors is comparatively small. However, there exist scenarios where chunking will not work as the set of support vectors is too large. Then, chunking will achieve only approximate solutions.

4.4.2 Decomposition

Osuna [14] proposed another kind of *working set algorithm* that also solves a series of quadratic programmes. First, Osuna showed that adding at least one pattern within each phase that violates the Kuhn-Tucker conditions will improve the objective function. Thus, convergence is already guaranteed for the above chunking method.

But in contrast to chunking, the *decomposition* algorithm operates on a working set of constant size. It starts with an arbitrary subset of training patterns B and solves the subproblem, while keeping the Lagrangians of all other patterns constant. As long as the Kuhn-Tucker conditions are violated by at least one sample $\mathbf{x}_j$ from the remaining set N, an arbitrary sample $\mathbf{x}_i \in B$ is chosen, both are interchanged, and the new subproblem is solved. With this method even problems with many thousands of support vectors can be handled.

4.4.3 Validation

The performance of a learning machine should not only be specified by its execution time, but primarily by its generalisation capabilities. Since by definition of the learning task, only a finite set of samples is available, the generalisation performance must be approximated.

The concept of *cross-validation* is to split the available samples into two sets. The first set is used to train the learning machine, which is then tested on the second set — the *validation set*. Depending on the learning task, different measures to determine the generalisation performance can be applied. Commonly, the percentage of falsely classified patterns is used for classification while the mean squared error determines the quality of regression estimation. Repeating this procedure several times will yield a confidence value to describe the quality of the learning machine.

In *k-fold cross-validation* the initial training set is split into a partition of k subsets, approximately equal in size. Cross-validation is repeated k times, each time using a single subset for validation and all the other $k-1$ subsets for training. Even for small k this process is very time-consuming, but it is an intuitive way to approximate the generalisation performance.

Setting k to the total number of samples leads to *leave-one-out cross-validation*, which is even more time-consuming and seldom used in practice.

4.4.4 Parameter Selection

All support vector approaches take a number of free parameters that have to be chosen in advance. Support vector regression with RBF kernels takes three parameters — the margin ϵ, the softness C, and the standard deviation σ of the kernel.

Grid search is a common but computationally very expensive technique for parameter estimation. The range of each parameter is set to a finite set of values so that the potential parameter combinations form a grid. Now, for every parameter set, the corresponding model — the learning machine — is evaluated by any validation method, resulting in a real number.

Given a support vector regression task with three parameters, each with

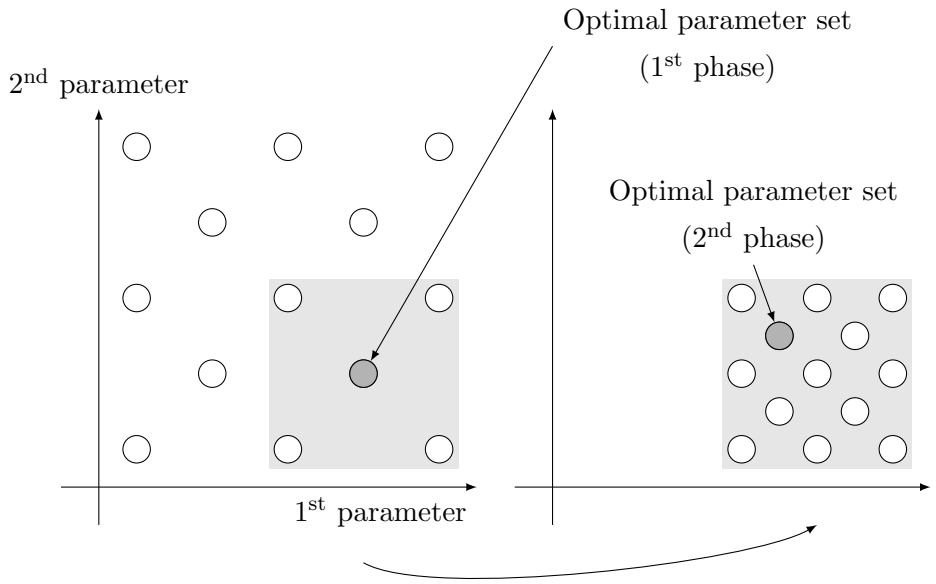


Figure 4.8: Sampling pattern based on design of experiments (DOE). During each search level, $2^d + 3^d$ parameter combinations are chosen (d = number of dimensions, in this case $d = 2$) and the new search space is centered around the best parameter set of the previous search.

10 potential values, the grid search with 5-fold cross-validation will require $10 \cdot 10 \cdot 10 \cdot 5 = 5000$ trainings and evaluations of a learning machine. In practice, the parameters are chosen from a very coarse grid with exponentially increasing steps, e.g. $C = 10^{-4}, 10^{-3}, \ldots, 10^{5}$.

A gradient descent based technique for parameter selection is *pattern search*, also known as *compass search* or *line search*. Starting with an initial parameter, trial steps in each direction with a certain step width are evaluated. If performance improves in any direction, the new parameters are selected to be the center of the next iteration. Otherwise the step width is reduced and the search is repeated. To overcome the problem of local minima, a coarse grid search serves to find a small number of initial points for pattern search.

There exist further more sophisticated parameter selection algorithms, of which only the algorithm proposed by Staelin [21] will be mentioned here.

Based on ideas from design of experiments (DOE), the method starts on a very coarse grid, in fact two nested grids with 3^d and 2^d samples, where d is the number of parameters to optimise in parallel (see fig. 4.8). The best parameter set is chosen and a new grid with doubled resolution and halved range is centered around it. Repeating this procedure results in a robust and fast parameter selection. The parameter, derived by grid search and DOE-base search, may differ significantly, whereas the qualities of the corresponding learning machines are nearly equal. This confirms that the quality measure has normally many local optima.

4.4.5 Outlier Reduction

Allthough the support vector machine is lesser affected by outliers than other machine learning algorithms, the problem may occur that many outliers increase the number of support vectors significantly. Thus, methods to reduce the number of support vectors are desired.

A simple but effective way is to first train the SVM on the whole training set. After having calculated the residuals, the 90^{th} percentile is discarded, and the SVM gets trained again on the reduced training set.

Downs et al. [3] proposed a method for exact simplification of support vector solutions. They identified and discarded those support vectors that were linearly dependent on other support vectors and updated the involved Lagrangian multipliers. The largest reduction of support vectors was possible for RBF kernels with large standard deviation and polynomial kernels with low degree. In some regression problems, up to 40% of the initial support vectors could be omitted without any loss of accuracy. Apparently, linear dependency of support vectors really occurs in practice.

4.4.6 MinOver Optimisation

So far, optimisation strategies have been discussed that do not take the particular structure of MinOver into account. An optimised version of MinOver for classification has been discussed in [9]. Now, the optimisation techniques will be generalised for SoftDoubleMinOver in the regression case. Starting with the algorithm in fig. 4.4, the computationally expensive parts are:

Input:

$(\mathbf{x}_i, y_i, v_i), i \in \{1, \ldots, l\}$ — training patterns and margin weights

ϵ, C, t_{max} — allowed error, softness, number of learning steps

Output:

α, b — Lagrangian multipliers and bias

initialise vector of Lagrangian multipliers α to $\mathbf{0}$

initialise vector of estimates for training patterns $\mathbf{s}$ to $\mathbf{0}$

initialise kernel cache $\mathbf{H}$ to $\mathbf{0}$

initialise cache pointers $\mathbf{p}$ to $\mathbf{0}$

forall $t \leftarrow 1, \ldots, t_{max}$ **do**

calculate residuals by $\mathbf{r} \leftarrow \mathbf{s} - \mathbf{y}$

determine vectors $i_{min} = \arg\min_i(r_i)$ and $i_{max} = \arg\max_i(r_i)$

if $(r_{i_{min}} - r_{i_{max}}) > 2\epsilon$ **then**

forall $i \in \{i_{min}, i_{max}\}$ **do**

determine cache line p_i

if *i is not cached* **then**

delete a cache line if necessary

set cache pointer p_i to free cache line

set cache line $\mathbf{H}_{p_i}$ to $(K(\mathbf{x}_1, \mathbf{x}_{p_i}), \ldots, K(\mathbf{x}_l, \mathbf{x}_{p_i}))$

end

update estimates $\mathbf{s} \leftarrow \mathbf{s} + \frac{1}{t} \cdot \mathbf{H}^T_{p_{i_{min}}} - \frac{1}{t} \cdot \mathbf{H}^T_{p_{i_{max}}}$

update estimate $s_{i_{min}} \leftarrow s_{i_{min}} + \frac{1}{t \cdot C}$

update estimate $s_{i_{max}} \leftarrow s_{i_{max}} - \frac{1}{t \cdot C}$

$\alpha_{i_{min}} \leftarrow \alpha_{i_{min}} + \frac{1}{t}$

$\alpha_{i_{max}} \leftarrow \alpha_{i_{max}} - \frac{1}{t}$

else

terminate

end

end

determine bias by $b \leftarrow \frac{1}{2}(r_{i_{min}} + r_{i_{max}})$

Figure 4.9: Optimised variant of SoftDoubleMinOver. The concept of a kernel cache and the optimised evaluation of the regression function can be incorporated into all mutations of MinOver.

1. The *evaluation of the kernel function* has to be repeated many times. In the worst case large parts of the kernel Matrix $(K(x_i, x_j))_{i,j=1}^{l}$ have to be recalculated during each learning step. As mentioned in previous sections, the complete storage of the kernel matrix in the working memory is not feasible for large training sets. So, a method is needed to minimise the number of kernel evaluations and to limit the memory footprint.

2. During each learning step the *sum of weighted kernels* is recalculated in order to get the residuals. However, only two weights differ between consecutive steps, so that the evaluation of the regression function can indeed be optimised.

3. Another expensive step is the *determination of the minimal and the maximal residual*. Both values have to be recalculated during each learning step, because all residuals may change within one iteration.

Optimised Evaluation of the Regression Function

The residual r_i for an arbitrary input vector $\mathbf{x}_i$ is determined by

$$r_i = \sum_{j=1}^{l} (\alpha_j K(\mathbf{x}_i, \mathbf{x}_j)) + \frac{\alpha_i}{C} - y_i - b.$$

The bias b does not depend on the training pattern, hence it is omitted during the learning process:

$$\begin{aligned} r_i &= \sum_{j=1}^{l} \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) - y_i \\ &= s_i - y_i. \end{aligned}$$

The number of arithmetic operations scales like $\mathcal{O}\left(t_{max} \cdot l^2\right)$. Within every iteration step only the two Lagrangian multipliers $\alpha_{i_{\min}}$ and $\alpha_{i_{max}}$ are modified. So, by storing the sums $s_1, \ldots, s_l$ within every step, they only have to be updated by

$$\begin{aligned} s_i = s_i &+ \frac{1}{t}\left(1 + \frac{\delta_{i\, i_{min}}}{C}\right) K(\mathbf{x}_{i_{min}}, \mathbf{x}_i) \\ &- \frac{1}{t}\left(1 + \frac{\delta_{i\, i_{max}}}{C}\right) K(\mathbf{x}_{i_{max}}, \mathbf{x}_i). \end{aligned}$$

Initialising the sums with 0 and applying the above iteration procedure will reduce the number of operations to $\mathcal{O}(t_{max} \cdot l)$. In order to avoid numerical errors, the complete sum should be recalculated after a fixed number of steps T. Selection of a large T, e.g. 1000, will not change the runtime significantly.

Kernel Caching

In general, the solution of a support vector machine only consists of a few support vectors. After a certain number of learning steps the SoftDoubleMinOver algorithm will only modify a small set of Lagrangian multipliers, and therefore only a few kernel values will be accessed again and again to evaluate the above sum.

The kernel function values that had already been evaluated are stored for later reuse in the *kernel cache* $\mathbf{H}$, consisting of h lines, each with l entries. Additionally, a vector $\mathbf{p}$ contains a pointer to the corresponding cache line for each training pattern. The pseudo cache line 0 indicates that no line has been cached for this pattern.

$$\mathbf{H}_{p_i,j} \quad = \quad K(\mathbf{x}_{p_i}, \mathbf{x}_j)$$

If there is no free cache line, a replacement algorithm discards one of the cache lines and replaces it by the new one. Common replacement algorithms are *Random*, *Least Recently Used* or *First-In First-Out*. A good choice for the cache size h would be the number of support vectors to keep the memory footprint small and to achieve a good speedup. Since the number of support vectors is not known in advance, h has to be determined experimentally.

Forgetting Rules and Prior Knowledge

The mentioned concepts can be applied in a straight-forward manner to the SoftDoubleMaxMinOver algorithm to achieve the benefits of a forgetting rule. Whenever the kernel function values of a vector are needed — either to amplify or reduce the influence of a support vector — they are stored in the kernel cache. After an α_i was set to 0 the corresponding cache line is freed. The sums are updated analogously to the SoftDoubleMinOver algorithm.

The incorporation of prior knowledge via pseudo-training sets and a margin weight $\mathbf{v}$ does not affect these techniques. The kernel cache and the sum of weighted kernel values can be administrated in the same way as before.

The SoftDoubleMinOver algorithm with kernel caching and the optimised evaluation of the regression function is shown in fig. 4.9. The implementation of other mutations of MinOver is straight-forward. In total, the number of operations scales like $\mathcal{O}(t_{max} \cdot l)$; the memory footprint depends mostly on the size of the kernel cache and scales like $\mathcal{O}(h \cdot l)$.

Chapter 5

Application to a Hot Rolling Mill

The mathematical foundations for machine learning theory are well understood, convergence proofs can be given and equations for upper bounds of the generalisation error exist. But the practical relevance of these methods can only be evaluated by applying them to a variety of different problems.

In the case of the blooming train the input variables are noisy and only very few sensory inputs are actually available. First, the data streams from different sources are associated and the relevant features are extracted for each single pass. The utilisable features are selected and plausibility checks discard improper data. A 2-layer architecture for regression estimation is introduced to incorporate prior knowledge and to allow local adaptations.

Details about the implementation and integration of the core learning algorithm and the automisation framework at the rolling mill will conclude the chapter.

5.1 Feature Extraction

Before the regression estimators for force, torque, spread, and reduction can be trained, a sufficient number of training vectors needs to be extracted from the core data that is stored by the material tracking system.

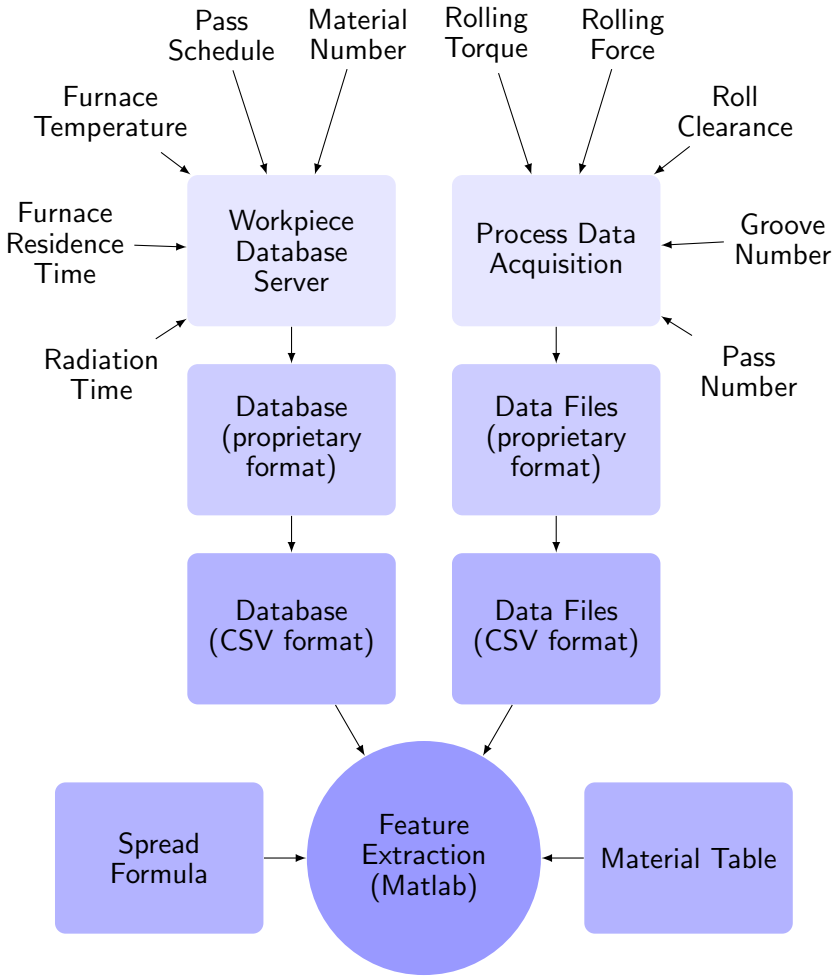


Figure 5.1: Association of different data sources.

5.1.1 Process Data Acquisition

The process data at the blooming train, obtained by various digital and analogue sensors, is processed and stored by the *Process Data Acquistion* (PDA) workstation. For each slab a new data file is generated, labelled with a unique identifier — the so-called *working plan number*. Additional information, such as material, pass schedule, and furnace temperature, is stored in a workpiece database. The PDA and the workpiece database use proprietary file formates, which first are extracted to comma-separated values. Then, a Matlab function combines information from data files and database,

Material number	DIN label	High-temperature strength
1.2343	X38CrMoV5 1	2.08
1.2344	X40CrMoV5 1	2.38
1.2360	X48CrMoV8-1-1	2.22
1.2390	X32CrMoV4 1	1.85
1.2394	X32CrMoV4 2	1.99
1.2397	32MoCrNi20 4 2	1.76
1.2419	105WCr6	1.77
1.2510	100MnCrW4	2.15
1.2690	X29CrMoW6 1 1	2.12
1.2703	74NiCr2	1.49
1.2704	74NiCr4	1.40
1.2791	D6A (48CrMoNi49)	1.65
1.2795	65NiCrMo3 2	1.54

Figure 5.2: Example of the material table. The material number uniquely identifies the DIN label and a value for the high-temperature strength.

and extracts a feature vector for each pass (see fig. 5.1).

5.1.2 Material

The material property that influences the rolling process the most is the high-temperature strength — a measure for the hardness of the material at the rolling temperature.

Each steel grade is identified by a unique material number according to common industrial standards. All slabs with the same chemical composition, and therefore the same material number, have identical high-temperature strengths. The available hardness testing methods, such as *Vickers hardness test* or the *Rockwell scale*, are based on the ability of the material to resist indentation from a standard source. The particular measuring method at Buderus is confidential, but a lookup-table of 170 material numbers and corresponding high-temperature strength values was provided (see fig. 5.2). Buderus certifies that these values are reproducible and do not vary among different charges.

5.1.3 Groove and Pass Number

The groove number not only determines the shape of the workpiece, but it influences also the occuring forces. A lot of parameters would have to be introduced to describe the shape of the groove in total, but since only 5 grooves are available, the groove number is used for simplicity.

Throughout the rolling process the current groove number and the pass number are recorded by the process data acquisition at a resolution of 10 ms.

5.1.4 Workpiece Dimensions

In geometric terms the slab is a *frustrum*. During each pass the volume of this frustrum remains the same while thickness is reduced and width is increased. In practice, the geometry of head and foot are different for the first six passes. So, for training the mean values of thickness and width will be used. By modelling the geometry of the slab and its deformation during each pass, these mean values are derived by the following procedure:

1. Determine the initial geometry of the slab's head ($\overline{h}_0$ and $\overline{w}_0$) by the pass schedule and append the missing foot dimensions ($\underline{h}_0$ and $\underline{w}_0$) from a lookup-table (see fig. 2.1).

   Unfortunately, the data files delivered from the PDA do not contain any information about the applied pass schedule and the entry dimensions of the slab. However, this information is stored in the workpiece database. Up to now, the schedule name contains only geometries, but no information about the material. Two slabs with identical dimensions but different high-temperature strength are rolled by the same schedule. Thus, either the blooming train does not operate at full capacity or the wear of the rolls is increased dramatically.

2. Set the entry dimension — entry thickness h_0 and entry width w_0 — to the mean values of thickness and width respectively:

$$h_0 = \frac{1}{2}(\overline{h}_0 + \underline{h}_0) w_0 = \frac{1}{2}(\overline{w}_0 + \underline{w}_0)$$

3. Determine current groove g and roll clearance d from the data of the current pass.

4. Determine the sum s of groove depth and roll clearance. If the slab's head thickness is larger than s, set the new thickness to s. Proceed the same way with the foot thickness.

5. Given the new thicknesses, calculate both width values of the slab.

   Unfortunately, the exit width is not measured, so learning from examples to predict the spread is not feasible. But some approximation formulas exist which yield a basic prediction of the spread, though they do not take the material influence into account. According to literature [6], these formulas may only be applied to free spread and not to bounded spread. Therefore, manual measurements were made to determine a reliable spread approximation formula that also works for bounded spread. However, the rolling mill crew measured only a few workpieces this way, because of the unacceptable working conditions near the hot slab. Besides, this procedure extends the rolling time, and thus influences the temperature of the slab. So, this method gives only a rough estimate of the spreading process.

6. Set the exit dimensions — exit thickness h_1 and exit width w_1 — to the mean values of thickness and width respectively.

7. If the slab is tilted after the pass, swap thickness and width of the model.

8. Continue with step 2 until no passes are left.

5.1.5 Furnace Temperature, Residence Time, and Radiation Time

Besides the high-temperature strength, also the temperature distribution within the slab is an important material parameter. This distribution cannot be determined, as no sensors may be placed *inside* the workpiece. An approach to approximate the inner temperature distribution is to measure furnace residence time, furnace temperature, and radiation time.

The furnace residence time takes values between 10 and 80 hours. The longer the workpiece is heated, the more likely it has an equally distributed temperature, which would be best for the rolling process.

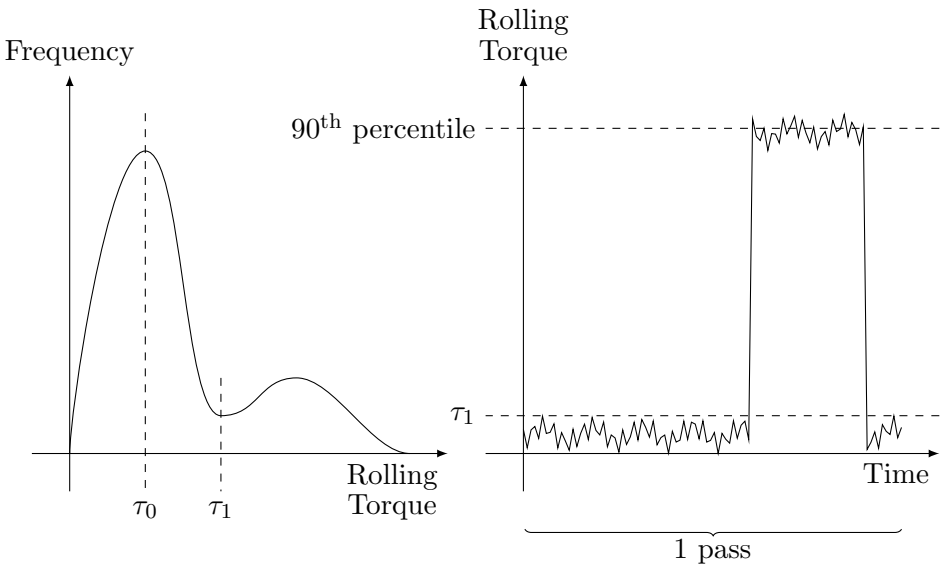


Figure 5.3: Extraction of rolling torque value.

The regular furnace temperature is about 1200 to 1300 °C, but varies with the filling degree of the furnace and the material of the workpieces.

The radiation time of a workpiece describes the time elapsed since the furnace exit. With increasing radiation time the workpiece will cool down, and the hardness will increase so that the rolling torque and force values will be larger. Experience shows that the mean temperature loss of the workpiece is about 0.2 °C/s. A temperature reduction of 100 °C causes an increase in hardness of 40 %. However, this gives only a very rough estimate of the real situation.

All three temperature-affecting parameters can be obtained directly from the workpiece database.

5.1.6 Rolling Torque

The rolling torque measuring device determines the tension and compression of the axis on which the rolls are mounted as an integer value with a resolution of 10 ms. This method is affected by temperature drift and vibrations so that large variances occur. The following procedure compresses the many

hundred torque values during one pass to a single value that best describes the largest value that is not an outlier.

1. Looking at the processing time of a whole slab, the entire rolling time is very short. During the remaining time — necessary for tilting, moving and reversing the slab — only an *idle rolling torque* is measured. In order to determine the periods of rolling, all intervals with idle rolling torque must be identified.

   First, the most frequent rolling torque value τ_0 within a complete rolling procedure is determined. The threshold between idle and non-idle is set to $\tau_1 = 2\tau_0$ for all passes within this data file (see fig. 5.3).

2. Within each pass the longest continuous interval with rolling torque values larger than the threshold is detected. Short intervals with a duration of less than 200 ms are discarded, as passes always take longer.

3. The 90^{th} percentile of the up to 500 values in the remaining interval is selected as the rolling torque value of the current pass.

5.1.7 Rolling Force

The rolling force can be measured in three different ways:

1. The rolling force can be determined via the rolling torque according to equation (2.2). Since the rolling torque is strongly affected by noise, this should only be done when no force measurement device is available.

2. Strain gauges to measure the rolling force are mounted at the stand of the rolls, but they own the same disadvantages as the torque measurement. Additionally, these strain gauges are susceptible to damages during the weekly roll change.

3. The probably most accurate and robust measuring technique uses load cells and a complicated power transmission system. The only disadvantage of these load cells is the much longer reaction time in relation to strain gauges.

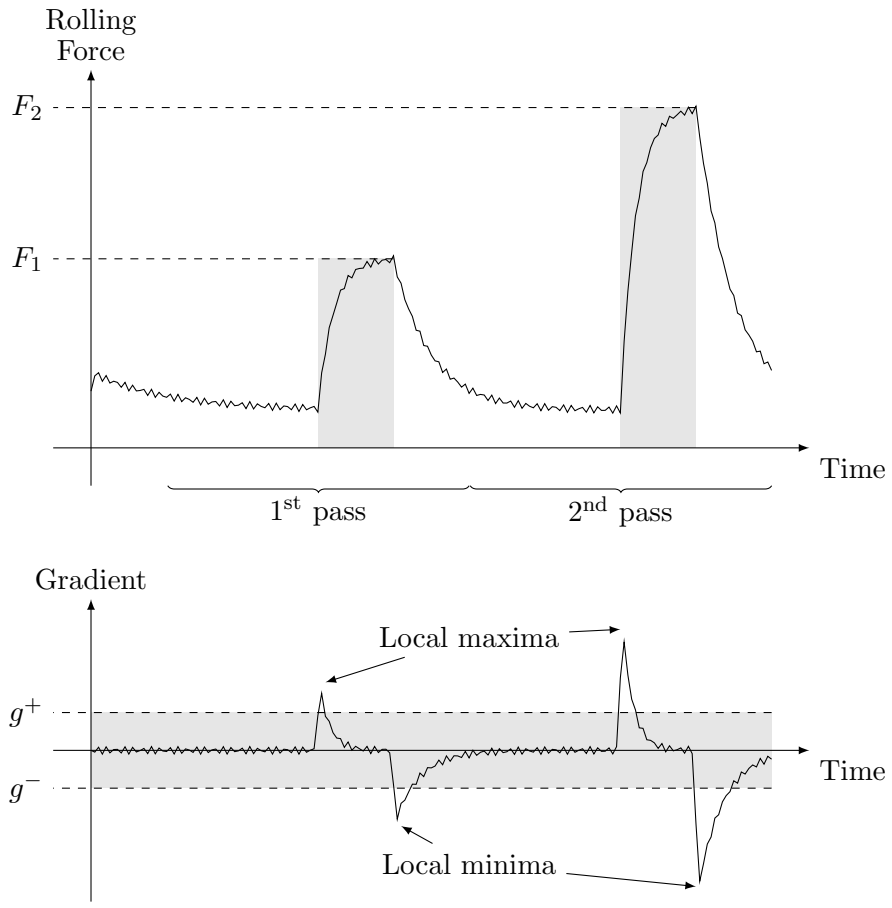


Figure 5.4: Extraction of rolling force values. The sequence between rising and falling edge is determined by selecting adequate minima and maxima of the gradient. Only those minima and maxima are selected which are below or above the thresholds g^- and g^+ respectively to reduce the influence of noise.

The first method was only used as long as no direct measuring devices were installed, since it delivers no new information. The second method is far too inaccurate because of various unpredictable influences on the sonsory inputs. The following procedure determines a value that best describes the maximal occuring rolling force across one pass, based on data obtained by the load cells, again at a resolution of 10 ms. Suppose, the data, obtained by the load cells at a resolution of 10 ms, is given by the series f.

1. Determine the gradient-like series

$$g = f * \left(1 \quad \underbrace{0 \quad \cdots \quad 0}_{2k+1} \quad -1 \right) \tag{5.1}$$

with an apropriate k (≈ 5).

2. Detect all local maxima of g above a certain threshold g^+ and all local minima below a threshold g^+.

3. Find the longest sequence within one pass that is enclosed by a maximum on the left and a minimum on the right (see fig. 5.4).

4. Determine the maximum value of f within this sequence. If the sequence is too short, set the maximum value to zero, since the pass was probably an empty pass.

5.1.8 Determination of a Spread Formula

The relative spread of a workpiece is defined to be

$$C = \frac{\Delta w}{\Delta h} = \frac{w_1 - w_0}{h_0 - h_1}.$$

Since no continuous width measurements are available, approximation formulas have to be used. A number of more or less complicated spread formulas has been developed and used in practice [6], of which four will be mentioned here:

$$\Delta w = k \cdot \Delta h \quad \text{with } k = 0.35 \tag{Geuze}$$

$$w_1 = w_0 + \frac{\Delta h}{6}\sqrt{\frac{2 \cdot r}{h_0}} \tag{Tafel and Sedlaczek}$$

$$w_1 = w_0 + \frac{w_0\sqrt{w_0 \cdot r}\Delta h}{3\left(w_0^2 + h_0 h_1\right)} \tag{Tafel, Sedlaczek, and Emicke}$$

$$w_1 = w_0 \left(\frac{h_0}{h_1}\right)^W \text{ with } W = 10^{-1.269\left(\frac{w_0}{h_0}\right)\left(\frac{h_0}{2r}\right)^{0.556}} \tag{Wusatowski}$$

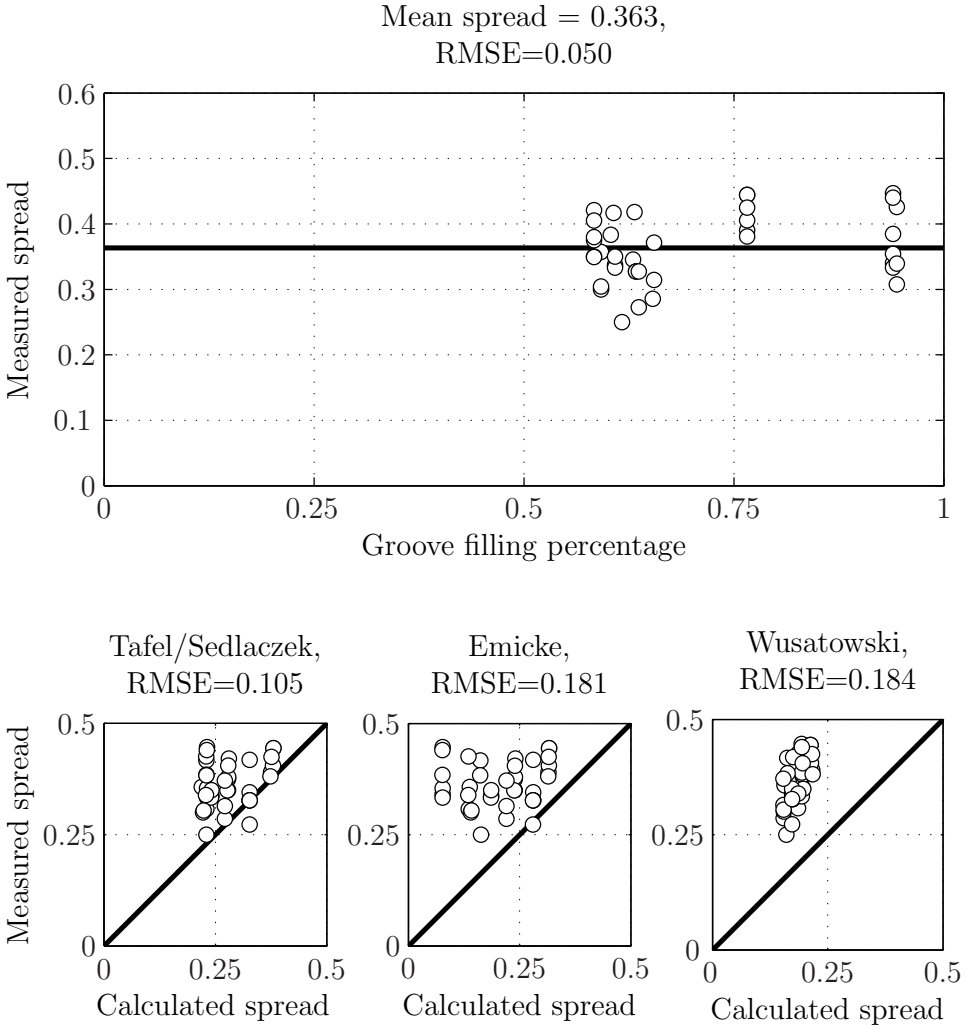


Figure 5.5: Errors of different spread formulas. The top figure reveals that the relative spread is nearly constant for different groove filling degrees. There is no effect of bounded spread observable. The bottom figures show the deviation of computed from measured spread for three formulas. In general, these formulas predict a smaller spread than actually was observed. The root mean square error (RMSE) specifies the numerical deviations.

Manual measurements of 6 blocks throughout the whole rolling process were made and the effective spread was compared to the computed spread. According to the measurements, the relative spread at the blooming mill is nearly constant for different entry widths. It became apparent that rolling

with bounded spread has actually no observable effect on the maximum exit width. The mean spread was 0.363, which is nearly the faktor k in the formula of Geuze. All other formulas showed much larger errors (see fig. 5.5). Often they predicted a significant smaller spread than was observed.

Thus, the spread is calculated using Geuze's formula, after having determined entry and exit thickness by geometric means from the process data.

5.1.9 Plausibility Checks

The feature extraction provides a set $\mathcal{V}$ of vectors with 12 entries:

$$\begin{pmatrix} \text{pass number} \\ \text{groove number} \\ \text{high-temperature strength} \\ \text{entry thickness} \\ \text{entry width} \\ \text{exit thickness} \\ \text{exit width} \\ \text{radiation time} \\ \text{furnace radiation time} \\ \text{furnace temperature} \\ \text{rolling torque} \\ \text{rolling force} \end{pmatrix} = \begin{pmatrix} p \\ g \\ \alpha \\ h_0 \\ w_0 \\ h_1 \\ w_1 \\ t_{rad} \\ t_{furn} \\ T \\ \tau \\ F \end{pmatrix}$$

This set also contains vectors that are not suitable for learning. Data of passes that violate the constraints described in section 2.4 should be discarded to avoid confusion of the learning machine. So, feature vectors where at least one value violates the predefined ranges or constraints were discarded as well as vectors that describe empty passes.

5.2 Feature Selection

Now, as the features have been extracted, we have to determine which of them are suitable for the learning task, and which have to be discarded. At least the global tendencies are known for all input variables. For example,

with increasing radiation time, the slab cools down and hardness increases which finally should require greater rolling torque.

Whenever an input variable has no observable influence on the output or the sensory inputs contradict the prior knowledge, it would be better not to use this variable to keep the estimators robust.

5.2.1 Temperature-Affecting Parameters

The influence of furnace temperature, furnace residence time, and radiation time on the rolling torque will now be further investigated. It is clear that these parameters have an influence, but it is unknown whether the current measuring techniques are accurate enough to detect these effects. With increasing furnace temperature and residence time the rolling torque within the first pass should decrease. In contrast, a long radiation time should lead to larger rolling torque values. In the following, these influences will be quantified. To avoid the effect of the rolling process on the temperature, only inputs derived from the first pass of each slab were analysed.

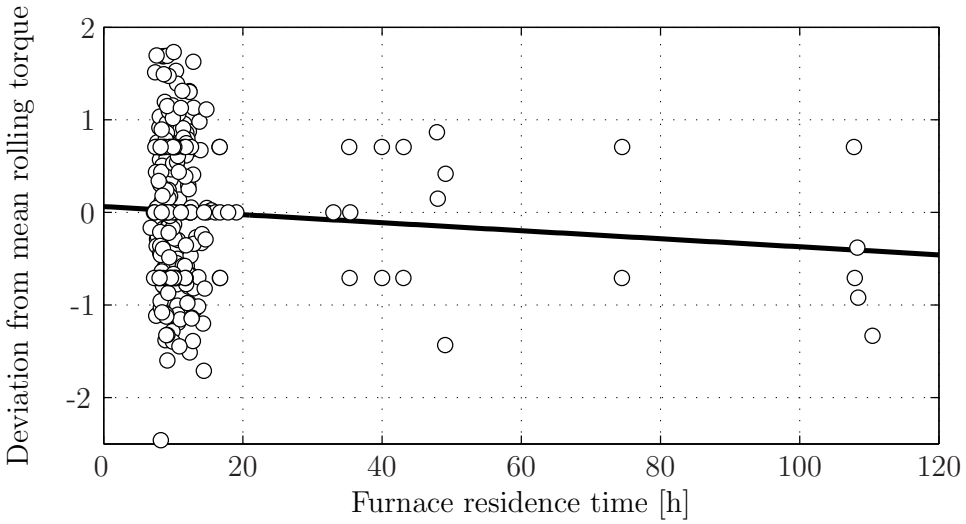


Figure 5.6: Influence of furnace residence time on the rolling torque. Each dot indicates the deviation from the mean rolling torque (of all slabs with the same material and scheduling parameters) within the first pass. The deviation is measured in multiples of the standard deviation. Additionally, the line of best fit is shown.

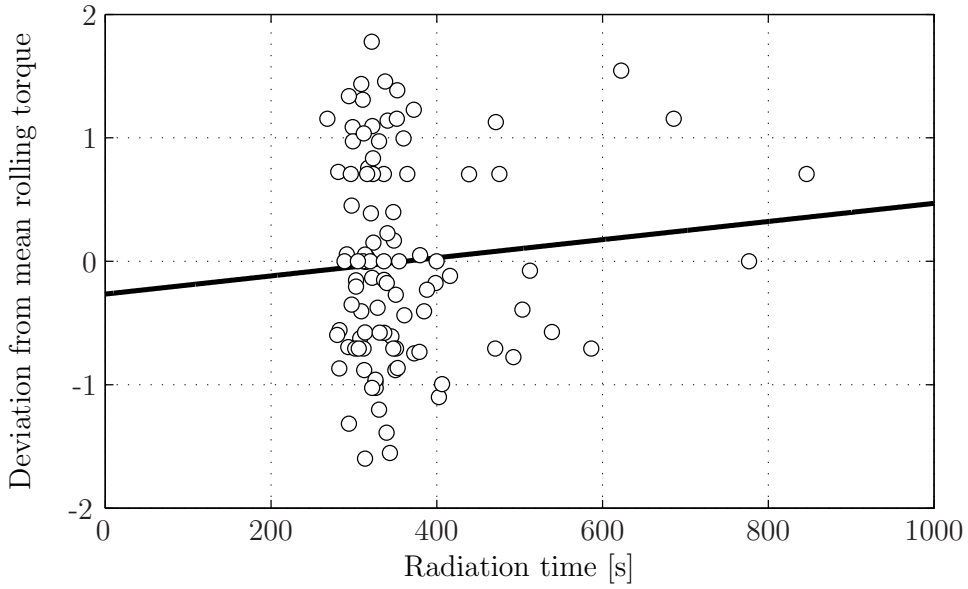


Figure 5.7: Influence of radiation time on the rolling torque. Again, the deviation from the mean rolling torque and the line of best fit are shown.

First, for each combination of pass schedule and material number the mean rolling torque value was determined. Systematic deviations from this mean rolling torque, when changing only one input variable, would indicate that this input indeed has an effect.

For analysing the furnace residence time, all input vectors with the most frequent furnace temperature (1300 °C $\leq T \leq$ 1305 °C) and radiation time (300 s $\leq t_{rad} \leq$ 360 s) were used. These intervals were kept as small as possible, so that temperature and radiation time can be considered to be equal for all input vectors. With increasing furnace residence time the rolling torque does not significantly deviate from the mean, although the linear least squares regression line shows a slightly negative gradient (see fig. 5.6). Using other furnace temperatures and radiation times does not help to find any tendencies.

Next, the radiation time of the feature vectors with the most frequent furnace temperature and furnace residence time (10 h $\leq t_{furn} \leq$ 12 h) was plotted against their deviation from the mean rolling torque. As proposed, the necessary rolling torque increases with rising radiation time (see fig. 5.7).

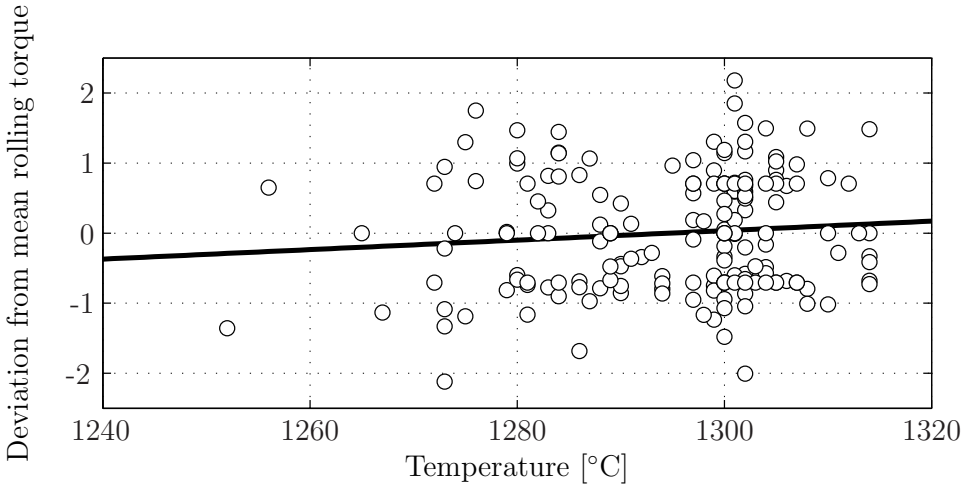


Figure 5.8: Influence of furnace temperature on the rolling torque. Again, the deviation from the mean rolling torque and the line of best fit are shown.

But there is no significant dependency because omitting only a few data points leads to a negative gradient.

The third parameter — the furnace temperature — was plotted against the deviation from the mean torque for the most frequent residence and radiation times. Again, there was no significant dependency observable (see fig. 5.8).

All three temperature affecting parameters seemed to influence the rolling torque slightly but the effects were not significant. Thus, the parameters were discarded and a preprocessing step was introduced. As described above, the mean temperature loss is about 0.2 °C/s, and a temperature loss of 100 °C increases the high-temperature strength by about 40%. The following formula is used to approximate the modified high-temperature strength:

$$\begin{aligned} \alpha' &= \alpha \cdot \left(0.2 \cdot \frac{0.4}{100} \cdot t_{rad} + 1\right) \\ &= \alpha \cdot (0.0008 \cdot t_{rad} + 1)\,. \end{aligned}$$

5.2.2 The Training Sets

Unfortunately, the number of available sensory inputs is very limited and some of the inputs — all temperature-affecting parameters — are not usable.

As the groove number is a discrete parameter for which no intuitive order exists, it does not make sense to use it on a continuous scale. Thus, for every groove a set of estimators is trained. Thus, the pass schedule calculator needs four different sets of estimators:

1. The rolling torque, necessary to achieve a certain thickness reduction, has to be predicted without knowing the rolling force. Thus, the training set

$$\mathcal{D}_{\tau,g} = \left\{ (\mathbf{x}_i, y_i) \,\middle|\, \mathbf{x}_i = \left(\alpha^{(i)}\ w_0^{(i)}\ \Delta h^{(i)}\right)^T,\ y_i = \tau^{(i)},\ g^{(i)} = g \right\}$$

   will be used for groove number g. Again, $\Delta h^{(i)} = h_0^{(i)} - h_1^{(i)}$ is the thickness reduction.

2. The same features are used to predict the rolling force without knowledge about the rolling torque.

$$\mathcal{D}_{F,g} = \left\{ (\mathbf{x}_i, y_i) \,\middle|\, \mathbf{x}_i = \left(\alpha^{(i)}\ w_0^{(i)}\ \Delta h^{(i)}\right)^T,\ y_i = F^{(i)},\ g^{(i)} = g \right\}$$

3. Vice versa, the thickness reduction can either be determined via rolling torque or rolling force.

$$\mathcal{D}_{\Delta h(F),g} = \left\{ (\mathbf{x}_i, y_i) \,\middle|\, \mathbf{x}_i = \left(\alpha^{(i)}\ w_0^{(i)}\ F^{(i)}\right)^T,\ y_i = \Delta h^{(i)},\ g^{(i)} = g \right\}$$

$$\mathcal{D}_{\Delta h(\tau),g} = \left\{ (\mathbf{x}_i, y_i) \,\middle|\, \mathbf{x}_i = \left(\alpha^{(i)}\ w_0^{(i)}\ \tau^{(i)}\right)^T,\ y_i = \Delta h^{(i)},\ g^{(i)} = g \right\}$$

So, each training set consists of three-dimensional input vectors and a corresponding output value. Additionally, all input dimensions and the output value are scaled to $[0, 1]$ by dividing each variable by the maximum.

5.3 Two Approximation Layers

A major problem when using neural networks or support vector machines for regression estimation is the behaviour in regions where sparse or no data is available. The regression function may lead to very good results in areas

where lots of training points are available, whereas it might show chaotic behaviour in all other areas. At the blooming train, the predictors should yield at least the correct tendencies for areas without any data points. The tendencies are derived from prior knowledge. Therefore, two approximation layers are constructed by the following procedure:

1. Determine a monotonic regression function $f_{simple}(\mathbf{x})$ that minimises the mean squared error as a global approximation.

   The reason for monotonicity is the following prior knowledge about the physical dependencies of the input variables. Expansion of equations (2.1) and (2.2) leads to the following approximations:

$$\begin{aligned} F &= k_{rm} \cdot A_c \\ &= k_{rm}(\alpha) \cdot l_d \cdot \bar{b} \\ &\approx k_{rm}(\alpha) \cdot \sqrt{r \cdot \Delta h} \cdot \frac{w_0 + 2w_1}{3} \end{aligned}$$

$$\begin{aligned} \tau &= 2 \cdot a \cdot l_d \cdot F \\ &\approx 2 \cdot a \cdot k_{rm}(\alpha) \cdot r \cdot \Delta h \cdot \frac{w_0 + 2w_1}{3} \end{aligned}$$

   The mean deformation resistance k_{rm} increases strictly monotonically, thus, F and τ are strictly monotonically increasing with respect to Δh, α, and w_0. Thickness reduction Δh increases strictly monotonically with respect to F (or τ) while decreasing strictly monotonically with respect to α and w_0. In detail, the monotonic regression functions were:

$$\begin{aligned} f_{F,simple}(\alpha, w_0, \Delta h) &= c \cdot \alpha \cdot w_0 \cdot \sqrt{\Delta h} \\ f_{\tau,simple}(\alpha, w_0, \Delta h) &= c \cdot \alpha \cdot w_0 \cdot \Delta h \\ f_{\Delta h(F),simple}(\alpha, w_0, F) &= c \cdot \left(\frac{F}{\alpha \cdot w_0} \right)^2 \\ f_{\Delta h(\tau),simple}(\alpha, w_0, \tau) &= c \cdot \frac{\tau}{\alpha \cdot w_0} \end{aligned}$$

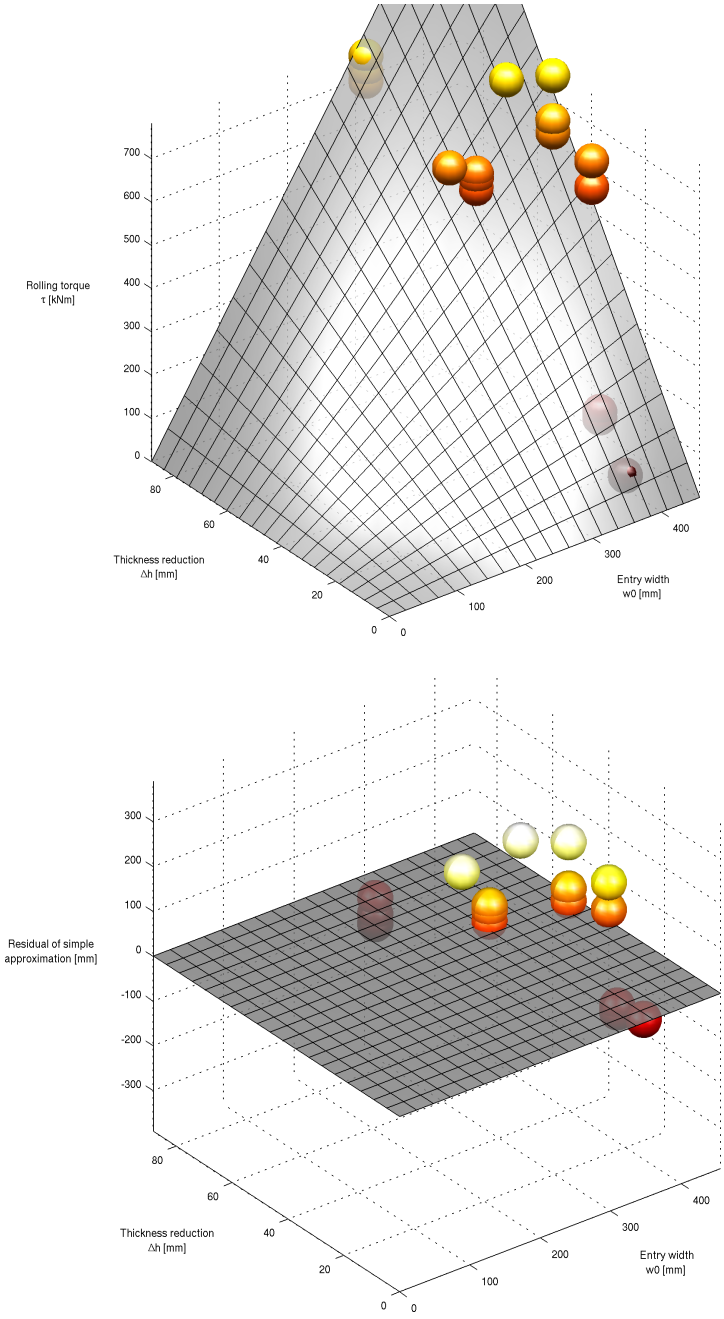


Figure 5.9: Simple approximation (first groove). Each ball represents one pass; the colors indicate high (yellow) or low (red) rolling torque values. The least-squares solution of the simple equation $f_{simple} = c \cdot \alpha \cdot \Delta h \cdot w_0$ for $\alpha = 1.31$ (top) predicts the correct tendencies, allthough the residuals are rather large (bottom).

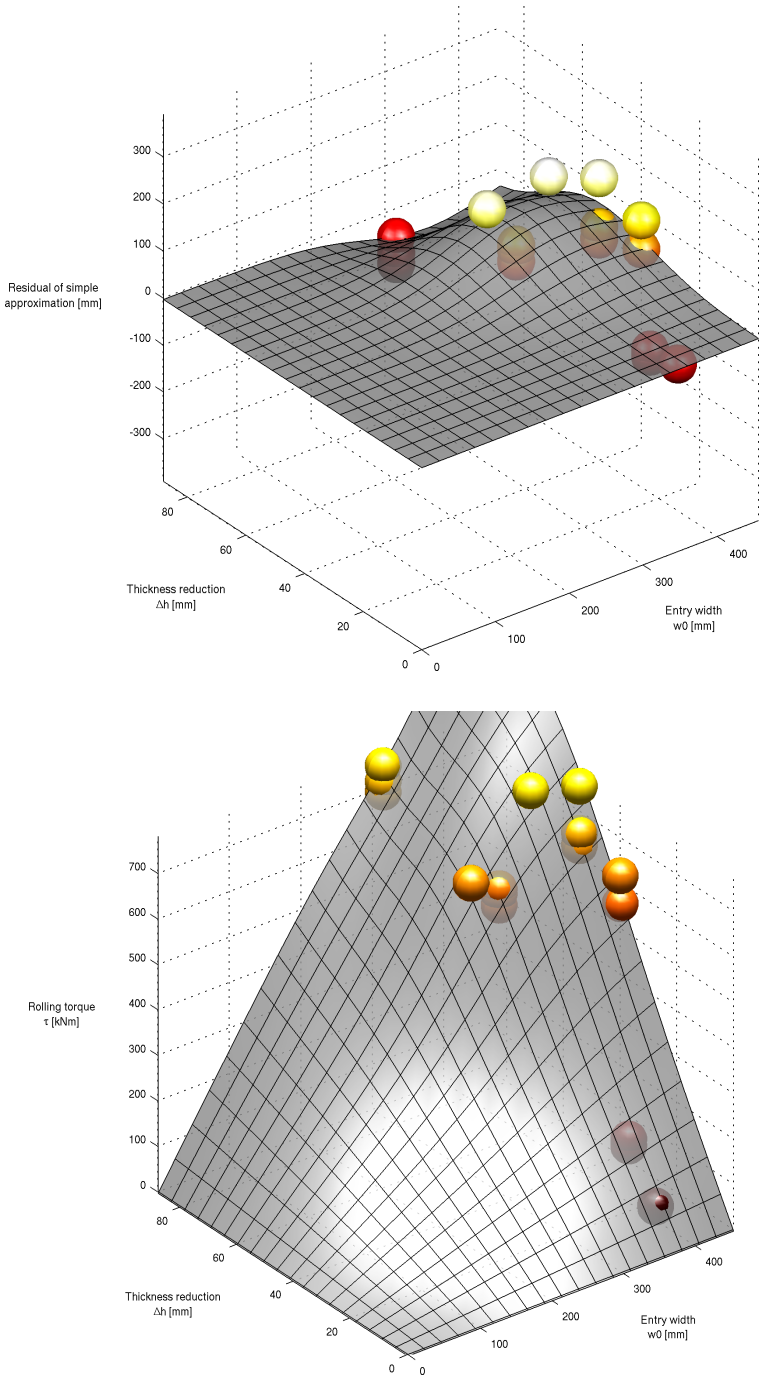


Figure 5.10: Reducing the residuals by support vector regression. A support vector machine with Gaussian kernels is trained on the residuals (top), then both approximations are added (bottom).

Because all input parameters are greater or equal to zero, the prior knowledge about F being proportional to $\sqrt{\Delta h}$ (see equation (5.2)) can be incorporated into $f_{F,simple}$ and $f_{\Delta h(F),simple}$ (see fig. 5.9).

2. Augment the training sets by pseudo-data points to incorporate more prior knowledge.

   In this case, prior information is required to pull the regression surface to zero when certain input variables are zero:

$$\begin{aligned} \Delta h = 0 &\Leftrightarrow F = 0 \\ \Delta h = 0 &\Leftrightarrow \tau = 0 \\ w_0 = 0 &\Rightarrow F = 0 \\ w_0 = 0 &\Rightarrow \tau = 0 \end{aligned}$$

   Therefore, pseudo-data sets are introduced for each output variable, groove number. When estimating the rolling force, an appropriate data set would be:

$$\begin{aligned} D^{+}_{F,g} = &\left\{ \left((\alpha\ w_0\ 0)^T, 0 \right) \middle| \alpha, w_0 = 0, \frac{1}{t}, \ldots, 1 \right\} \cup \\ &\left\{ \left((\alpha\ 0\ \Delta h)^T, 0 \right) \middle| \alpha, \Delta h = 0, \frac{1}{t}, \ldots, 1 \right\} \end{aligned}$$

   The number t of steps for each dimension must not be too large, so that the number of pseudo-data points is small related to the number of all data points. A value of $t = 10$ is a good choice when dealing with a few thousand data points. All other data sets are constructed analogously and the original data sets are augmented by their corresponding pseudo-data sets.

3. Determine the residuals of the simple approximation by subtracting $f_{simple}(\mathbf{x}_i)$ from the output values y_i:

$$y'_i = y_i - f_{simple}(\mathbf{x}_i) \quad \forall \mathbf{x}_i \in \mathcal{D}$$

   This results in a modified training set $\mathcal{D}' = \{(\mathbf{x}_i, y'_i)|\}$.

4. Determine the regression function $f_{svm}(\mathbf{x})$ for the modified training set by a support vector machine with Gaussian kernels. The usage of

Gaussian kernels ensures that this regression function differs from zero only in the proximity of training points. Beyond this proximity the regression function is zero (see fig. 5.10).

Additionally, a data-dependent ϵ is applied to weight pseudo-data points by a larger amount than other points. This ensures the regression function to address more importance to prior knowledge.

5. Combine both regression functions to the overall approximation

$$f(\mathbf{x}) = f_{simple}(\mathbf{x}) + f_{svm}(\mathbf{x}).$$

Advantages of this 2-step approximation are the simple incorporation of prior knowledge and the prevention of unpredictable global behaviour of the regression function. First, a simple global approximation is derived by incorporating prior knowledge. In this case, monotonic functions were used to describe the general behaviour according to the physics of longitudinal rolling. Within other applications, the derivation of prior knowledge may be totally different, but can be incorporated in the same way as long as a simple representation exists.

The application of a support vector machine with Gaussian kernels to the residuals of the first approximation ensures that only in regions with high training data density the global approximation is affected. In all other regions no modifications are made to the simple regression function.

5.4 Implementation

Feature extraction and the training procedures were implemented in Matlab, as well as the validation algorithms. The simple approximation f_{simple} was determined by the common least-squares approach with help of QR decomposition, while the SoftDoubleMinOver algorithm with kernel caching was used to determine f_{svm}. The obtained coefficients, support vectors, weights and parameters were exported to a plain text file. The steering of the rolling mill is implemented in ibaLogic, a graphical programming tool for signal processing and automisation. The single components have to be available as dynamically linked libraries for Microsoft Windows. Therefore, a C++

implementation for evaluation of the regression function was programmed, which also contains parsing routines to import the obtained support vectors and additional parameters. This library was then integrated into the pass schedule calculator framework, together with the implementation of the rolling strategy and various signal processing components to process the sensory inputs.

All components were tested on a simulation of the hot rolling mill and finally installed at Buderus.

Chapter 6

Results

A series of novel concepts has been introduced in the preceding chapters to use the support vector machine for prediction of rolling mill parameters. The 2-level approach and pseudo-data sets were proposed to incorporate prior knowledge to the MinOver algorithm for regression. The algorithm was further extended by kernel caching and an optimised evaluation of the regression function so that it scales linearly with the number of training patterns and learning steps.

Now, the optimal training parameters are selected by grid search and the quality of estimation is evaluated numerically. A runtime analysis and first experiences with the implementation of the pass schedule calculator at Buderus will conclude the chapter.

6.1 Parameter Selection

The support vector machine minimises the error according to the ϵ-insensitive loss function for each particular set of parameters ϵ, C and σ.

A quality criterion is needed to select the optimal parameter set among all parameter sets within a specific range. Frequently used quality measures are:

Mean Absolute Percentage Error (MAPE)

$$\frac{1}{P}\sum_{i=1}^{P}\left|\frac{y_i - f(\mathbf{x}_i)}{y_i}\right| \cdot 100\%$$

The MAPE is an intuitive measure to derive the mean percentual deviation of the approximation from the observation.

Root Mean Square Error (RMSE)

$$\sqrt{\frac{1}{P}\sum_{i=1}^{P}\left(y_i - f(\mathbf{x}_i)\right)^2}$$

In contrast to the MAPE, the deviation is squared, so that large deviations are much more important than small ones.

Root Mean Square Percentage (RMSP)

$$\sqrt{\frac{1}{P}\sum_{i=1}^{P}\frac{\left(y_i - f(\mathbf{x}_i)\right)^2}{y_i^2}} \cdot 100\%$$

The RMSP is the normalised version of the RMSE.

The following analysis is based on process data recorded during one month at the blooming train of Buderus. About 6300 workpieces were rolled within this period. Unfortunately, a large percentage of the data cannot be used for training, as necessary information such as high-temperature strength is not available for each single workpiece. Broken sensors as well as the mentioned plausibility checks further reduce the number of training data points so that only data from 4000 passes remained for training.

Now, the optimal parameter sets — according to the quality criteria above — were estimated for each groove and output variable by 10-fold cross validation on a grid with 1000 nodes ($C = 10^{-4}, 10^{-3}, \ldots, 10^{5}$, $\epsilon = 2^{-5.5}, 2^{-5}, \ldots, 2^{-1}$, $\sigma = 2^{-7}, 2^{-6}, \ldots, 2^{2}$). Obviously, there exist local minima (see fig. 6.1) and the grid node with least error is not necessarily the global minimum on the continuous scale. So, further refinement of the grid could yield better parameter sets. As the error varies by a large amount, an automatic parameter selection is absolutely necessary.

Evaluating all grid searches for all output variables (see figs. 6.2, 6.3, and 6.4) demonstrates that the usage of the support vector machine significantly reduces the validation error — no matter which type of error model is used. When predicting thickness reduction, the validation error is reduced by up to

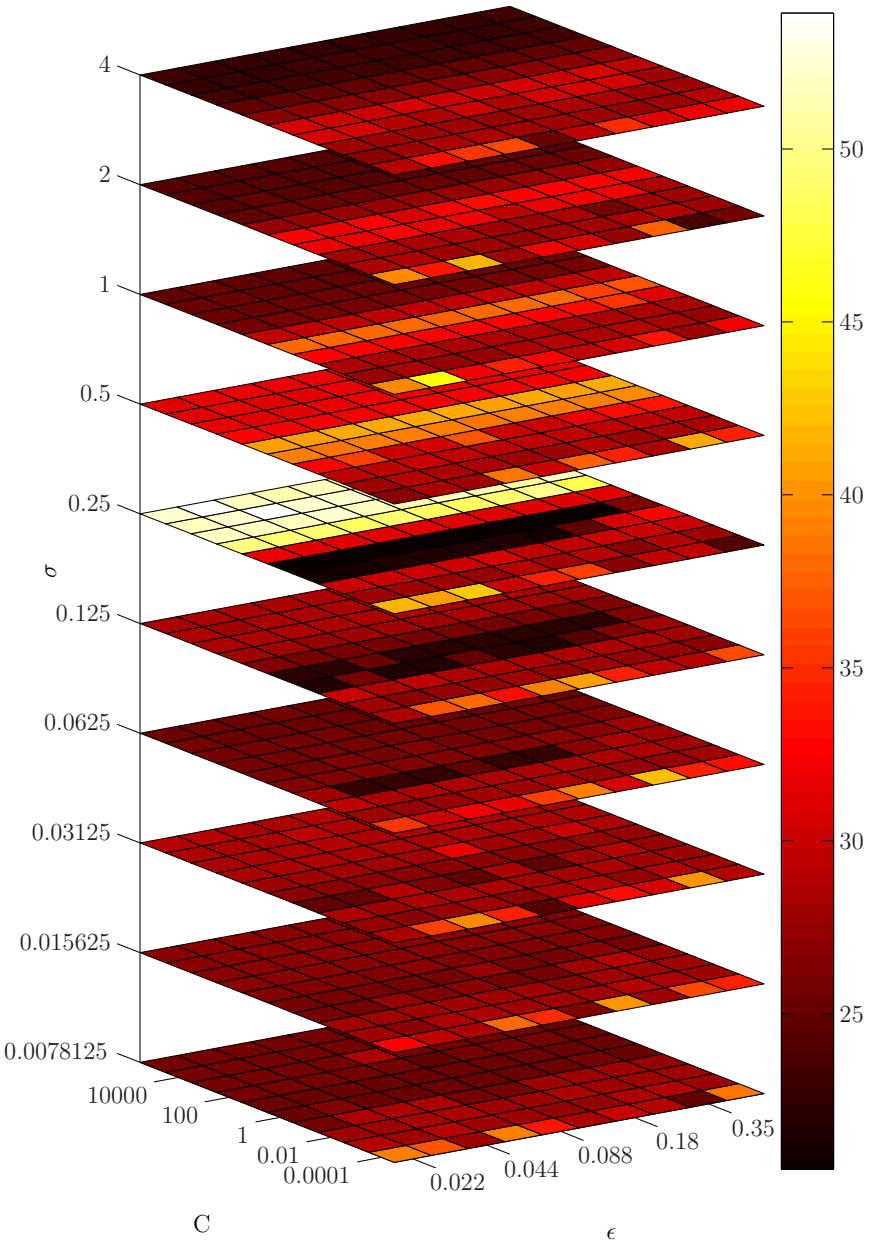


Figure 6.1: Example of grid search. The evaluation of the MAPE (torque estimator, groove 1) for each parameter combination by 10-fold cross validation shows that local minima exist. Numerically, the smallest error is achieved when using $C = 1$, $\epsilon = 0.044$ and $\sigma = 0.25$, but many other parameter sets yield nearly the same error (dark red areas). On the whole grid the error varies between 20 and 55 %.

Groove	MAPE		RMSE		RMSP	
	Simple	SVM	Simple	SVM	Simple	SVM
1	25.78	20.24	112.20	93.24	43.53	30.81
2	16.01	12.44	137.69	99.98	19.76	14.98
3	12.74	9.90	57.55	40.15	15.64	12.22
4	19.86	12.64	73.85	40.38	24.97	15.64
5	19.10	14.58	94.64	59.55	23.99	17.61

(a) Rolling torque prediction

Groove	MAPE		RMSE		RMSP	
	Simple	SVM	Simple	SVM	Simple	SVM
1	28.73	19.07	755.35	588.39	51.19	29.94
2	16.24	12.52	650.09	462.67	19.74	14.76
3	12.42	9.74	312.37	220.61	15.42	11.62
4	18.39	11.65	418.17	224.06	24.06	14.15
5	19.68	15.35	661.53	416.65	24.79	18.86

(b) Rolling force prediction

Figure 6.2: Results of 10-fold cross validation on a 10-by-10-by-10 grid. The RMSE should be interpreted in relation to the particular ranges (torque: 0, ..., 1400 kNm, force: 0, ..., 9000 kN).

75% and in most cases the resulting errors are between 10 and 20%. As the accuracy of the current measuring techniques for rolling torque and rolling force is about 15%, this is nearly an optimal result.

The remaining errors may be explained by numerous inaccuracies during process data acquisition and by unavailable input variables. The temperature of the workpiece is not measured and also the indirect temperature prediction technique — with help of radiation time, furnace residence time, and furnace temperature — fails due to the facts presented in 5.2.1. An irregular temperature distribution in the workpiece mostly affects the first passes and therefore the prediction capabilities in the first groove. Thus, the prediction errors for rolling torque and force are largest in the first groove.

Groove	MAPE		RMSE		RMSP	
	Simple	SVM	Simple	SVM	Simple	SVM
1	18.13	10.43	8.26	6.62	24.08	13.40
2	15.48	6.22	15.20	6.62	18.65	9.05
3	12.92	5.43	9.85	4.06	15.75	6.60
4	21.56	17.49	13.75	10.40	29.76	23.12
5	18.52	9.14	9.35	4.13	22.53	10.27

(a) Prediction of thickness reduction by rolling torque

Groove	MAPE		RMSE		RMSP	
	Simple	SVM	Simple	SVM	Simple	SVM
1	35.16	15.98	16.72	9.37	43.26	22.26
2	32.01	9.60	30.71	9.44	37.41	12.70
3	26.76	6.87	19.83	5.10	31.80	8.66
4	42.96	21.58	26.21	13.72	60.88	30.55
5	36.79	11.95	17.90	6.02	43.87	15.37

(b) Prediction of thickness reduction by rolling force

Figure 6.3: Results of 10-fold cross validation on a 10-by-10-by-10 grid. The RMSE should again be interpreted in relation to the range of the thickness reduction (0, . . . , 85 mm).

Additionally, the much too simple spread approximation influences the prediction errors indirectly. Starting with an initial entry width, the exit width is determined by the spread equation of Geuze with an error of w'. So, if the workpiece is tilted afterwards, the entry thickness of the next pass is modified by w' which leads to a modified assumed thickness reduction. Thus, within each pass not only the rolling force and torque values are subject to errors but also the thickness reduction.

Since different quality criteria were used, the best parameter set may not be the same for each criterion — one has to be selected. In the scope of this specific application the best parameters according to the MAPE

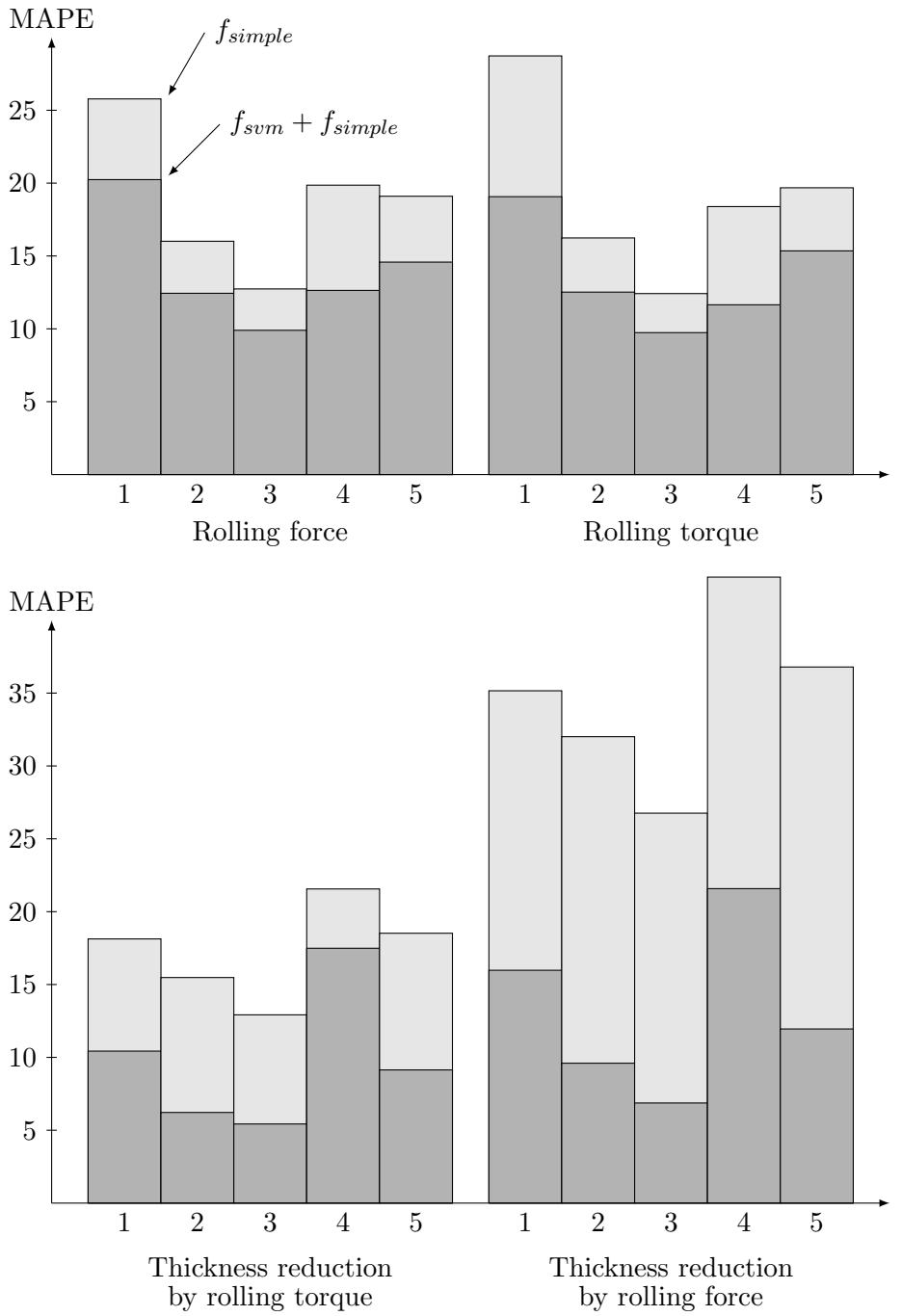


Figure 6.4: Performance improvements with the 2-layer approach. The 2-level approximation results in a MAPE between 10 and 20, whereas the simple approximation without local adaptations by the support vector machine always produces much larger error rates.

are used, which is motivated by the different weighting of deviations by the three criteria. The estimators tend to have larger relative errors for small output values than for large output values. So, those quality criteria that use the squared deviation, such as the RMSE and RMSP, attach too much importance to low output values. However, the correct prediction for large values is much more important, as wrong estimations will dramatically increase the wear of the rolls. Despite of this, the optimal parameter sets of the three quality criteria do not differ very much, i.e. the corresponding grid nodes have short distance.

6.2 Runtime Analysis

The parameter selection for the learning machine is the most crucial step with respect to time efficiency in the whole machine learning task (see fig. 6.5). Format conversion and decompression as well as feature extraction and selection have to be done but once for each data file, while the training and parameter selection has to be performed at regular intervals, so that the learning machine will recognize a changed behaviour of the rolling mill.

Step	Runtime
Format conversions and decompression from proprietary file formats to comma-separated values with process data taken from one month	$\approx$ 8 h
Feature extraction with Matlab functions	$\approx$ 10 min
Feature selection and plausibility checks	$\approx$ 1 min
Training with about 4000 data points and 10000 learning steps for a fixed set of parameters ϵ, C and σ	$\approx$ 1 min
Grid search with 10-fold cross validation for 1000 nodes to estimate the optimal parameter set	$\approx$ 65 h
Prediction of 400000 values, implemented in C++	$\approx$ 10 s

Figure 6.5: Runtime analysis. All calculations were made on an Intel Pentium 4 with 3 GHz and 1 GB of working memory.

One single training procedure of all four learning machines will only take about one minute on a state-of-the-art computer with 3 GHz and 1 GB of working memory, but using the proposed grid search and 10-fold cross validation will last for nearly three days. In order to retrain the learning machine periodically, there are various ways to improve the runtime.

1. The implementation of the Matlab routines in C will probably improve runtime by a factor between 10 and 20, so that the above mentioned grid search algorithm may take less than 5 hours. Thus, it would be possible to retrain the learning machine every day with data from at least one month.

2. The next step would be the implementation of an online learning algorithm. It would no longer be necessary to retrain on the complete data set, but only on the small set of all novel data points since that last training. Such an online algorithm has the advantages of much shorter runtime but training and validation will become more difficult. Especially dememorisation should be done with care, in order not to forget essential data points.

6.3 Experience with the Novel Pass Schedule Calculator

The novel pass schedule calculator has not been in use long enough to quantify the improvements derived by the introduced architecture and the use of machine learning theory, but a number of observations concerning the new system can already be mentioned here.

The design of new pass schedules for unconventional workpiece geometry or new materials has been simplified by orders of magnitude. A pass schedule is now generated automatically for each geometry and material and no longer requires user interaction. This dramatically reduces development time.

The number of sensor defects as well as the mean time to repair is much higher than was thought at the beginning of the project. Presently, the

learning algorithms cannot learn anything from such fragmentary data vectors and the preprocessing step will omit all those data points.

The prediction errors of rolling torque, force, and thickness reduction were mostly within acceptable bounds. Although the estimation is only based on a few input variables and strongly affected by noise, the generalisation capabilities are sufficient to be used in practice.

It became apparent that some of the predefined constraints, such as roll bite angle or the upper bound of the thickness reduction, were too subjective. The *feeling* what might be a good pass schedule differed widely between the blooming mill operators. So, the number of manual changes to the pass schedule during the rolling process increased because some operators did not believe that the calculated values can be used in practice. They had been working for decades with the former hand-crafted pass schedules and a lot of convicing has to be done to introduce the novel schedules.

A thorough comparison of the novel pass scheduling system to similar systems is quite difficult to achieve because no comparable systems exist. Neural networks have been used in steel manufacturing for a long time, but not at rolling mills with as few sensory inputs as at the blooming train. So at the moment, automised pass schedule calculation with help of predefined rolling strategies and regression estimation techniques to predict rolling force, torque, and thickness reduction can be regarded as unrivalled.

Chapter 7

Discussion

The numerical results of the preceding chapter showed that even with few and propably unconfident input parameters regression estimation with the support vector machine is effective. In this case, effectiveness means that errors close to the expected accuracy of the sensory inputs are achieved and smaller errors compared to previous techniques. Although the process data acquistion has to be improved, the results already show that the proposed concepts work excellent in practice. Besides the simplification of the pass schedule calculation also completely new application areas arise, such as online recalculation.

Now a critical review of the applied techniques will follow and some novel ideas on further work will be mentioned.

7.1 Incorporation of Prior Knowledge to MinOver

A major issue when dealing with regression estimation is an adequate and simple incorporation of prior knowledge. In the scope of predicting values at the hot rolling mill two different approaches were used. Either prior knowledge is expressed by an explicit mapping or by single data points.

The explicit mapping takes a certain number of free parameters to be determined by least-squares approximation. This concept is flexible, since no restrictions to the mapping are made — not even all input variables have to be used. The second step is to train a support vector machine

with Gaussian kernels on the residuals of the first approximation. Thus, the prior knowledge preserves the global tendencies, whereas the support vector machine learns the local non-linearities.

Prior knowledge in the form of pseudo-data points can be incorporated when no explicit mappings are known but only some high-confident data-points. Then, the training set is augmented by these points and a data-dependent ϵ-value is defined. By increasing the importance of pseudo-data points the shape of regression surface is locally dominated by prior knowledge points. This approach was implemented as an intuitive extension to MinOver.

Once again, MinOver appeared to be extendable to more and more learning tasks in an intuitive way, that does not require complicated optimisation techniques. In practice, the core learning algorithm for regression estimation takes no more than 100 lines of code within a higher programming language. Thus, especially in heavy industries where no dedicated machine learning groups exist — such as steel manufacturing, the MinOver algorithm with its different mutations is a user-friendly tool to establish artificial intelligence.

Further work should be done to reformulate the support vector machine, so that arbitrary prior knowledge can be incorporated directly — by equalities as well as by inequalites, information about monotonicity, local and global extreme values or other mathematical properties.

7.2 Validation and Parameter Selection Methods

Unfortunately, the presented support-vector machine for regression takes three parameters that have to be estimated by very time-consuming validation methods. No matter which type of parameter selection method is used — grid search, pattern search, or combinations of both — they require the training of hundreds to thousands of learning machines with the aim to select but one of these machines. A more efficient way would be to include those quality criteria that define the *best* learning machine already in the entire training procedure. The drawback of this approach is the necessity to reformulate the primal and dual representations of the support-vector ma-

chine for each novel quality criterion. This also demands modifications to a training algorithm such as MinOver. Surely, it would be far too complicated to design the algorithm each time by hand.

A novel approach would be to define a learning machine description language. The learning task, quality criteria and prior knowledge could be

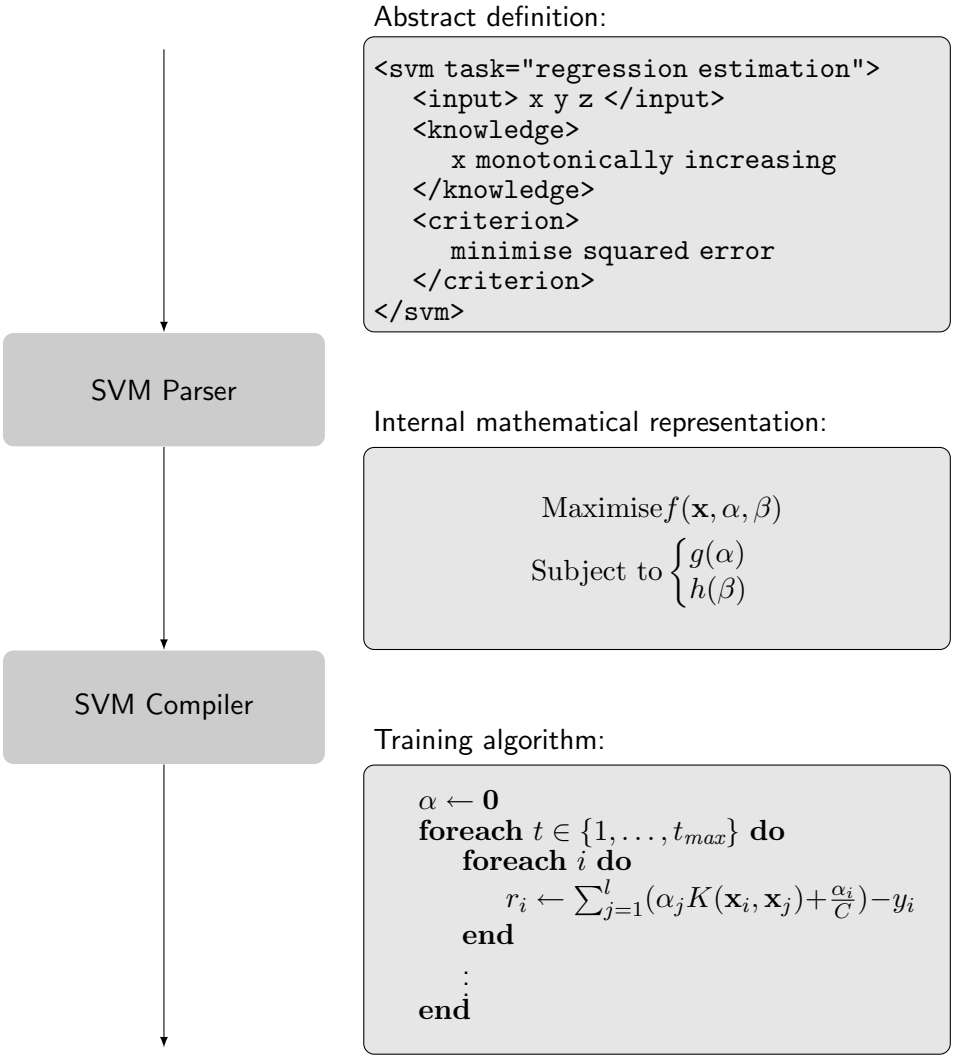


Figure 7.1: Concept of a learning machine compiler. After the abstract definition of a learning machine is transformed into the mathematical representation, the training algorithm is generated by the SVM-Compiler.

stated as abstract definitions. These definitions are parsed and the mathematical representation of the learning task is formulated as an optimisation problem. Then, the SVM-Compiler generates the learning algorithm — perhaps a variant of MinOver — in pseudo-code or any other programming language. Thus, for each learning task an individually optimised training algorithm could automatically be designed (see fig. 7.1).

With such a compiler the user would only have to state the type of machine learning task and his prior knowledge about the problem. He would not need to define *how* the machine should learn but *what* it should learn.

The common parameter selection methods are not only far too time-consuming, but they are also far away from being robust. Pattern search will only find local minima, while grid search not necessarily finds any locally

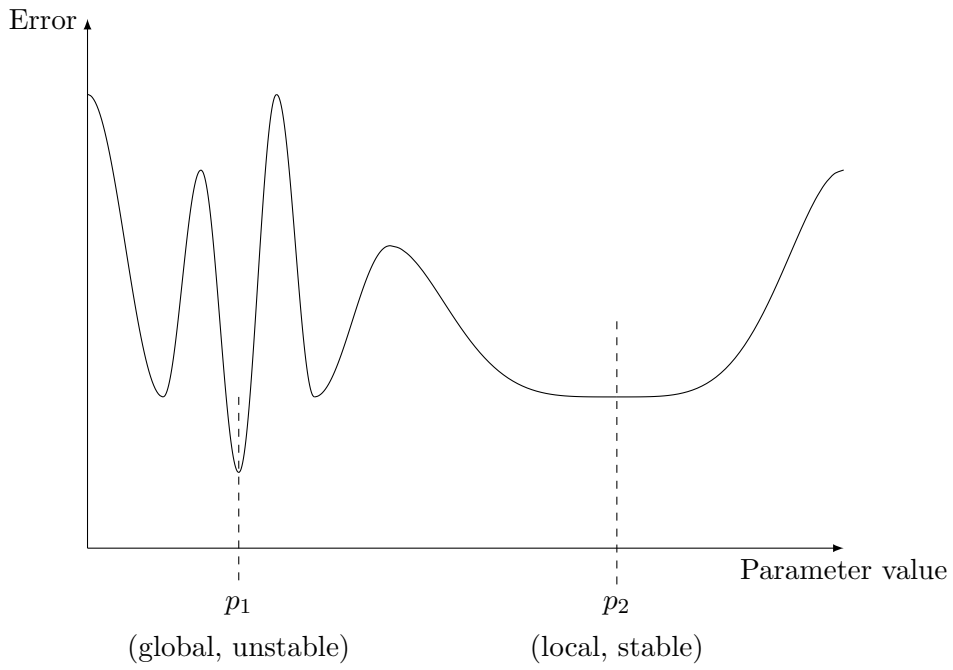


Figure 7.2: Stability of parameter selection methods. Allthough p_2 does not determine the global minimal point, it may be better to select p_2 instead of p_1, since the variance in the neighbourhood of p_2 is much lower than at the global minimal point p_1.

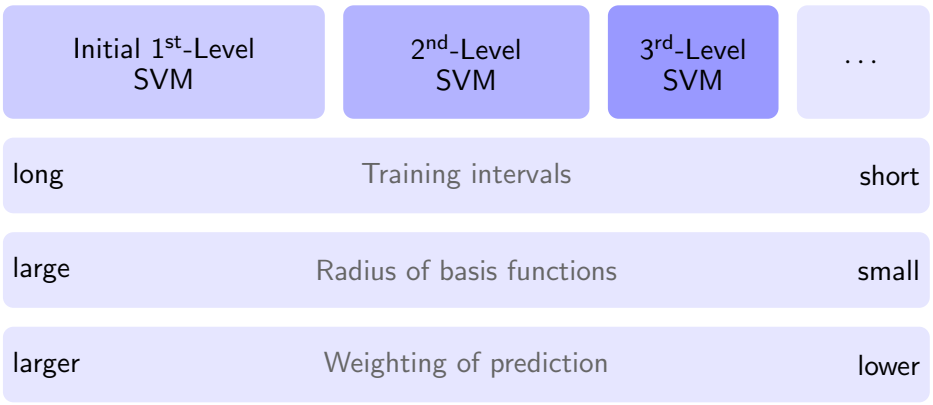
optimal parameter set, as it works on a discrete grid. Both methods select the parameter set that best fulfils the predefined quality criteria within the search space. Thereby, no attention is paid to the variance of the quality measure.

If the optimal set is located within an area of high variance, slight modifications of the training set may corrupt the generalisation capabilities significantly. A robust learning algorithm should rather select the parameter set within a region of low variance (see fig. 7.2). Thus, the parameter selection methods should be enhanced to find estimators that not only have high quality but also low quality variance in their neighbourhood.

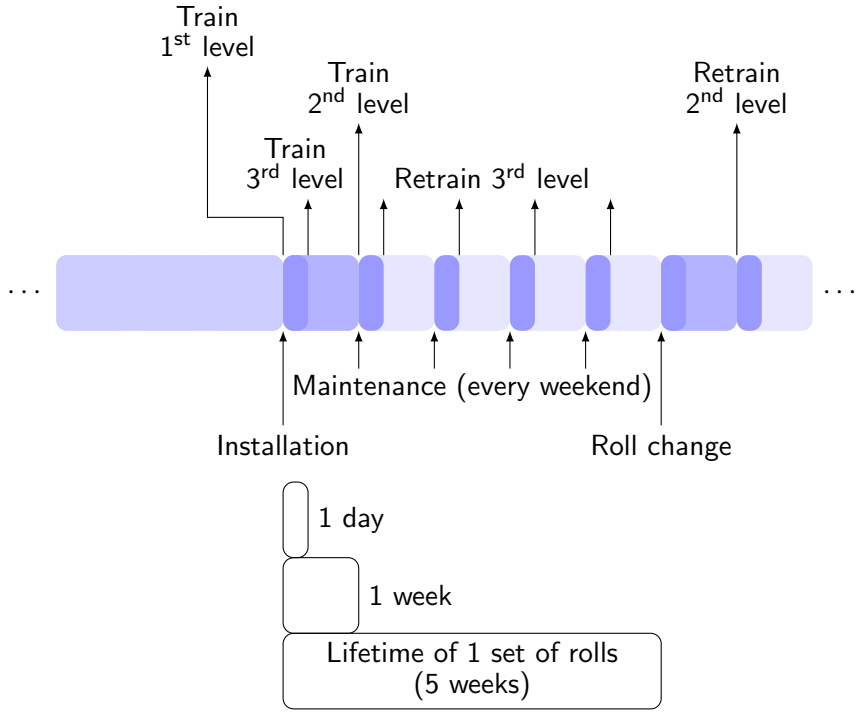
7.3 Online Learning

All the discussed learning algorithms are offline algorithms that have to be retrained in certain intervals to ensure proper adaptation. This requires the repetition of the whole learning algorithm — the parameter selection could possibly be abbreviated, since large changes should normally not occur. Nevertheless, an online algorithm that retrains the support vector machine as soon as new data is available would yield better prediction capabilities and have shorter *reaction time*. Online algorithms should have the ability to dememorise information, but they should not demomorise *essential* information. But this is a common problem when dealing with online adaption — the learning machine may adapt to the data from the near past, but forget everything about the distant past.

Again, approximation with different layers may be a solution. Starting with an initial learning machine, trained with a handpicked high-confident training set, *offset learning machines* may be trained in certain intervals on the error derived by the first learning machine (see fig. 7.3). Each level tries to reduce the error of the preceeding levels. To preserve former knowledge the weighting of higher levels and the radius of the basis functions is reduced. Thus, higher levels act more locally, while lower levels have global influences. To decrease reaction time, the training intervals of the higher levels are shortened (see fig. 7.3(a)).



(a) Layers of support vector machines



(b) Example: Training of rolling mill predictors

Figure 7.3: An online-learning concept. A hierarchy of support vector machines is defined, each with a different training interval, basis function, and weighting. In practice, the training interval should be linked to the different maintenance intervals. So, the first day after the weekly maintenance is used to collect data for training the third level. Data of the whole first week after a roll change is used for training the second level. And the first level is trained with a representative data set before the installation.

This concept can be further motivated by periodical maintenance intervals in industrial applications. Whenever parts or the whole machine — in this case the rolls of the rolling mill — are replaced, a new representative data set is collected. When only small changes are made, the training can be performed shortly after. But when the complete set of rolls is changed, longer periods are needed to gain representative data sets (see fig. 7.3(b)).

7.4 Rolling Strategy

So far, the machine learning approach was only used for predicting values of certain rolling mill sensors. The rolling strategy, i.e. the groove series, intermediate slab geometries, and tilting operations, is still selected statically.

A next step should be the appliance of intelligent search algorithms to find an optimal rolling strategy. Therefore, more accurate definitions of what is meant to be optimal are required. Some constraints, such as proper aspect ratios, have already be defined. Others, such as shape constraints and roll bite angles, need to be formalised more accurately.

The different slab geometries can be interpreted as nodes of a complex network. Each pass that changes the geometry is represented by an edge between two nodes. Now, exhaustive search algorithms such as Depth First or Breadth First Search could be used to find the *best* rolling strategy that converts the entry dimensions to the goal dimensions. One strategy may be better than others if it requires less groove changes, less tilting operations, or if it fulfils any other desired quality criterion.

The various geometry constraints limit the number of possible paths through the network so that the above mentioned search algorithms may be feasible in practice. Nevertheless, it would be much more efficient to use more *intelligent* search algorithms such as *genetic algorithms* or *simulated annealing*.

7.5 Outlook

The cooperation with steel producing industry and automisation system manufacturers showed that machine learning concepts cannot only optimise production techniques but also simplify scheduling and planning of tasks. To further improve the prediction of rolling mill parameters, the installation, calibration, and continuous maintenance of new sensorical inputs is essential. The prediction capabilities should also be enhanced by online-learning approaches — of special interest should be an online-variant of the MinOver algorithm. Another major issue is the speedup of the core learning algorithm not only to accelerate a single training procedure but also to enable fine-grained grid or pattern search techniques for parameter selection.

Further optimisation of the rolling process at the blooming train would be achieved by intelligent search techniques to derive optimal dynamic rolling strategies.

Finally, there are numerous issues to deal with when making machine learning *user-friendly*. Graphical user interfaces are needed that encapsulate and hide the learning machine and its various parameters from the end user, but allow him to select intuitively what has to be learned. A general learning machine description language would enable a wider range of programmers to incorporate artificial intelligence into their applications.

Glossary

billet	Knüppel, Block
biting condition	Greifbedingung
bloom	Bramme, Block
blooming train	Blockstraße
bounded spread	beschränkte Breitung
cogging train	Vorstraße
cold strip	Kaltband
deformation resistance	Formänderungswiderstand
drop forging	Gesenkschmiedestück
drop forging press	Gesenkschmiede
electric arc furnace	Elektrolichtbogenofen
empty pass	Leerstich
finishing train	Fertigstraße
free spread	freie Breitung
furnace residence time	Ofenverweildauer
furnace temperature	Ofenraumtemperatur
groove	Kaliber
groove depth	Kalibertiefe
groove width	Kaliberbreite
high-temperature strength	Warmfestigkeit
hot rolling mill	Warmwalzstraße

hot strip	Warmband
ingot	Gussblock
ladle furnace	Pfannenofen
length of contact arc	gedrückte Länge
lever-arm coefficient	Hebelarmbeiwert
pass	Stich
pass schedule	Stichplan
pass schedule calculator	Stichplanrechner
projected contact area	gedrückte Fläche
radiation time	Abstrahlzeit
roll bite angle	Greifwinkel
roll clearance	Anstellung, Walzensprung
roll gap	Walzspalt
rolling force	Walzkraft
rolling speed	Walzgeschwindigkeit
rolling strategy	Walzstrategie
rolling torque	Walzmoment
semi-finished steel	Halbzeug
side guide	Verschieber
slab	Bramme, Block
spread	Breitung
stand	Gerüst
steel casting	Stahlguss
thickness reduction	Dickenabnahme
tilt	kanten
tilting device	Kanter
tilting operation	Kantvorgang

Bibliography

[1] Noga Alon, Shai Ben-David, Nicolò Cesa-Bianchi, and David Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. *J. ACM*, 44(4):615–631, 1997.

[2] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[3] Tom Downs, Kevin E. Gates, and Annette Masters. Exact simplification of support vector solutions. *Journal of Machine Learning Research*, 2:293–297, 2001.

[4] Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. A unified framework for regularization networks and support vector machines. Technical report, Cambridge, MA, USA, 1999.

[5] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[6] Stefan Kern. Überarbeitung und Optimierung einer Blockstraßenkalibrierung. Master's thesis, Institut für Bildsame Formgebung, Rheinisch-Westfälische Technische Hochschule Aachen, 1997.

[7] Werner Krauth and Marc Mézard. Learning algorithms with optimal stability in neural networks. *J. Phys. A: Math. Gen.*, 20:745–752, 1987.

[8] Harold W. Kuhn and Albert W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492. Berkeley, California, 1951.

[9] Kai Labusch. MaxMinOver: Ein neues iteratives Verfahren zur Supportvektor-Klassifikation mit Anwendungen in der Gesichtserkennung. Master's thesis, University of Lübeck, 2004.

[10] Quoc V. Le, Alex J. Smola, and Thomas Gärtner. Simpler knowledge-based support vector machines. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 521–528, New York, NY, USA, 2006. ACM Press.

[11] Thomas Martinetz. MaxMinOver: A simple incremental learning procedure for support vector classification. In *IEEE Proceedings of the International Joint Conference on Neural Networks (IJCNN 2004)*, pages 2065–2070, Budapest, Hungary, 2004.

[12] Thomas Martinetz. MinOver revisited for incremental support-vector-classification. In C.E. Rasmussen, H.H. Bülthoff, M. Giese, and B. Schölkopf, editors, *DAGM 2004*, volume 3175 of *LNCS*, pages 187–194. Springer-Verlag Berlin Heidelberg, 2004.

[13] Thomas Martinetz, Otto Gramckow, Peter Protzel, and Günter Sörgel. Neuronale Netze zur Steuerung von Walzstraßen. *atp - Automatisierungstechnische Praxis*, 10/96, 1996.

[14] Edgar Osuna, Robert Freund, and Federico Girosi. Improved training algorithm for support vector machines, 1997.

[15] John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research (MSR), April 1998.

[16] John Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

[17] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[18] Martin Schlang, Einar Broese, Björn Feldkeller, Otto Gramckow, Michael Jansen, Thomas Poppe, Clemens Schäffner, and Günter Sörgel. Neural networks for process control in steel manufacturing. In *ICASSP '97: Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97) -Volume 1*, page 155, Washington, DC, USA, 1997. IEEE Computer Society.

[19] Daniel Schneegaß. Der DoubleMaxMinOver Approach zur Bestimmung der Support Vektoren bei Klassifikation und Regression mit Anwendungen in der Prozessindustrie. Master's thesis, Universität zu Lübeck, 2005.

[20] Daniel Schneegaß, Kai Labusch, and Thomas Martinetz. MaxMinOver Regression: A Simple Incremental Approach for Support Vector Function Approximation. In *Artificial Neural Networks - ICANN 2006*, pages 150–58, Berlin/Heidelberg, 2006. Springer.

[21] Carl Staelin. Parameter selection for support vector machines. Technical Report HPL-2002-354R1, Hewlett Packard Laboratories, November 19 2003.

[22] Vladimir N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York, 1982.

[23] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory (Information Science and Statistics)*. Springer, November 1999.

[24] Vladimir N. Vapnik and Alexey Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[25] Vladimir N. Vapnik and Alexey Y. Chervonenkis. Necessary and sufficient conditions for the uniform convergence of means to their expectations. *Theory of Probability and its Applications*, 26(3):532–553, 1982.

[26] Xiaoyun Wu and Rohini Srihari. Incorporating prior knowledge with weighted margin support vector machines. In *KDD'04*, pages 326–333, Seattle, Washington, August 2004.