Monitoring symptom progression in Parkinson's Disease using Least Squares Support Vector Regression

K. den Hertog

Technische Universiteit Delft

TU Delft

Monitoring symptom progression in Parkinson's Disease using Least Squares Support Vector Regression

MSc Thesis

by

K. den Hertog

to obtain the degree of
Master of Science in Aerospace Engineering at the Delft University ofTechnology,
to be defended publicly on Tuesday August 30, 2022 at 09:30.

Student number:	4457803	
Supervisors:	Prof. dr. ir. M. (Max) Mulder	TU Delft
	Dr. ir. D.M. (Daan) Pool	TU Delft
	Dr. ir. J.J.M (Johan) Pel	Erasmus Medisch Centrum
External comittee member	Dr. A (Alexei) Sharpanskykh	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

This report contains my Master's Thesis, which is the final part of my 7 years as a student at Delft University of Technology. I chose to focus my research on a project only slightly related to Aerospace Engineering. In this project models to identify pilot control behaviour are applied to develop methods for aiding the diagnosis of neurological disorders. My work continues on the work performed by my predecessors, in particular Rick de Vries and Lieke Lugtenborg. It focuses on identifying behavioural changes in control performance, caused by Parkinson's Disease, through use of a machine learning model: Least Squares Support Vector Regression. My choice for this subject came both out of a very personal interest (my grandfather suffered from Parkinson's Disease), as well as a desire to learn about and work with machine learning algorithms.
Part I of this report contains the scientific paper that covers the main findings of my thesis work. It improves upon the work done by Lieke by increasing the accuracy and decreasing the latency of detecting behavioural changes due to Parkinson's Disease. Part II contains my preliminary work, covering the literature related to my thesis and motivation my selection of the Least Squares Support Vector Regression. Part III contains the appendices belonging to the scientific article.
Work on this thesis was not always easy and straightforward, and I could not have completed it without the valuable input from my supervisors. Daan and Johan, without your positivity, patience, suggestions, stimulation and kindness this thesis would not be what it is today. Thank you for everything!
Besides my thanks to my supervisors, I would like to thank my beloved friends and family for keeping me sane during the two Covid-stricken years I worked on this thesis. Martijn and Lina, thank you for the many walks we took, always providing a listening ear and sharing in my struggles and accomplishments. Papa en Mama, thanks for the support, editing, and the occasional "schop onder mijn kont". And finally I would like to thank you for taking the time to read this thesis. I hope you enjoy my work, and find inspiration in it!

K. den Hertog
Delft, August 2022

Contents

I

Scientific Article

to be graded for AE5310 Thesis Control and Operations

Monitoring symptom progression in Parkinson's Disease using Least Squares Support Vector Regression

K. den Hertog (MSc Student)
Supervisors: dr. ir. D.M. Pool*, dr. ir. J.J.M. Pel**, Prof. dr. ir. M. Mulder*
*Control & Simulation, Department Control and Operations, Faculty of Aerospace Engineering, Delft University of Technology, Delft, Netherlands
**Erasmus MC department of Neuroscience, Rotterdam, Netherlands

***Abstract*—Parkinson's Disease is a neurodegenerative disease that has a decline in motor behaviour as one of its main symptoms. This decline is currently monitored using subjective measures, such as questionnaires and clinical observations. More detailed and objective tracking of this decline can improve treatment of the disease and allow for earlier identification. Earlier work in this area has lead to the development of a proof-of-concept for detecting behavioural changes in motor performance, but the linear regression model used was limited in its accuracy and had a high latency. This paper uses a non-linear Least Squares Support Vector Regression (LS-SVR) model to increase performance in those areas. LS-SVR was chosen for its ability to regress complex non-linear relationships and because it is highly adaptable and scalable, yet easy to understand and work with. Test data were simulated based on earlier measured data which resulted in a complete and varied data set that allows for exploring many different situations. Results from the machine learning model when applied to the test data are promising. With the right combination of hyperparameter settings an improvement of 80% in accuracy was reached, and latency could be reduced by 30%. Additionally, a sensitivity analysis of the hyperparameters revealed further room for improvement with more careful tuning. Finally, it was shown that by using LS-SVR to make predictions on data from future trials a further reduction in latency can be achieved. Overall, the new model shows definite improvement over the earlier model and can be developed further in subsequent steps towards a clinical applicable method.**

***Index Terms*—Manual tracking, Parkinson's disease, cybernetic approach, motor performance, behavioural changes, machine learning, regression, least squares support vector machine**

NOMENCLATURE

α	Lagrange multiplier
β_0, β_1	Linear regression parameters (intercept and slope)
δ	Change in parameter value due to PD symptoms
ϵ	Regression error
γ	LS-SVR regularisation constant
μ	Parameter average
ω_{nms}	HC neuromuscular frequency
$\phi(\mathbf{x})$	Kernel function
σ	RBF kernel parameter
σ_p	Parameter standard deviation
τ	Time delay of HC dynamics
ζ_{nms}	HC neuromuscular damping ratio
D	Difference metric
D_{lim}	Difference threshold value
$\bar{D}$	Averaged difference metric
H_{ce}	Controlled element dynamics
H_{pe}	HC dynamics
HC	Human controller
K	Radial Basis Function kernel
K_c	Controlled element gain
K_p	HC gain
LS-SVR	Least Squares Support Vector Regression
N	Sample size
N_h	Number of healthy trials
N_s	Number of symptomatic trials
PD	Parkinson's Disease
RMS_e	Tracking performance
RMS_u	Control performance
b	LS-SVR bias term
e	System error
$f(\mathbf{x})$	LS-SVR regression function
f_t	Target signal
l_{det}	Detection window
n	Nonlinear remnant
n_{future}	Number of predicted trials
u	Control signal
$\mathbf{w}$	LS-SVR coefficient vector
$\mathbf{x}$	Vector of trial numbers
x	Trial number
y	System output
y_p	Parameter value
y_{LSSVR}	Value of the LS-SVR regression
y_{avg}	Average of training data

I. INTRODUCTION

Around the world, more than 10 million people are living with Parkinson's Disease (PD) [1], a neurodegenerative disease that impacts almost everything in a patient's life, from simple tasks like drinking a cup of coffee, to more complex and abstract problems such as living a happy life [2]. One of the main ways PD affects a patient's life is through influencing motor performance. The disease causes cell death of dopaminergic neurons within the basal ganglia, leading to a shortage of the neurotransmitter dopamine [3]. Dopamine is vital for the

communication between brain areas that are responsible for coordination and effective planning of movements, and a lack of it causes the classical motor symptoms associated with PD: bradykinesia, postural instability, rigidity, and tremors [4]. Parkinson's Disease is incurable, and its symptoms are progressive. Luckily, symptom severity can be reduced by therapy and medication, but treatment is dependent on the disease stage [1, 5]. Currently, the main way of determining the disease stage and planning treatment is by use of subjective clinical rating scales: the patient's motor symptoms are evaluated by a physician, and an extensive questionnaire is filled in [6]. These results are mapped to a rating scale, such as the often used 5-point Movement Disorder Society's Unified Parkinson's Disease Rating Scale (MDS-UPDRS) [7]. Evaluating disease progression in such a way comes with a few disadvantages: it is subjective and time-intensive, and has therefore a low resolution [8].

Much research is being done into more objective methods of tracking PD progression [9, 10]. Brooks et al successfully mapped human behaviour coding of video recordings of PD patients to MDS-UPDRS, as a proof-of-concept for a method of quantifying the scale [11]. Patel et al ran a pilot for a method of estimating the severity of the PD motor symptoms, using wearable sensors, and it achieved low estimation errors (2-4%) [12]. Finally, Zhan et al used multiple input sources (voice, finger tapping, gait, balance, and reaction time), obtained by smartphone, in a machine learning model in an attempt to quantify PD severity [13]. This resulted in a novel metric correlating well with parts of the MDS-UPDRS. However, this research is all in the proof-of-concept stage and much work has still to be done to develop a clinically applicable method.

Another approach to objectively tracking PD progression is by evaluating a decline in fine-motor performance, as the dopamine-deficiency of PD patients leads to limitations in performance of eye-hand tasks [14]. This performance can be analysed by using eye-hand coordinated tracking tasks. Much research has been done in this area [15, 16, 17, 18, 19, 20], but, as with the other proof-of-concept methods, no clinically applicable method has been developed yet. A new collaborative research project has been set up by the Erasmus Medical Centre Department of Neuroscience and the Delft University of Technology Faculty of Aerospace Engineering to change this. In this project, methods are being developed to quantify and analyse the effects of neurodegenerative disorders [21]. The main tool of this project is cybernetics; using this to analyse and track the motor performance of a human controller within a dynamic system [22] in order to detect deviating behaviour. Earlier research related to PD involved creating a specific tracking task and using that to show that the loss of motor performance due to PD can be quantified [23]. Lugtenborg subsequently proved that a sudden change in tracking task control behaviour, caused by PD progression, can be detected using linear trend analysis methods [21, 24].

While Lugtenborg's work showed that behavioural changes are indeed present in PD progression, the linear regression method used still has its limitations [21]. Changing trends only showed for certain tracking parameters, and in a maximum of 50% of the cases. Furthermore, linear regression is slow in responding to changes (it has high latency), meaning a lot of data with changed behaviour is needed before a change is is actually detected. On average, it took 10 trials before a change in trend was detected. Finally, only historic data are used in the trend analysis. Predicting the control performance for future trials might lead to earlier notice of changing behaviour and could provide medical staff with valuable insights.

In this paper a non-linear regression model is investigated as a more advanced approach for the same data set that was previously used [21]. Some of the most advanced regression models fall in the domain of machine learning. Unlike traditional statistical methods, machine learning algorithms require no prior knowledge of the underlying distribution of the data, but are able to infer this themselves [25, 26]. Furthermore, most machine learning algorithms are adaptable to a variable and scaling data set, and can still be quick to run [27]. Many different regression algorithms can be used, but one of the simplest, yet still powerful, is Least Squares Support Vector Regression (LS-SVR) [28]. LS-SVR is a type of machine learning model, capable of regressing complex non-linear relationships and being highly adaptable and scalable [25, 29], and easy to understand and work with. General Support Vector Regression as well as LS-SVR specifically have seen application in various research areas [30, 31, 32, 33], including PD research [34, 35].

The goal of this paper is to use LS-SVR for detecting a change in control behaviour due to PD progression and improve upon the linear regression model used by Lugtenborg [21]. It is hypothesised that LS-SVR will have both a higher accuracy as well as less latency than linear regression, and the eventual aim is to provide earlier notice of a change in PD symptoms in a clinical setting. In order to test the potential improvement, data sets comparable to ones used in earlier research [23, 21] will be customly generated. Due to the COVID-19 pandemic getting measurements of real participants was undesired, and having a simulated data set allows for easy generation of large data sets, and testing of many different cases. A direct comparison is run between LS-SVR and the linear regression method used in earlier work [21]. To detect the changes in the simulated behavioural data a new detection metric is introduced. Two different versions of this metric are investigated, one only looking at historic data, and the second taking projected future data into account, with the aim of further reducing latency. Furthermore, the influence of some key hyperparameters present the method for detecting trend deviations will be investigated, to be able to set them to their optimal value. Insight into this allows for further optimalisation of the method in later stages of its development leading to clinical application.

This paper starts with giving the background information for the research in Section II; explaining the control task, the two regression models, and the method used for detecting behavioural trends. Section III describes the method for generating data sets, as well as the different steps taken in investigating the behavioural change detecting performance of the LS-SVR model. The results of these steps are laid out in Section IV, and Section V contains a discussion on the implications of this research and its results. Finally, the paper is concluded in Section VI.

II. Background

A. Cybernetic approach

The basis for the method used in this research is a tracking task. In such a task, the human controller (HC) is asked to control a dynamic system that is subject to disturbances. Analysis of control performance in this task gives insight into the cybernetic control behaviour of the HC [36]. For this research project the data are based upon a horizontal-axis pursuit tracking task, which was developed and used in preceding studies [21, 23], and is similar to tasks used in other PD-related studies [17]. In the tracking task the participant is asked to keep the tracking error to a minimum by manipulating the controlled element via a touch screen. In earlier work the participants were asked to perform the tracking task in a number of *trials*. In this way a longitudinal data set was obtained [21]. For this paper the longitudinal data set was simulated, this is further explained in Section III-A.

1) Tracking task: A schematic overview of the tracking task is given in Figure 1 [21], and its corresponding block diagram is shown in Figure 2 [37]. The *Controller* block indicates the HC, and H_{CE} is the controlled element. The latter's dynamics, which are based on previous research, are taken to be single integrator. In Equation (1) these dynamics are given, with the gain, K_c, taken to be equal to 1 [21].

$$H_{ce}(s) = \frac{K_c}{s} \tag{1}$$

The input to the tracking task is determined by a forcing function that takes the form of a quasi-random multisine signal [21, 23].

2) Cybernetic HC model: From experimental data of the tracking task a cybernetic model can be fitted that describes the detailed dynamics of the HC. The model used for this research is the extended crossover model, defined by McRuer and Hex (1967) as a combination of a linear HC model and a nonlinear remnant [38]. The linear part consists of the dynamics of the neuromuscular system, a pure delay term, and a pilot equalisation model, taken to be a pure gain term for a single-integrator control task. The contribution of the nonlinear remnant can separated from the linear HC dynamics by proper design of the forcing function [36]. The used HC model is given in Equation (2).

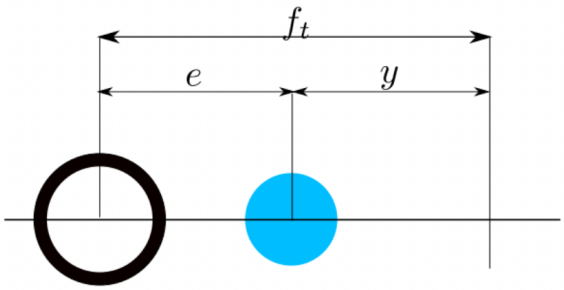


Fig. 1: Schematic overview of the tracking task performed by the participants [21]. f_t is the target signal, and the black ring is the target. The human controller controls the blue circle, the controlled element. y the system output, and e the error, $f_t - y$.

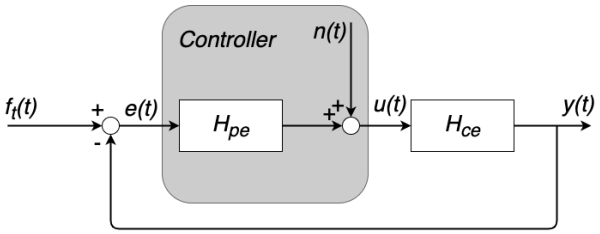


Fig. 2: Block diagram of the tracking task, adapted from El et al [37]

$$H_{pe}(s) = K_p e^{-s\tau} \underbrace{\frac{\omega_{nms}^2}{s^2 + 2\zeta_{nms}\omega_{nms}s + \omega_{nms}^2}}_{\text{Neuromuscular dynamics}} \tag{2}$$

The HC model contains four parameters. These are the controller gain (K_p) the time delay (τ) and the neuromuscular frequency and damping ratio (ω_{nms} and ζ_{nms}, respectively). These four parameters are used for tracking the HC's control dynamics across trials and participants, and allow for comparisons to be made.

In addition to the parameters of the HC model, two performance parameters will be included, with the aim of providing additional parameters on which the analysis can be based [21]. These are two well known metrics for tracking performance (RMS_e) and control activity (RMS_u). The first, RMS_e, is determined using Equation (3). It indicates whether or not the HC reduced the tracking error while controlling the system, in which case $\text{RMS}_e < 1$. For $\text{RMS}_e = 1$, the HC does not give any input, and for $\text{RMS}_e > 1$ the HC is increasing the tracking error with their input. RMS_u, given by Equation (4), is a measure of control activity; a high value indicates that the HC is using an active control strategy, while for a lower value of RMS_u this is more reserved.

$$\text{RMS}_e = \text{RMS}(e)/\text{RMS}(f_t) \tag{3}$$

$$\text{RMS}_u = \text{RMS}(u)/\text{RMS}(f_t) \tag{4}$$

3) Effect of Parkinson's Disease on the HC parameters: Earlier research investigated what happens to the nominal values of the HC parameters when the HC is experiencing early-stage PD symptoms [23]. It was shown that there is a slight increase in values for τ and ω_{nms}, a significant decrease for K_p and a significant increase for the value of ζ_{nms}. In Table I the mean and standard deviation for the HC parameters are given, based on the average for the participants in the study [21]. These values are considered typical for a healthy subject. Table II gives a range for how these values can change when PD symptoms are introduced, with δ indicating the change to the nominal mean [21]. Table II also shows the maximum δ as a percentage of the nominal standard deviation.

TABLE I: Nominal HC parameter values for healthy subjects [21]

	K_p [-]	τ $[s]$	ζ_{nms} [-]	ω_{nms} $[rad/s]$
μ	0.91	0.37	0.38	8.7
σ_p	0.26	0.14	0.19	4.3

TABLE II: Range for the change in HC parameter values caused by PD symptoms, δ [21, 23]

	Range for δ	Max δ as % of σ_p
K_p	$-0.9 < \delta < 0$ [-]	-346%
τ	$0 < \delta < 0.07$ $[s]$	50%
ζ_{nms}	$0 < \delta < 0.31$ [-]	163%
ω_{nms}	$0 < \delta < 5$ $[rad/s]$	116%

B. Regression modelling

In this paper two different regression models were compared. The linear regression model used by Lugtenborg is taken as a baseline [21], and compared against a new nonlinear regression model: Least Squares Support Vector Regression. This section explains the two models, as well as the implemented approach for detecting behavioural changes in the tracking task data.

B1. Linear Regression

1) Model: The linear regression model is defined by Equation (5). For each of the parameters of interest (K_p, τ, ζ_{nms}, ω_{nms}, RMS$_e$, RMS$_u$) the relation between the parameter value, y_p and the trial number, x, can be determined.

$$y_p = \beta_0 + \beta_1 x + \epsilon \tag{5}$$

2) Dectecting changes: In earlier work a student's t-test was used to determine if there is a significant change in de trend present in the data [21]. The null-hypothesis, no change ($\beta_1 = 0$), was tested against the alternate hypothesis of there being a change of the trend in the data ($\beta_1 \neq 0$) [21, 39]. By using Equation (6), which has a t-distribution with $N-2$ degrees of freedom if $\beta_1 = 0$ (N being the sample size), the significance of a potential change could be determined [39]. When the null-hypothesis is rejected with a significance of $p < 0.05$ a change is detected in the PD progression data.

$$t = \frac{\hat{\beta}_1}{\sqrt{\frac{\sum (y_{p_i} - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2}{N-2}}} \tag{6}$$

B2. LS-SVR

Least Squares Support Vector Regression (LS-SVR) is a reformulation of the standard support vector regression algorithm [28, 40, 41]. It uses linear programming and is therefore both easy to understand and easy to implement [42]. In its most basic form, LS-SVR is only capable of linear regression, but through the use of kernels non-linear regression can be performed [41]. These kernels are defined in literature, and different variants exist that have different properties [30].

1) Model: The basic idea of LS-SVR is to find the optimal hyperplane; that is, to fit a function, $f(\mathbf{x})$, through the training data that lie close to as many datapoints as possible. Where in ordinary least squares one fits this line by minimising the sum of the squared errors, in support vector regression the objective is to minimise the coefficients in the estimation function, handling the error in the constraints of an optimisation problem. The estimation function $f(\mathbf{x})$, for a given input $\mathbf{x}$, is given by Equation (7), where ϕ indicates the non-linear kernel used.

$$f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x}) + b \tag{7}$$

To find the coefficient vector $\mathbf{w}$, LS-SVR is formulated as an optimisation problem, as given in Equation (8). The objective function consists of two terms: the minimisation of the coefficients, $\mathbf{w}$, and the minimisation of the errors, ϵ, regulated by a constant, γ. The objective function is subject to one constraint, namely that the regression result has to equal the actual value y_{p_k} (corresponding to input $\mathbf{x}_k$), excluding the allowed error.

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \ \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + \frac{1}{2}\gamma \sum_{k=1}^{N} \epsilon_k^2$$
$$\text{subject to} \quad y_{p_k} = \mathbf{w} \cdot \phi(\mathbf{x}_k) + b + \epsilon_k, \quad k = 1, ..., N \tag{8}$$

Solving the optimisation problem leads to the estimator given in Equation (9), where α_k are the Langrange multipliers, and b is the bias [42, 43].

$$f(\mathbf{x}) = \sum_{k=1}^{N} \alpha_k \phi(\mathbf{x}) \cdot \phi(\mathbf{x}_k) + b \tag{9}$$

As stated before, ϕ indicates the non-linear kernel function. Different kernels can be used, but for this paper the Radial Basis Function kernel is chosen, as it is versatile and has adequate computing performance when applied to this particular data set [30]. It is given by Equation (10), with σ being the kernel parameter.

$$K(\mathbf{x}, \mathbf{x}_k) = e^{-\frac{||\mathbf{x} - \mathbf{x}_k||^2}{2\sigma^2}} \tag{10}$$

2) Detecting changes: In order to detect changes in control performance a detection metric for the LS-SVR model has been defined, along with a variation that takes predicted trials into account. This metric is called the "difference metric", and its basic version is determined by taking the difference between the LS-SVR regression and the average of the training

data for the last added trial, as given in Equation (11), where (D) is the difference metric, evaluated at the last added trial.

$$D = |y_{\text{LSSVR}} - y_{\text{avg}}| \tag{11}$$

An example of the difference metric is illustrated in Figure 3, where the average of the training data is shown, along with the LS-SVR regression. The difference for the current added trial, trial 52, is shown by the thick black line. The full time-plot of the difference metric is shown in blue: for each trial the difference is determined, and when no PD symptoms are present in the data the difference is expected to stay below the threshold. However, when PD symptoms are introduced, the LS-SVR will deviate from a horizontal line as the values of the parameters change with δ (Table II). The difference will increase significantly, and as a result a changing trend is detected if D is larger than some threshold value D_{lim}. To prevent false positive detection, analysis is actually done for stable changes: the difference should not go below the threshold again for a number of subsequent trials, (l_{det}). In Figure 3 the triangle indicates were the a change in the trend is detected. D_{lim} and l_{det} are also shown.

The variation on the difference metric utilises one of the inherent advantages of a machine learning algorithm, like LS-SVR: using the regression to predict control performance for future trials, leading to earlier detection of changes. Instead of only taking the difference at the last added trial, the LS-SVR model is used to predict the parameter value for n_{future} trials ahead, allowing for difference calculation at those trials as well. These differences are subsequently averaged, in order to arrive at the new metric, $\bar{D}$. In Figure 3 this is illustrated as well. The black shaded area indicates n_{future}, and it is in this area that D will be averaged. The new metric is given in Equation (12). The detection procedure remains unchanged, with a changing trend being detected if $\bar{D}$ is larger than the threshold D_{lim} for l_{det} subsequent trials.

$$\bar{D} = \frac{1}{(n_{future})} \sum_{k=0}^{n_{future}} |y_{\text{LSSVR}} - y_{\text{avg}}|_k \tag{12}$$

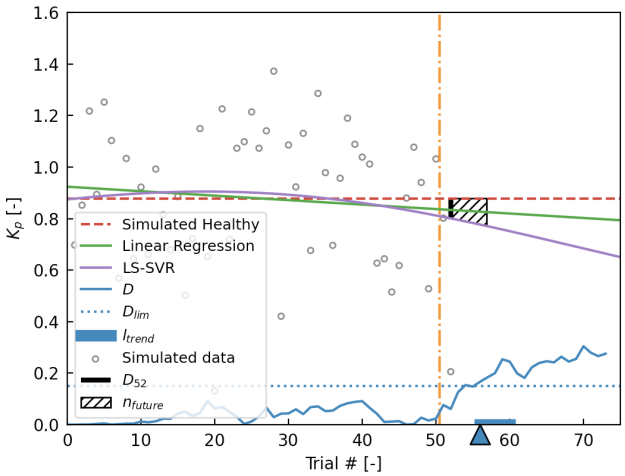


Fig. 3: The black line in the figure indicates the difference taken as the evaluation metric. The blue line is a plot for the difference, with the triangle indicating where a changing trend is detected in the data. The LS-SVR regression shown in this figure is from Trial 52.

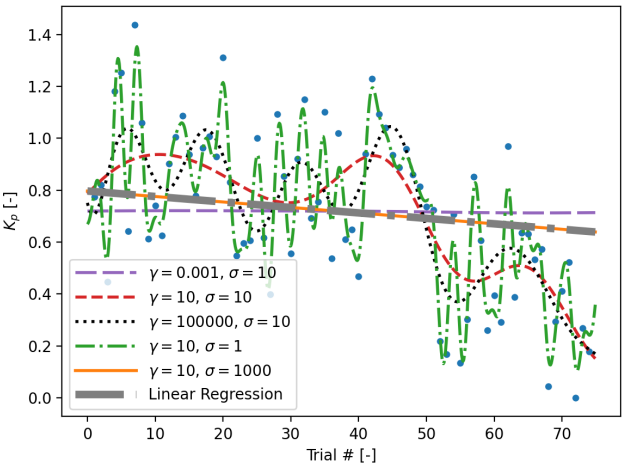


Fig. 4: Different values for the hyperparameters lead to different regression lines. As training data are the K_p data of one simulated participant used, with $N_h = 50$ and $N_s = 25$.

C. LS-SVR hyperparameters

The LS-SVR model used in this paper has two hyperparameters: γ and σ. γ is the regularisation constant of the LS-SVR objective function, Equation (II-B1). It determines the trade-off between the two parts of the objective function. A lower value leads to less emphasis being placed on minimising the errors, thus less overfitting. σ is the kernel specific hyperparameter, see Equation (10), which determines how "flexible" the regression is. A larger value means that the regression is flatter, but therefore a worse fit for the data, while a higher value allows the regression line to match the training data.

Examples of LS-SVR fits using different hyperparameter values are shown in Figure 4, along with the linear regression. Note that with sigma being large, the regression is forced to be as flat as possible, leading to the same result as linear regression. In the figure one of the main advantages of LS-SVR over linear regression can also be seen: with the right combination of hyperparameters LS-SVR is capable of higher order regression, and can therefore quickly change when the underlying trend in the data changes. This is in contrast with linear regression, where the previous trend has to be overcome with sufficient new data.

D. Online learning approach

In order to reflect how this approach would be used in a real world scenario, both models will be run in an online learning approach [44]. This means that, starting with a single trial, the machine learning model is trained and trend analysis is performed. These results are stored, and the second trial is

introduced into the training data set, and training and change detection are performed again. This continues until all training data are used. This is opposed to batch learning, where the entire training data set is inputted into the model, and training and trend analysis is performed only once. For the eventual application of this research to PD evaluation, online learning is the better approach, as it allows for continuous monitoring of (potential) patients.

III. Method

A. Data generation

The data set used in this research is a simulated data set, based on earlier experimental work [21]. By using a simulation-based approach, varying scenarios can be tested and the machine learning model is not constrained by incomplete or inconsistent data. An example of the simulated data set can be found in Figure 5. The data generation was done in three steps. First, for the HC model parameters (K_p, τ, ζ_{nms} and ω_{nms}), data for N_h healthy trials are independently drawn from a normal distribution, its parameters (μ, σ_p) equal to the average parameters of the participants in the earlier study [21], see Table I.

In addition to the N_h healthy trials, a decline in control performance due to PD is simulated for N_s symptomatic trials. For these trials, data are drawn from the same distribution as for the healthy trials, and subsequently appended with a δ value, taken from a U(a, b) distribution with parameters from Table II. Lastly, the two performance parameters (RMS$_e$ and RMS$_u$) are determined by simulating the tracking task shown in Figure 1. This is done using the block diagram in Figure 2, without including the remnant ($n(t)$), and determining the RMS of the relevant signals [21]. This is done for each set of HC model parameter values, resulting in a complete data set of 6 metrics for each trial that can be simulated for as many different trial runs and participants as desired.

B. Model implementation

The two regression models used in this study are implemented in Python, using existing libraries. For linear regression the *LinearRegression* model from the Scikit-learn library is used [45], and for the LS-SVR model the Scikit-learn libary is used to make a custom regressor class [46].

C. Method hyperparameters

The method described in this paper has a number of settings, or hyperparameters, that influence the behaviour of the trend detection, and the performance of the method. The hyper-parameters are divided into three groups. First there are the hyperparameters related to the data generation procedure:

- N_h: the number of healthy trials in the training data.
- N_s: the number of symptomatic trials in the training data.
- Simulation parameters: μ and σ_p for the normal distribution used in data simulation (Table I).
- δ: offset values for symptomatic trials (Table II).

Secondly, there are the hyperparameters related to the LS-SVR model:

- γ: hyperparameter the in LS-SVR cost function.
- σ: hyperparameter in the LS-SVR kernel.

And finally, there are the hyperparameters related to the detection method for behavioural changes.

- D_{lim}: the threshold value for the difference in the LS-SVR detection method.
- l_{det}: number of subsequent trials with a detected change needed for a stable detection.

For optimum performance of the method these hyperparameters will need to be tuned, which will be discussed in Section III-D2.

D. Analysis procedure

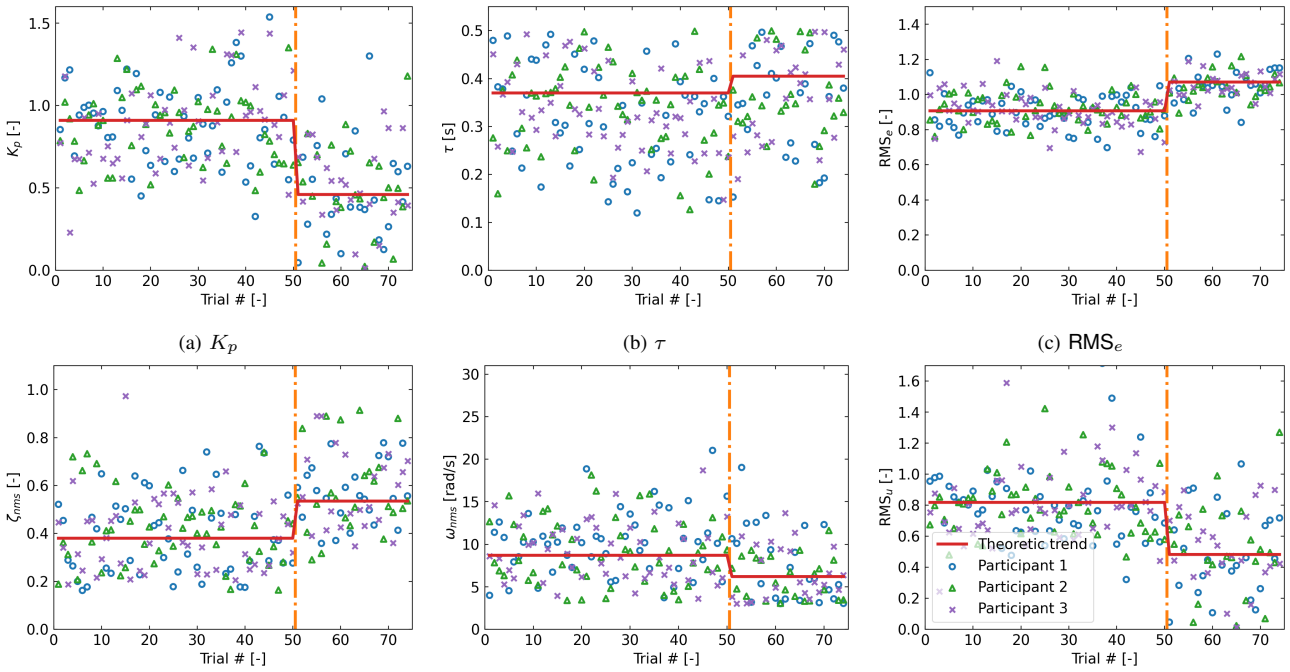
In order to test the novel regression model proposed in this paper, a number of analysis steps will be taken. The goal of these steps is to prove the usefulness of the LS-SVR model, and determine the optimal configuration of its hyperparameters. Furthermore, it will allow for insight into the strengths and weaknesses, and limits of the model. The steps that were taken in the analysis are shown in Figure 6.

1) Detecting changes using LS-SVR: The first step is to test if the LS-SVR model is actually able to detect changes in the trend in the data. For this, two groups of 25 participants are simulated. The first is a healthy group, with 75 simulated healthy trials ($N_h = 75$, $N_s = 0$). The second is a symptomatic group, where $N_h = 50$ and $N_s = 25$. If the model works as intended, for the majority of participants in the second group a change will be detected, while for the participants in the first group no changes should be detected. For simulating the data for the trials, the simulation parameters and δ from Table I and Table II, respectively, are used. The LS-SVR and detection method hyperparameters are given in Table III, and these are the default hyperparameter settings used throughout the paper. γ and σ are varied for each of the six model parameters (K_p, τ, ζ_{nms}, ω_{nms}, RMS_e, RMS_u), as each parameter shows different stochastic variation that requires different hyperparameter settings. D_{lim} is also varied, as its value depends on the range of possible values for the specific model parameter.

A binary classification approach is used for determining detection performance. For the healthy group, a False Positive (FP) is defined as any detection of a change in behavioural data. For the symptomatic group, a False Positive is defined as any detection in the healthy part of the data set and a False Negative occurs when no change is detected.

TABLE III: Default hyperparameter values, obtained heuristically, based on test runs with the model.

	K_p	τ	ζ_{nms}	ω_{nms}	RMS$_e$	RMS$_u$
γ	5	1	5	1	5	40
σ	50	50	100	100	10	100
D_{lim}	0.15	0.03	0.06	1.0	0.1	0.1
l_{det}	5	5	5	5	5	5



(a) K_p (b) τ (c) RMS$_e$

(d) ζ_{nms} (e) ω_{nms} (f) RMS$_u$

Fig. 5: Example of simulated data for three participants, all with the same settings. Data for the first 75 trials are drawn from the N(μ, σ_p) distribution in Table I. Subsequently, for the last 25 symptomatic trials a delta is drawn from the U(a, b) distribution from Table II and added to the already drawn value. The dashed orange line marks the transition from healthy to symptomatic, and the red line indicates the theoretic averages.

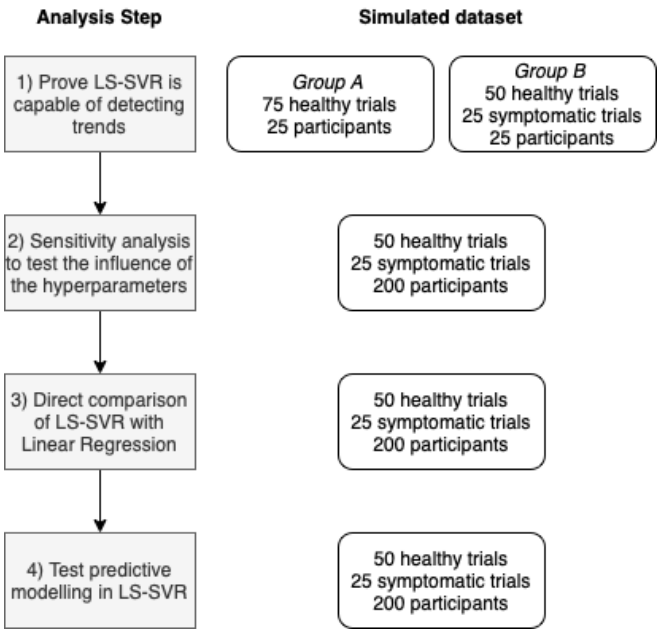


Fig. 6: Schematic overview of the analysis steps taken in this paper, as well as the data set used for each step. The data sets in step 2-4 have the same settings, but are independently generated.

2) Sensitivity analysis: The second step is to perform a sensitivity analysis on the method hyperparameters, in order to determine their influence on the detection performance, and find the optimal values. In this step only the hyperparameters related to the LS-SVR model and the detection method are evaluated: γ, σ, D_{lim}, and l_{det}.

For the sensitivity analysis a single simulated data set is used, consisting of 200 participants. From preliminary experiments it was concluded that this number is sufficient for averaging the stochastic variation between the data sets. Each simulated data sets has 50 healthy trials and 25 symptomatic trials to match the experimental setup used by Lugtenborg [21], with simulation parameters and δ taken from Table I and Table II, respectively. γ and σ together control the LS-SVR regression and therefore the influence on each other's results is taken into account. The other two hyperparameters are expected to be more independent, and are therefore optimised independently the others. Each time, the hyperparameters of interest will be varied along a range, while the other LS-SVR related hyperparameters will be fixed to the same values as described in the first step (Table III).

Two performance parameters will be evaluated, the first a measure for accuracy, and the second one for latency:

- Recall: amount of successful detections out of the 200 detections that should take place.
- Average detection lag: the average trial at which a a change in the trend is first detected. Only detections from trial 50 onward (True Positive) are taken into account, and

all parameters for which no change in trend is detected get an average detection lag penalty value of 50.

The results are averaged over all 200 participants.

3) Direct comparison of LS-SVR with linear regression: Once the hyperparameters have been optimised through sensitivity analysis a direct comparison between the linear regression and the novel LS-SVR method can be made as the third step. For this step 200 participants are simulated, again with 50 healthy trials, 25 symptomatic trials, using the values in Table I and Table II.

The linear regression model is run as described in Section II-B. For the LS-SVR model two cases were initially identified: Case A, with all four hyperparameters optimised for the highest recall, and Case B, all four optimised for the lowest average detection lag. However, during the course of this analysis step it was discovered that the optimum settings identified for Case A and Case B might not lead to overall optimal results. This is likely due to interactions between the different hyperparameters, which are not present when optimised in isolation (as done for D_{lim} and l_{det}). Therefore Case C and Case D are introduced. Here, only the optimal values for the LS-SVR hyperparameters (γ and σ) are taken, with Case C being optimised for highest recall (values from Case A), and Case D being optimised for least average detection lag (values from Case B). For D_{lim} and $l_{\text{det}} = 5$ the dafault values are taken, as given in Table III.

4) Using projected future trials: In the final step the variation on the detection metric is used, as explained in Section II-B2 in order to incorporate projected future trials into the detection method for changes in the trend. The aim of this is to further increase recall and respond quicker to a change in control behaviour. To test this, again 200 participants with a 50/25 split of healthy and symptomatic trials are simulated, using the values in Table I and Table II, and the default hyperparameters from Table III. The model is run with eight different values of n_{future}: [0, 2, 4, 6, 8, 10, 12, 14], with a value of zero corresponding to the standard detection metric.

IV. Results

A. Step 1: Detecting changes using LS-SVR

For testing the method for detecting changes in the behavioural data 25 healthy participants and 25 participants with PD symptoms were simulated. In this step the LS-SVR hyperparameters were not optimised, but chosen heuristically, based on earlier test runs with the model, as explained in Section III-D1.

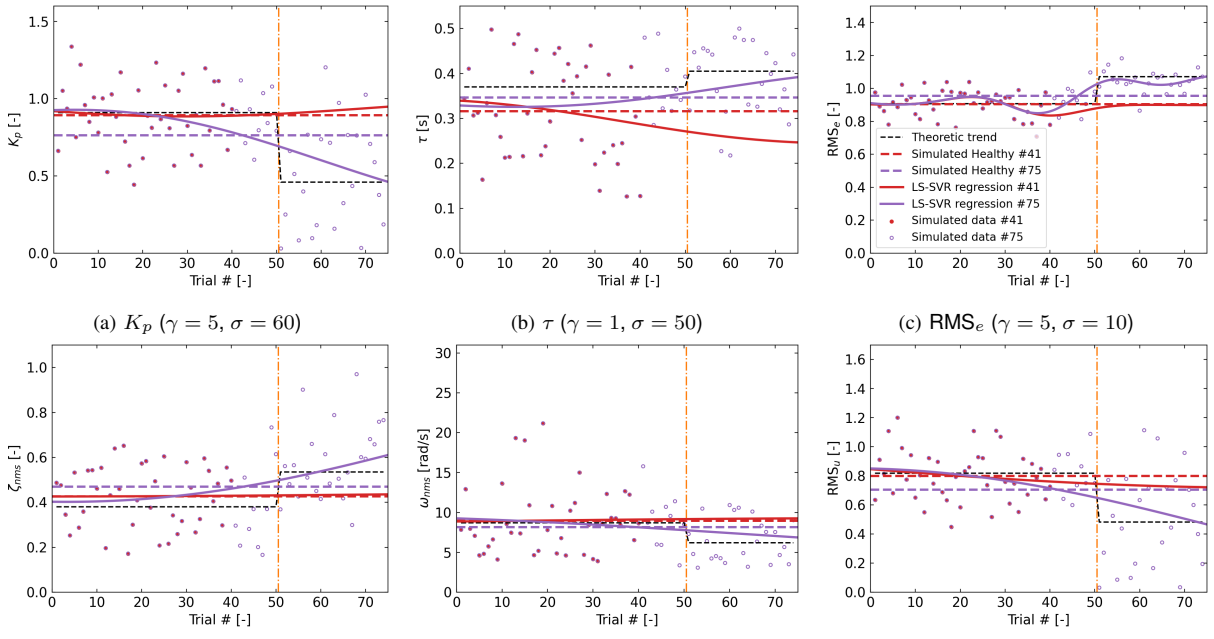
The core of the method proposed in this paper is the LS-SVR regression model. With each trial added in the online learning approach the LS-SVR regression model is retrained and the average of the training data is recalculated, in order to determine D. An example of the regression results for a single (simulated) participant from the symptomatic group can be found in Figure 7. Here the LS-SVR model is shown for two different trials, trial 41 (red) and trial 75 (purple), along with the raw input data, their average, and the theoretic trend in the data. PD symptoms start at trial 50 (similar to the data presented in Figure 5), and the difference between the regression in both trials can be clearly seen, particularly in K_p, ζ_{nms}, and RMS_e. On trial 41, where no PD is present, the regression line deviates much less from the average of the training data (dashed line) as on trial 75, where PD symptoms are present. The influence of the LS-SVR hyperparameters can be seen in the contrast between RMS_e and RMS_u: because of the low value of sigma (10) for RMS_e (compared to 100 for RMS_u) the regression is allowed to more closely track the individual datapoints, which results in a higher order regression.

Figure 8 shows examples of the difference metric (D) for the τ data from Figure 7, plotted for each trial (x), for different participants. When the difference is more than D_{lim} for five consecutive trials a change in behavioural data is detected, indicated with an arrow at the start of the five trials. For the healthy group (Figure 8a), participant 1 shows no changes. But for participant 9 a clear changing trend is detected at trial 65, while there should not be one. This is an example of a false positive: due to the particular combination of data and hyperparameter settings for this participant, the LS-SVR model deviates considerably from the training data average. For the symptomatic group three examples are given in Figure 8b. Participant 4 shows no trend: D never comes above the threshold. This is an example of a false negative. For participant 5 a change is detected as expected: from trial 61 onward the difference stays above the threshold. For this particular case the detection lag is 11. In contrast, for participant 18 a false positive detection occurs: the difference comes above D_{lim} before symptoms are introduced, but does not stay there.

The full results of the first analysis step are summarised in the confusion matrix in Table IV. The desired result is true negative (TN) for the healthy group, and true positive (TP) for the symptomatic group. As can be seen in Table IV, for the healthy group the desired result is reached in the majority of the cases: 64% for RMS_u, as lowest, 96% for K_p, as highest. For the symptomatic group this is also the case for K_p (96%), ζ_{nms} (84%), RMS_e (72%), and RMS_u (80%). However, this is not the case for τ and ω_{nms}, where there are approximately equal true positives and false negatives, meaning changing trends are only detected in about half of the cases. This result is explained by the first analysis step being performed with unoptimised parameters, leading to suboptimal results.

B. Step 2: Sensitivity analysis

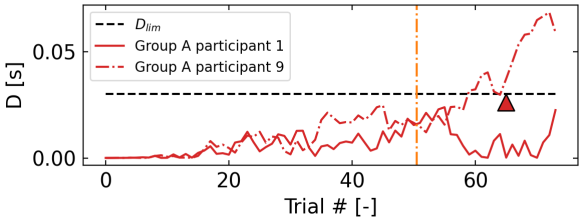
1) Sensitivity analysis of γ and σ: In order to optimise the detection results presented in part A the model hyperparameters have to be optimised. This is done through a sensitivity analysis. First the results of the optimisation of γ and σ are given. Figure 9 shows the results in terms of total detections, and Figure 10 for average detection lag. In



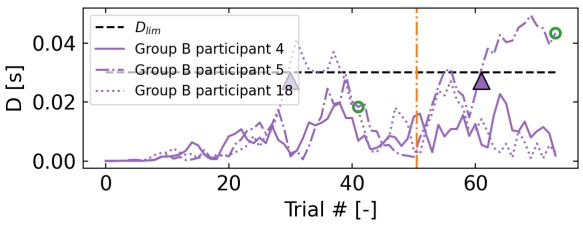
(a) K_p ($\gamma = 5$, $\sigma = 60$) (b) τ ($\gamma = 1$, $\sigma = 50$) (c) RMS$_e$ ($\gamma = 5$, $\sigma = 10$)

(d) ζ_{nms} ($\gamma = 5$, $\sigma = 100$) (e) ω_{nms} ($\gamma = 1$, $\sigma = 100$) (f) RMS$_u$ ($\gamma = 40$, $\sigma = 100$)

Fig. 7: LS-SVR regression results for participant 5 in the symptomatic group of step 1. The red lines indicate the results from trial 41, and the purple lines the results from trial 75.



(a) Difference over time for 2 participants from the healthy group



(b) Difference over time for 3 participants from the symptomatic group

Fig. 8: Results from the detection procedure in step 1 for τ, for different participants. The triangles show where a change in trend is detected, and the green circles indicate the values for D corresponding to the two regression results in Figure 7.

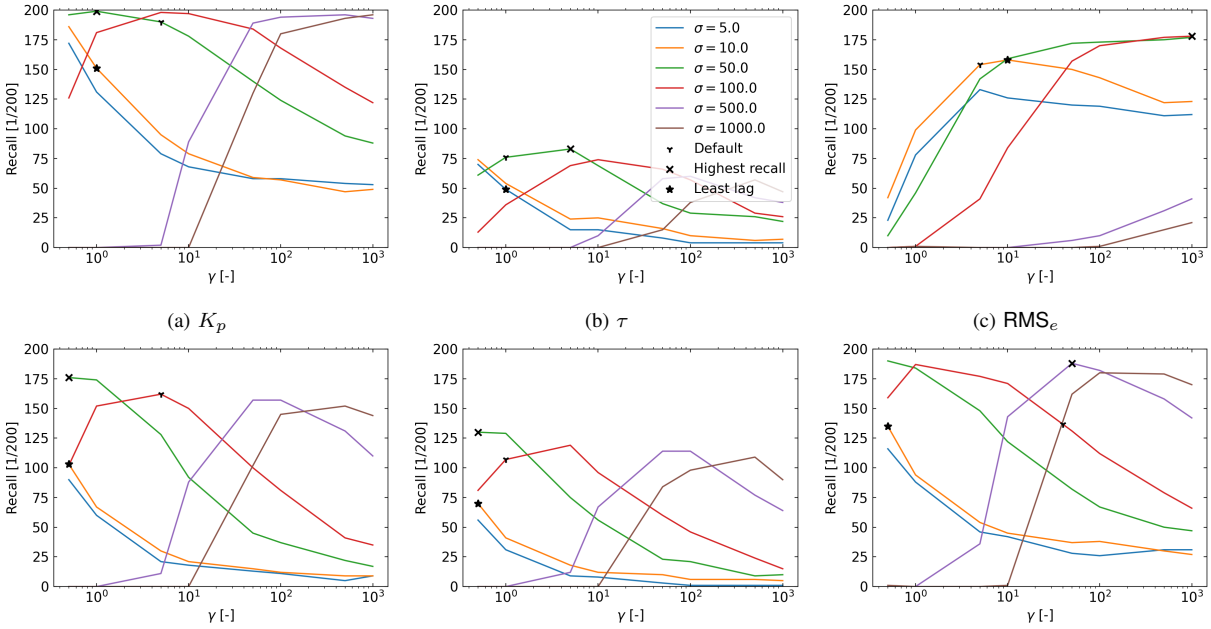
the graphs, γ is plotted along the x-axis, with results from different values of σ shown as separated lines. For the total detections (Figure 9) clear optima can be found: for example, τ shows an optimum recall at $\gamma \approx 5$ and $\sigma = 50$. For many parameters optima can be found in multiple combinations of hyperparameter values, with generally higher values of

TABLE IV: Confusion matrix per parameter for the healthy and the symptomatic group in Step 1, as percentage of the total of 25 simulated participants. The healthy group contains no trend changes in the behavioural data, so there are no True Positives. Similarly, for the symptomatic group in every simulated participant a trend change is present, so there are no True Negatives.

		K_p	τ	ζ_{nms}	ω_{nms}	RMS$_e$	RMS$_u$
Healthy Group	TP	0%	0%	0%	0%	0%	0%
	FP	4%	20%	12%	12%	4%	36%
	TN	96%	80%	88%	88%	96%	64%
	FN	0%	0%	0%	0%	0%	0%
Sympto-matic Group	TP	96%	44%	84%	40%	72%	80%
	FP	4%	8%	12%	12%	8%	20%
	TN	0%	0%	0%	0%	0%	0%
	FN	0%	48%	4%	48%	20%	0%

σ requiring higher values of γ for optimality. There are also combinations of hyperparameter values that lead to no detections, e.g., combinations of low γ and high σ. For the average detection lag results (Figure 10) this is taken into account: combinations with no detections are left out when calculating the detection lag. This can for example be seen in RMS$_e$ (Figure 10c), where for $\sigma = 500$ and $\sigma = 1000$ only results for $\gamma > \sim 50$ are given. In the average detection lag graphs it can be seen that in general higher values for γ and lower values for σ lead to less detection lag. This was expected, as this combination of hyperparameter values leads to the highest order regression, allowing for the quickest response to changes.

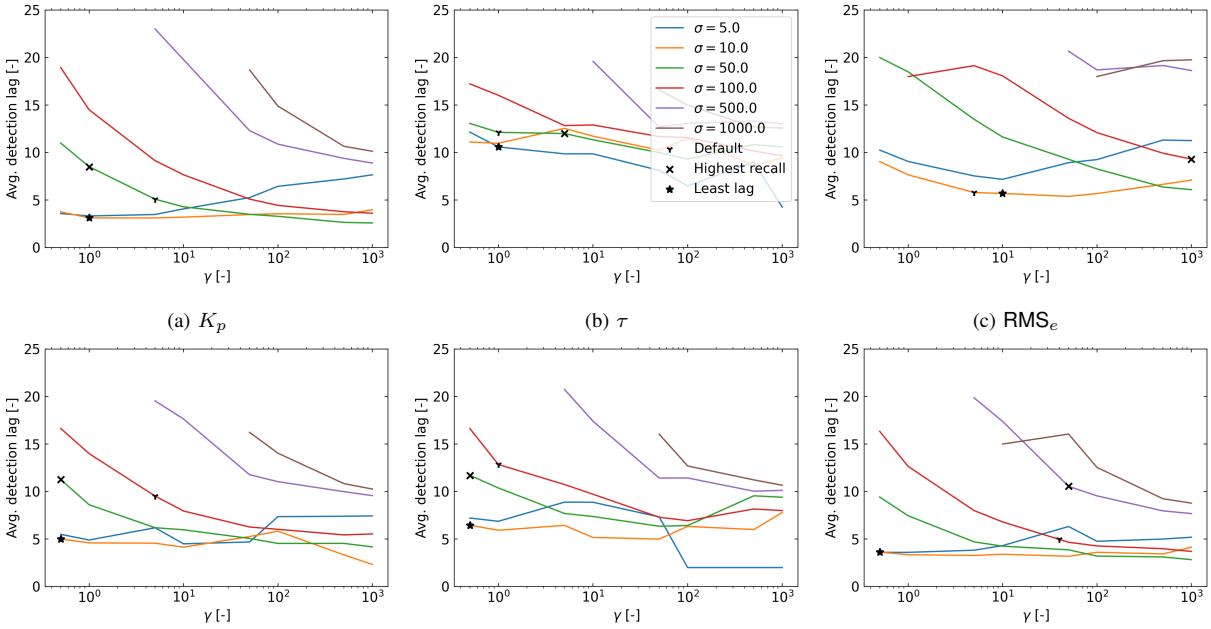
2) Sensitivity analysis of D_{lim} and l_{det}: Sensitivity analysis is also performed for the two detection method hyperparameters, D_{lim} (Figure 11) and l_{det} (Figure 12). For



(a) K_p (b) τ (c) RMS$_e$

(d) ζ_{nms} (e) ω_{nms} (f) RMS$_u$

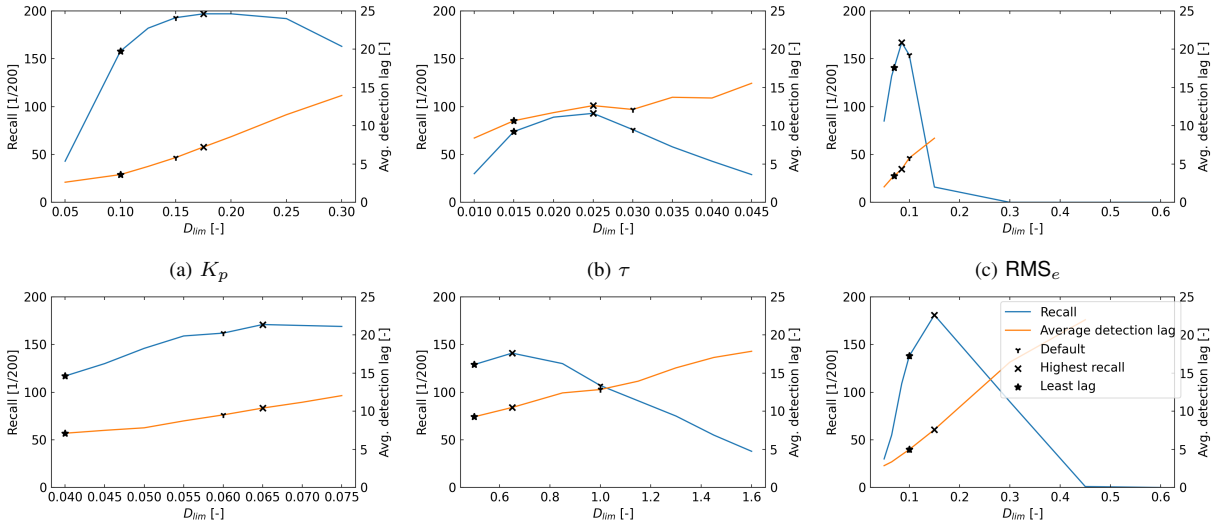
Fig. 9: Sensitivity analysis results for different values of γ and σ, recall. The markers indicate default hyperparameter values (Table III), as well as those optimised for highest recall (Table Va) and for least detection lag (Table Vb).



(a) K_p (b) τ (c) RMS$_e$

(d) ζ_{nms} (e) ω_{nms} (f) RMS$_u$

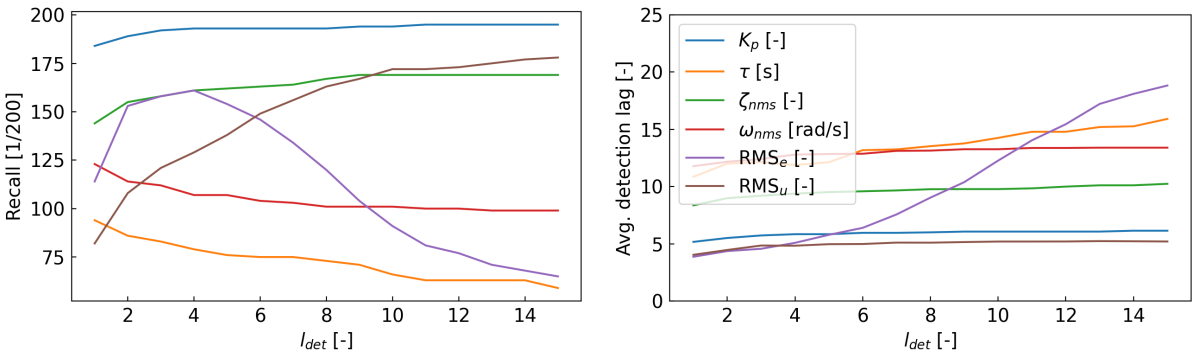
Fig. 10: Sensitivity analysis results for different values of γ and σ, average detection lag. The markers indicate default hyperparameter values (Table III), as well as those optimised for highest recall (Table Va) and for least detection lag (Table Vb).



(a) K_p (b) τ (c) RMS_e

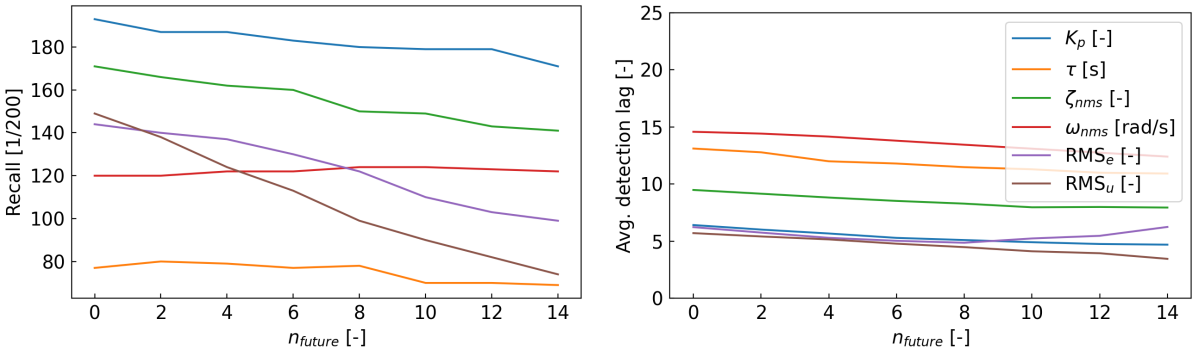
(d) ζ_{nms} (e) ω_{nms} (f) RMS_u

Fig. 11: Sensitivity analysis results for different values of D_{lim}, both recall and average detection lag. The markers indicate default hyperparameter values (Table III), as well as those optimised for highest recall (Table Va) and for least detection lag (Table Vb).



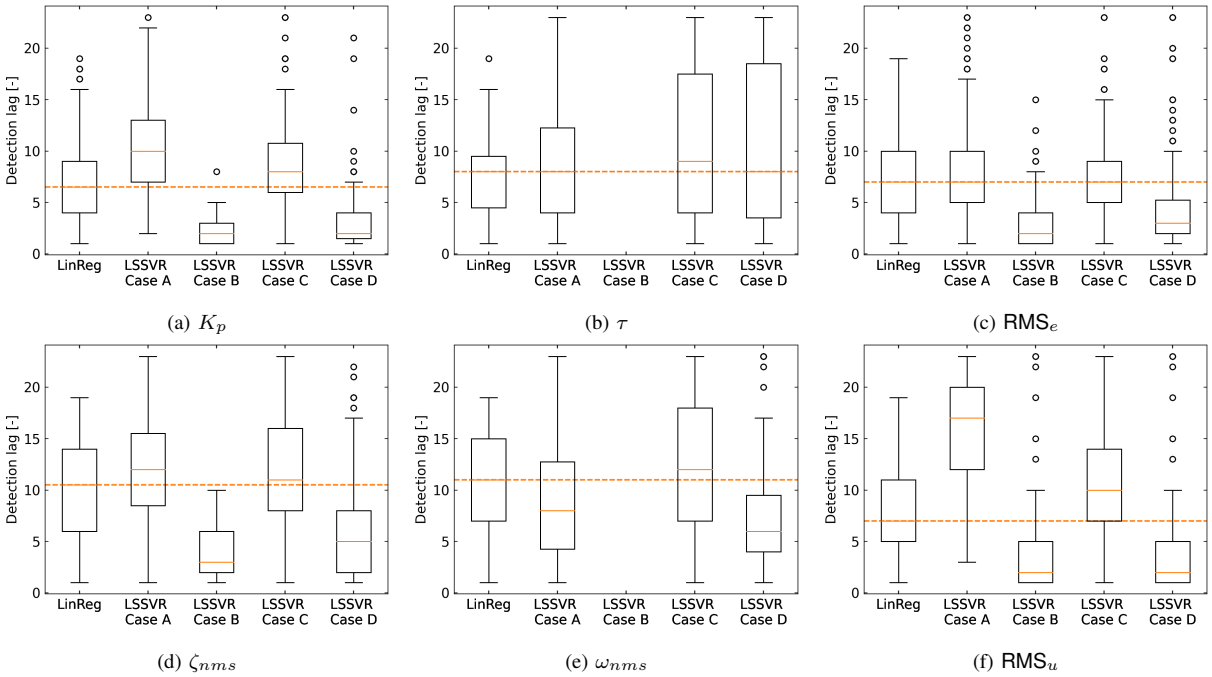
(a) Recall per parameter (b) Average detection lag per parameter

Fig. 12: Sensitivity analysis results for l_{det}



(a) Recall per parameter (b) Average detection lag per parameter

Fig. 13: Recall and average detection lag for varying n_{future}



(a) K_p (b) τ (c) RMS$_e$

(d) ζ_{nms} (e) ω_{nms} (f) RMS$_u$

Fig. 14: Boxplot of detection lag for each of the four LS-SVR cases, and linear regression. The dashed horizontal line indicates the median for linear regression.

D_{lim} optima can be found in the recall, while the average detection lag increases for higher values of D_{lim}. This is expected, as a higher threshold for detection means the difference has to increase more. As a result it takes more trials for the difference to pass the threshold value. The influence of l_{det} on the detection results is less prominent. For all but RMS$_e$ (in both cases) and RMS$_u$ (for recall) the results stay approximately constant.

3) Selecting optimal hyperparameter values: Once the sensitivity analysis has been completed, the optimal values could be selected, and used in subsequent analysis steps. Two optimised settings were taken: one optimised setting solely for the highest recall (Case A), and a second, heuristically optimised setting for the lowest average lag (Case B). For Case A the values with highest recall were selected, while for Case B the lowest value of average detection lag was taken as leading, but it was checked if there is sufficient recall for this combination of hyperparameters. For example, when looking at ζ_{nms}, a combination of $\gamma = 1000$ and $\sigma = 10$ leads to the least detection lag (Figure 10d). However, here the recall is only $10/25$, therefore this combination of values was deemed insufficient. For l_{det} a minimum of three subsequent trials is taken. The selected hyperparameter values for the two cases are given in Tables Va and Vb, respectively. As described in Section III-D3, Case C and D were introduced as well. Their hyperparameter values are given in Tables Vc and Vd, respectively.

TABLE V: Hyperparameter values for the different cases in Step 3

(a) Case A, all optimised for highest recall

	K_p	τ	ζ_{nms}	ω_{nms}	RMS$_e$	RMS$_u$
γ	1	5	0.5	0.5	1000	50
σ	50	50	50	50	100	500
D_{lim}	0.175	0.025	0.065	0.65	0.085	0.15
l_{det}	5	3	9	3	4	15

(b) Case B, all optimised for least lag

	K_p	τ	ζ_{nms}	ω_{nms}	RMS$_e$	RMS$_u$
γ	1	1	0.5	0.5	10	0.5
σ	10	5	10	10	10	10
D_{lim}	0.1	0.015	0.040	0.5	0.07	0.1
l_{det}	3	3	3	3	3	5

(c) Case C, γ and σ optimised for highest recall

	K_p	τ	ζ_{nms}	ω_{nms}	RMS$_e$	RMS$_u$
γ	1	5	0.5	0.5	1000	50
σ	50	50	50	50	100	500
D_{lim}	0.15	0.03	0.06	1.0	0.1	0.1
l_{det}	5	5	5	5	5	5

(d) Case D, γ and σ optimised for least lag

	K_p	τ	ζ_{nms}	ω_{nms}	RMS$_e$	RMS$_u$
γ	1	1	0.5	0.5	10	0.5
σ	10	5	10	10	10	10
D_{lim}	0.15	0.03	0.06	1.0	0.1	0.1
l_{det}	5	5	5	5	5	5

C. Step 3: Direct comparison of LS-SVR with linear regression

With the optimised values for the hyperparameters a comparison between detection using an LS-SVR model and a linear regression model can be made to see if the former is an improvement on the latter. The data set consists of 200 simulations of a PD patient, and behavioural change detection was performed using linear regression, as well as Case A and Case B for LS-SVR. During the analysis it was discovered that the optimum settings identified for Case A and Case B

might not lead to optimal results, as explained in Section III-D2, therefore Case C and D were introduced as well.

Table VI gives the confusion matrix for these results. For most parameters, optimising for highest recall (Case A and C) will indeed lead to more detections in LS-SVR than for linear regression. For example, for ω_{nms}, linear regression has a 28% recall, versus 61% and 68% for Case A and C respectively. This is the case for all parameters in Case C, and all but RMS_u in Case A (67.5% vs 70% for linear regression). Another interesting point with LS-SVR is that False Negatives are rare, especially in Case B and D (less than 5%). However, for these cases (when optimising for the lowest average detection lag, Case B and D) a lot of the detections are False Positives: meaning they occur before the point were symptoms are actually introduced. For Case B, only RMS_e and RMS_u show an adequate detection performance (39.5% and 66.5% respectively), with RMS_u being close to the linear regression. Case D has around the same recall as the linear regression model, while on average detecting changes much earlier. This implies that LS-SVR has a much higher potential than linear regression, since for Case D only γ and σ were optimised. Even better results can be obtained with better hyperparameter selection for D_{lim} and l_{det}.

The detection lag results can be found in Figure 14 which gives a boxplot of the detection lag. For the cases optimised for minimum detection lag, Case B (K_p, RMS_e and RMS_u) and Case D (all parameters), the median detection lag is between 4 and 7 trials lower than for linear regression or LS-SVR case A and C (an up to 30% improvement), and the spread is less as well. Only τ is an exception, as for this parameter the median is similar for all cases, while the spread is actually lowest for linear regression. Interesting to note is that for Case B there were no detections in τ en ω_{nms}. This can also be seen in Table VI: the regression becomes too sensitive, and only results in False Positives. Furthermore, for Case A and Case C, optimised for highest recall, it depends on the selected hyperparameters if they perform better than linear regression. For example, linear regression results in earlier detections for K_p, ζ_{nms}, and RMS_u, but for τ, ω_{nms}, and RMS_e the results between linear regression and LS-SVR are more similar. Finally, it can be seen that for Case D, optimised for average detection lag, results in terms of recall are quite similar to the results for linear regression, while the average detection lag is significantly lower.

D. Step 4: Using projected future trials

The results of using data from projected future trials in the regression is given in Figure 13. It shows the recall and average detection lag for different values of n_{future}. For all parameters it can be seen that the average detection lag decreases with 2-3 trials when looking further ahead, as was expected. For all parameters except RMS_e the lag keeps decreasing, while for RMS_e there is an optimum at $n_{future} = 8$. It can also be seen that the recall goes down as well, with RMS_e and RMS_u showing the greatest decline. Only for ω_{nms} does the recall slightly increase. This is unexpected, and the reasoning will have to be investigated.

TABLE VI: Confusion matrix per parameter for the different regression models in step 3, as percentage of the total of 200 simulated participants. Note that there are no True Negatives in the data set.

		K_p	τ	ζ_{nms}	ω_{nms}	RMS_e	RMS_u
Linear regression	TP	75%	11.5%	58%	28%	75%	70%
	FP	22.5%	16%	20%	19%	21%	20.5%
	FN	2.5%	72.5%	22%	53%	4%	9.5%
LS-SVR Case A	TP	98%	24%	85.5%	61%	91%	67.5%
	FP	0%	72.5%	2.5%	34.5%	6%	3%
	FN	2%	3.5%	12%	4.5%	3%	29.5%
LS-SVR Case B	TP	14%	0%	4.5%	0%	39.5%	66.5%
	FP	86%	100%	95.5%	100%	60.5%	33.5%
	FN	0%	0%	0%	0%	0%	0%
LS-SVR Case C	TP	99%	36%	86.5%	68%	90.5%	88.5%
	FP	0.5%	46.5%	2.5%	8%	9%	5%
	FN	0.5%	17.5%	11%	24%	0.5%	6.5%
LS-SVR Case D	TP	73.5%	23.5%	57.5%	27.5%	76%	66.5%
	FP	26.5%	72%	42%	69.5%	21.5%	33.5%
	FN	0%	4.5%	0.5%	3%	2.5%	0%

An example of the change in the difference for varying n_{future} is given in Figure 15. The results are given for K_p, as this parameter consistently shows the best results, and is indicative of what can be achieved. Here it can be seen that by looking further ahead the difference graph shifts to higher values of $\bar{D}$, and the threshold (D_{lim}) is crossed earlier. Furthermore, in Table VII the confusion matrix for K_p is given. When n_{future} increases the amount of False Positives increases, from 3.5% (n_{future} = 0) to 14.5% (n_{future} = 14.5). This is unexpected, and needs to be investigated further.

TABLE VII: Confusion matrix for K_p in Step 4, as percentage of the total of 200 simulated participants

n_{future}	0	2	4	6
TP	96.5%	93.5%	93.5%	91.5%
FP	3.5%	6.5%	6.5%	8.5%
FN	0%	0%	0%	0%

n_{future}	8	10	12	14
TP	90%	89.5%	89.5%	85.5%
FP	10%	10.5%	10.5%	14.5%
FN	0%	0%	0%	0%

V. Discussion and Recommendations

A. Discussion

The goal of this paper was to investigate LS-SVR as a potential improvement upon the linear regression model used in by Lugtenborg for detecting behavioural changes in PD progression [21]. By generating custom data sets different configurations were tested, and the effect of changing the regression model on the detection results was directly compared in terms of recall and average detection lag.

In Step 1 the ability of LS-SVR to detect changes was tested by comparing the detection results of a healthy group of simulated participants with a symptomatic group. For the

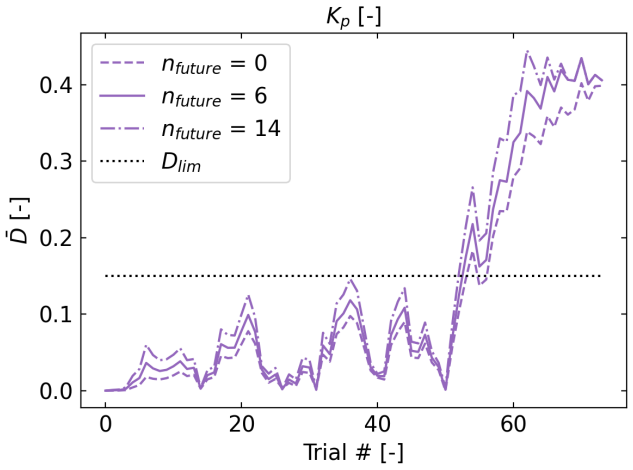


Fig. 15: Difference graph for varying n_{future}. Only K_p is shown, for a single simulated participant. The horizontal line is the threshold, D_{lim}.

healthy participants, the desired result (True Negative) was obtained in at least 80% of cases, for all parameters except RMS_u (which still has over 60% accuracy). It shows the ability of LS-SVR, with careful hyperparameter setting, to filter out the day-to-day variation present in the data, and only to react to actual changes in the control performance over time. This is confirmed when looking at the simulated patients. In particular, for K_p, ζ_{nms}, RMS_e and RMS_u a changing trend is accurately detected (True Positive) in more than 70% of cases. This indicates an improvement over the results in earlier work, where more than 50% of cases were True Positives [21]. In general, this analysis step shows that LS-SVR is able to accurately detect the changes in behavioural data from simulated in PD patients, and correctly detects no changes in data from simulated healthy participants.

These initial results are promising, but the goal of this paper was to investigate a potential improvement of LS-SVR over linear regression. Therefore a direct comparison was made between linear regression and four different configurations of the LS-SVR model (Case A-D, as explained in Section III-D3). When looking at recall, LS-SVR (Case A and C) performs better than linear regression, with Case A showing a an average increase of 50% across all parameters, and Case C 80%. While this was expected, as these cases are optimised for highest recall, the difference is still large. Especially when looking at the two worst performing parameters overall, τ and ω_{nms}, more than the double amount of detections take place, and for τ in Case D even triple. This shows that detection based on LS-SVR is more sensitive than when using a linear regression model, and more capable of distinguishing between the natural variation and a trend due to PD. Interesting to note is that even when optimised for low detection lag, as in Case C, the recall of LS-SVR is similar to that of the linear regression approach from the earlier work [21].

When looking at average detection lag, it can also be seen that LS-SVR is an improvement over linear regression. Linear regression on the dataset used for the experiment in step 3 has an average detection lag across all parameters of ca. 8 trials. This is slightly better than Lugtenborg's results, were the average was 10 trials[21]. However, LS-SVR shows better results. For both Case B and D the average detection lag is significantly lower, across all parameters, except τ: ca. 2 trials on average for Case B, and ca. 4 for Case D. However, for Case B the recall is really poor. Only in RMS_e are changes being detected over half of the time. For τ and ω_{nms} the recall is 0/200, and most runs result in a False Positive detection. This can be explained by the nature of the LS-SVR model when optimised for the least detection lag, as in Case B. In such cases the regression is highly sensitive to changes: it will take on a higher order, and track the natural variation in the data more closely. This leads to more trend changes being detected in the healthy part of the data. For Case D a balance has been found, in which the majority of detections are True Positives, with the best RMS_e having the highest recall of 152/200. Only τ and ω_{nms} have a low recall (47/200 and 52/200 respectively). In addition to this higher recall than for Case B, Case D is still reacting to changes faster than LS-SVR, with a 4 trial improvement. Another difference between the detection performance of linear regression and LS-SVR is that the spread of the average detection lag is in general less for Case B and D. This indicates better consistency for LS-SVR. In general it can be concluded that LS-SVR performs better than linear regression in both recall and average detection lag.

However, the specifics of the results are highly dependant on the model parameter, as well as the hyperparameter settings. From the results it can be concluded that K_p, RMS_e, and RMS_u, and to a certain extent ζ_{nms}, show the best performance in detecting behavioural changes, with ω_{nms} and in particular τ showing less changes detected. This can be explained by comparing the mean and standard deviation for data simulation and δ for these parameters (Tables I and II). For K_p the average δ is double the standard deviation, while for τ the average δ is four times smaller. This means that the natural variation in τ is more likely to mask the actual change due to PD in the data than with K_p. This is in line with the findings in earlier work [23, 21], where the difference between healthy participants and PD patients is greatest in K_p and smallest in τ. For further study towards clinical application it is thus recommended to focus on these high performing parameters.

When comparing the new LS-SVR model proposed in this paper with the linear regression model used in earlier work a definite improvement can be found. However, there is a large influence of the value of the hyperparameters, γ, σ, D_{lim}, and l_{det}, on the results, both in terms of recall and average detection lag, and understanding this influence would help setting up the method in this paper for best detection performance. As explained in Section IV-B, optimal values in the individual hyperparameters can be found. However, the sensitivity analysis in this paper is done individually. From

the results it can be concluded that the hyperparameters are highly dependant upon the values of the other parameters. An indication of this can already be found in the results for the sensitivity analysis for γ and σ: changing the value of one significantly the effect of the other. This was expected, as both parameters control the order of LS-SVR regression. However, from the results in step 3, when comparing Case A and C and Case B and D, it can be concluded that the interaction with the values for D_{lim} and l_{det} is of greater influence than expected. Only taking the optimised values for γ and σ, and setting D_{lim} and l_{det} to the value used in the sensitivity analysis, leads to better results than taking the optimised values for all four hyperparameters. In a future study a complete dependent sensitivity analysis is needed for even better performance of the detection method using the LS-SVR model.

The final step was to use data from predicted trials to improve the detection performance. The results from this step show that this is indeed the case: behavioural changes are indeed detected earlier by looking further ahead. Though the difference was only 2-3 trials, this can still be clinically relevant. On the other hand, the amount of False Positives detections increased when increasing n_{future}. The results for using projected future trials are a good first step. However, there are still some unknowns. For example, the interaction with the different hyperparameters, especially l_{det}, could be of great influence on the results. Further research is required before this aspect of the detection method can be fully utilised. With this step the trade-off between optimising the model for a high recall or for low detection lag was clearly influential. This trade-off has been comping up throughout this paper. For clinical application one wants to have an as early detection as possible, but not at the cost of too many False Positives, and False Negatives need to be avoided as much as possible. This requires careful tuning of all hyperparameters, and further research should focus upon expanding on the work done in this paper and develop a working prototype for clinical application.

B. Recommendations for future research

The results obtained in this research show that LS-SVR is an improvement upon linear regression for detecting behavioural changes in PD tracking data. As a proof-of-concept the model used in this paper shows great promise. However, more research is needed before this approach can be used in a clinical setting. There are three main areas on which future research should focus.

Firstly, for this paper a simulated data set was used, based upon the measured data from earlier work [21]. While the data simulation procedure accurately reflects the average measured data taken, all the simulated participants used in this paper are based on a single normal distribution for each parameter (Table I), which might lead to biased results. Additionally, the method of simulating PD symptoms used in this research is not completely reflective of reality. While this paper uses a step function to simulate a sudden decline in control performance, in real patients this decline would be more gradual. It is expected that LS-SVR would still be an excellent way of tracking this decline, as the underlying distribution is irrelevant to the regression algorithm, and with the right combination of hyperparameters any distribution can be modelled [42]. This will have to be tested with an extended simulation data set with different parameters for the simulation and different underlying trends in the data. Furthermore, validation with measured data from a varying group of healthy people and PD patients is desired.

Secondly, the work in this paper focuses on a general group analysis. Most results are in the form of averages and accumulations, and the way the hyperparameter values have been determined is based upon these results. Furthermore, the procedure for differentiating between True Positives and False Positives in this paper is based upon prior knowledge (the trial number at which PD simulated symptoms start). For practical application in a clinical setting however, the model should be applicable to the individual. Hyperparameter settings and other model details will vary from person to person, and a procedure for finding their values will have to be created. A recommended first next step is thus to gather data from actual patients, and develop a method for hyperparameter tuning that leads to the desired results in terms of detection lag, and the amount of False Positives and True Negatives.

Lastly, in this paper, analysis is based on the individual model parameters. Combining the model parameters into a multivariate regression model might lead to even better results. Such multivariate (Least Squares) Support Vector Regression models are used in literature to predict an outcome variable, such as air pollution [47] and brain lesion mapping [48]. For this research project a multivariate model could be used to predict a PD-score, for example relating it to the MDS-UPDRS, or developing a new scoring metric.

VI. Conclusions

The goal of this study was to investigate the potential of Least Squares Support Vector Regression (LS-SVR) as an improvement over linear regression in detecting behavioural changes in control performance due to Parkinson's Disease. The new model was compared in terms of detection accuracy and latency, and the effect of its hyperparameter settings on the detection performance of the method was investigated.

From this research it can be concluded that LS-SVR is indeed an improvement, detecting changes in behavioural data earlier and more often: depending on the hyperparameter settings an improvement of 50-80% in accuracy and a 30% improvement in latency was found. In addition, the sensitivity analysis done in this research revealed that with optimal tuning of the hyperparameters even better results can be reached. Finally, the effect of using data from predicted future trials was investigated, showing potential for further reducing latency.

Further research steps should focus upon experiments with actual patients to validate the work done in this paper and to develop procedures for individual hyperparameter tuning. Overall, LS-SVR shows definite improvement over the earlier linear regression model and this paper is a further step towards a clinical applicable method.

REFERENCES

[1] L. Kalia and A. Lang, "Parkinson's disease," *The Lancet*, vol. 386, no. 9996, pp. 896–912, 2015.

[2] J. Palfreman, *Hersenstormen: De ziekte van Parkinson en de raadselen van het brein*. (I. Pieters, trans.). Amsterdam, The Netherlands: Uitgeverij Balans, 2016.

[3] N. Eriksen, A. Stark, and B. Pakkenberg, "Age and Parkinson's Disease-Related Neuronal Death in the Substantia Nigra Pars Compacta," in *Birth, life and death of dopaminergic neurons in the substantia nigra*, G. D. Giovanni, V. D. Matteo, and E. Esposito, Eds. New York, USA: Springer, 2009, ch. 16, pp. 203–213.

[4] J. Jankovic, "Parkinson's disease: Clinical features and diagnosis," *Journal of Neurology, Neurosurgery and Psychiatry*, vol. 79, no. 4, pp. 368–376, 2008.

[5] N. Singh, V. Pillay, and Y. Choonara, "Advances in the treatment of Parkinson's disease," *Progress in Neurobiology*, vol. 81, no. 1, pp. 29–44, 2007.

[6] D. Gelb, E. Oliver, and S. Gilman, "Diagnostic Criteria for Parkinson Disease," *Archives of Neurology*, vol. 56, no. 1, pp. 33–39, 1999.

[7] C. Goetz, B. Tilley, S. Shaftman *et al.*, "Movement Disorder Society-Sponsored Revision of the Unified Parkinson's Disease Rating Scale (MDS-UPDRS): Scale presentation and clinimetric testing results," *Movement Disorders*, vol. 23, no. 15, pp. 2129–2170, 2008.

[8] M. Myszczynska, P. Ojamies, A. Lacoste *et al.*, "Applications of machine learning to diagnosis and treatment of neurodegenerative diseases." *Nature Reviews Neurology*, vol. 16, no. 8, pp. 440–456, 2020.

[9] A. Espay, P. Bonato, F. Nahab *et al.*, "Technology in Parkinson's Disease: Challenges and Opportunities Alberto," *Movement Disorders*, vol. 31, no. 9, pp. 1272–1282, 2016.

[10] A. Sánchez-Ferro, M. Elshehabi, C. Godinho *et al.*, "New Methods for the Assessment of Parkinson's Disease (2005 to 2015): A Systematic Review," *Movement Disorders*, vol. 31, no. 9, pp. 1283–1292, 2016.

[11] C. Brooks, G. Edena, A. Changa *et al.*, "Quantification of discrete behavioral components of the MDS-UPDRS," *Journal of Clinical Neuroscience*, vol. 61, pp. 174–179, 2019.

[12] S. Patel, K. Lorincz, R. Hughes *et al.*, "Monitoring Motor Fluctuations in Patients With Parkinson's Disease Using Wearable Sensors," *IEEE Transactions on Information Technology in Biomedicine*, vol. 13, no. 6, pp. 864–873, 2009.

[13] A. Zhan, S. Mohan, C. Tarolli *et al.*, "Using Smartphones and Machine Learning to Quantify Parkinson Disease Severity: The Mobile Parkinson Disease Score," *JAMA Neurology*, vol. 75, no. 7, pp. 876–880, 2018.

[14] C. de Boer, *Visuomotor integration in neurodegenerative brains*. PHD Thesis. Rotterdam, The Netherlands: Erasmus University Rotterdam, Neuroscience Department, 2015.

[15] K. Flowers, "Lack of prediction in the motor behaviour of parkinsonism," *Brain*, vol. 101, no. 1, pp. 35–52, 1978.

[16] ——, "Some frequency response characteristics of parkinsonism on pursuit tracking," *Brain*, vol. 101, no. 1, pp. 19–34, 1978.

[17] ——, "Visual 'Closed-Loop' and 'Open-Loop' Characteristics of Voluntary Movement in Patients With Parkinsonism and Intention Tremor," *Brain*, vol. 101, no. 1, pp. 19–34, 1978.

[18] R. Jones and I. Donaldson, "Tracking tasks and the study of predictive motor planning in Parkinson's disease," *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, vol. 11 pt 3, pp. 1055–1056, 1989.

[19] A. Hufschmidt and C. Lucking, "Abnormalities of tracking behavior in Parkinson's Disease," *Movement Disorders*, vol. 10, no. 3, pp. 267–276, 1995.

[20] M. Oishi, P. Talebifard, and M. McKeown, "Assessing manual pursuit tracking in Parkinson's disease via linear dynamical systems," *Annals of Biomedical Engineering*, vol. 39, no. 8, pp. 2263–2273, 2011.

[21] L. Lugtenborg, *Identifying Behavioural Changes due to Parkinson's Disease Progression in Motor Performance Data*. MSc Thesis. Delft, The Netherlands: Delft University of Technology, Faculty of Aerospace Engineering, 2020.

[22] M. Mulder, D. Pool, D. Abbink *et al.*, "Manual Control Cybernetics: State-of-the-Art and Current Trends." *IEEE Transactions on Human-Machine Systems*, vol. 48, no. 5, p. 468–485, 2018.

[23] R. de Vries, *Using Human Operator Modeling for Quantifying Loss of Motor Skills due to Parkinson's Disease*. MSc Thesis. Delft, The Netherlands: Delft University of Technology, Faculty of Aerospace Engineering, 2016.

[24] R. Chandler and E. Scott, *Statistical methods for trend detection and analysis in the environmental sciences*, 1st ed. Chichester, UK: John Wiley & Sons, Ltd., 2011.

[25] C. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer Science+Business Media, 2006.

[26] D. Bzdok, N. Altman, and M. Krzywinski, "Statistics versus machine learning." *Nature Methods*, vol. 15, p. 233–234, 2018.

[27] H. Rajula, G. Verlato, M. Manchia, N. Antonucci, and V. Fanos, "Comparison of Conventional Statistical Methods with Machine Learning in Medicine: Diagnosis, Drug Development, and Treatment." *Medicina (Kaunas)*, vol. 56, no. 9, p. 455, 2020 Sep.

[28] J. Suykens, T. van Gestel, J. D. Brabanter, B. de Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific Publishing Company, 2002.

[29] S. R. Gunn, *Support vector machines for classification and regression*. Southampton, UK: Technical Report,

Department of Electronics and Computer Science. University of Southampton, 1998.

[30] A. Smola and B. Schölkopf, "A tutorial on support vector regression." *Statistics and Computing*, vol. 14, pp. 199–222, 2004.

[31] Z. Guo and G. Bai, "Application of Least Squares Support Vector Machine for Regression to Reliability Analysis," *Chinese Journal of Aeronautics*, vol. 22, no. 2, pp. 160–166, 2009.

[32] M. Bermúdez, L. Cea, and J. Puertas, "A rapid flood inundation model for hazard mapping based on least squares support vector machine regression," *Journal of Flood Risk Management*, vol. 12, no. S1, 2019.

[33] Y. Gu, W. Zhao, and Z. Wu, "Online adaptive least squares support vector machine and its application in utility boiler combustion optimization systems," *Journal of Process Control*, vol. 21, no. 7, pp. 1040–1048, 2011.

[34] Ömer Eskidere, F. Ertas, and C. Hanilci, "A comparison of regression methods for remote tracking of Parkinson's disease progression," *Expert Systems with Applications*, vol. 39, no. 5, pp. 5523–5528, 2012.

[35] M. Salmanpour, M. Shamsaei, A. Saberi *et al.*, "Machine learning methods for optimal prediction of motor outcome in Parkinson's disease," *Physica Medica*, vol. 69, pp. 233–240, 2020.

[36] M. van Paassen and M. Mulder, "Identification of human operator control behaviour in multiple-loop tracking tasks." *IFAC Proceedings Volumes*, vol. 31, no. 26, pp. 455–460, 1998.

[37] K. van der El, D. Pool, H. Damveld *et al.*, "An empirical human controller model for preview tracking tasks." *IEEETransactions on Cybernetics*, vol. 46, no. 10, p. 2609–2621, 2015.

[38] D. McRuer and H. Jex, "A review of quasi-linear pilot models." *IEEE Transactions on Human Factors in Electronics.*, vol. HFE-8, no. 3, pp. 231–249, 1967.

[39] M. Allen, *Understanding Regression Analysis*. New York, NY, USA: Plenum Press, 1997.

[40] J. Valyon and G. Horváth, "A robust LS-SVM regression," *Transactions on Engineering, Computing and Technology*, vol. 7, pp. 148–153, 01 2005.

[41] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer, 1995.

[42] J. Suykens and J. Vandewalle, "Least squares support vector machine classifier," *Neural Processing Letters*, vol. 9, p. 293–300, 1999.

[43] H. Wang and D. Hu, "Comparison of SVM and LS-SVM for regression," in *2005 International Conference on Neural Networks and Brain*, vol. 1, 2005, pp. 279–283.

[44] A. Géron, *Hands-on machine learning with scikit-learn, keras, and tensorflow: concepts, tools, and techniques to build intelligent systems (Second)*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2019.

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

[46] D. van Poucke, "LSSVMlib," https://github.com/DannyVanpoucke/LSSVMlib, 2022.

[47] Y. Zhang, D. Kimberg, H. Coslett *et al.*, "Multivariate lesion-symptom mapping using support vector regression," *Hum. Brain Mapp.*, vol. 35, pp. 5861–5876, 2014.

[48] O. Kisi, K. Parmar, K. Soni *et al.*, "Modeling of air pollutants using least square support vector regression, multivariate adaptive regression spline, and M5 model tree models," *Air Quality, Atmosphere & Health*, vol. 10, p. 873–883, 2017.

II

Preliminary Report

graded for AE4020 Literature Study

List of Figures

List of Tables

1 Introduction

Around the world, more than 10 million people are living with Parkinson's Disease (PD)[1], a neurodegenerative disease that impacts almost everything in a patients life, from simple tasks such as drinking a cup of coffee, to more complex and abstract problems such as living a happy live. It is a progressive disease, meaning it gets worse over time, its causes are largely unknown, and as of yet no cure has been discovered. Furthermore, the degradation and symptom progression are poorly understood and hard to predict, and palliative care is highly patient specific and trial and error based. But, fortunately, research is ongoing in all these areas. Biologists and pathologists are studying PD's causes [1], physicians and pharmacists are looking for cures [2], and different types of research are being done into objectifying diagnoses, quantifying disease progression, and evaluating symptom treatments.

One of the research areas currently active is in quantifying disease progression based on the motor symptoms of Parkinson's. Some of the main debilitating factors of PD are symptoms such as bradykinesia (slowness of movement), rigidity, and a rest tremor, occurring in almost all patients [3]. These motor symptoms are some of the most obvious features of the disease, and are qualitatively well described. However, evaluating the progression of the disease and the effectiveness of treatment is currently being done using questionnaires and subjective evaluation by neurologists [4]. But so far no objective evaluation method has been developed, and there is an unmet need for more quantitative ways of evaluating these motor symptoms.
One way this problem can be solved is through a line of study currently being conducted by a collaboration between the TU Delft Faculty of Aerospace Engineering, and the Erasmus Medical Center. This line of research applies cybernetic methods to research the decline in fine-motor performance caused by neurodegenerative disorders. Cybernetics in this context is defined by Mulder et al. as "a system-theoretical, model-based approach to understand and mathematically model how humans control vehicles and devices" [5], and it is widely used in Aerospace Engineering to model the pilot of an aircraft. This modelling of humans can also be applied in other areas, as is done in this thesis.

The idea behind the research conducted in this area is that a tracking task, in which a human operator needs to track a target on a screen, can be used to measure certain cybernetic parameters. Earlier research proved that these cybernetic parameters differ significantly for PD patients when performing a tracking task [6]. Lugtenborg (2020) used this knowledge to create a 'proof-of-concept' for a diagnostic tool that will measure the tracking task control of PD patients, and from this longitudinal data identify behavioural trends in motor performance [7]. Whereas Lugtenborg looked at linear trends in the data, this thesis will also look at non-linear trend detection methods. This will be done by surveying different machine learning methods capable of detecting non-linear trends. Machine learning is a form of artificial intelligence, in which an algorithm improves automatically through experience. Lately, it has been hailed as the solution to many problems, though a lot of applications are still 'proof-of-concept'. Likewise, machine learning has seen various applications in healthcare and research into neurodegenerative disease specifically, and is seen by some as the solution to better diagnosis and treatment [8]. Also for this thesis it will be explored as a solution to finding non-linear trends, since it is accurate, can capture highly complex relationships and is easily adapted and scaled. In this thesis a selection of machine learning algorithms is tested and the best performing further improved.

[1] https://www.parkinson.org/Understanding-Parkinsons/Statistics, retrieved on 8 November 2020

This report gives the initial findings of the literature study, as well as a providing a plan for further research. It starts by outlining the research objective and research questions in the rest of this chapter. Afterwards, Chapter 2 gives an overview of Parkinson's Disease and discusses its symptoms, diagnosis, progression and treatment. It also explains the effects PD has on fine-motor performance and thus on tracking task control. Subsequently, Chapter 3 covers the basics of the tracking task used for this research, as well as a discussion on the nature of the data set and simulation of new data. After that, in Chapter 4 the field of machine learning is outlined. It gives an overview of the most important concepts, and surveys the wide range of algorithms available. Then, in Chapter 5 the selection of the candidate models is discussed, and the selected algorithms are explained. Finally, in Chapter 6 the literature study is concluded, and the steps for further research are explained.

1.1. Research objective and research questions

This research project has the following objective:

> **"To contribute to the development of quantitative ways of evaluating Parkinson's Disease progression, in particular fine-motor skills, by using nonlinear trend analysis to identify behavioural changes in tracking task control."**

Keeping the objective of this research in mind, the main research question can be stated as follows:

> **"What is the best method for detecting a change in tracking task motor behaviour due to Parkinson's disease symptoms in varying human controller data?"**

Sub-questions can be defined, arising from the main research question:

1. How can the change in a PD patients tracking task control be simulated?
   (a) How does the motor behaviour of PD patients change over time?
   (b) What data needs to be simulated, and what are the steps that need to be taken in order to simulate this change in tracking task motor behaviour?
   (c) How can the change of motor behaviour be accurately represented in the simulated data, and how can a variability be introduced?
   (d) Which bootstrapping method best preserves the dependence present in the participant data?
2. What criteria are relevant for evaluating the performance of the chosen models?
3. Do nonlinear trend detection methods perform better than linear methods when applied to this data set of simulated tracking task data?
   (a) How do linear methods perform when applied to this data set?
   (b) Which nonlinear methods are available, and which of these is most fitting for solving the problem in this research?
   (c) How does the chosen nonlinear model compare to the earlier used linear methods when applied to varying levels of motor symptom change?
   (d) Does the model have to be unique for each individual, or can a training dataset be used that combines data from multiple individuals?

2

Parkinson's Disease

In this chapter some of the aspects of Parkinson's Disease (PD) relevant to this research topic are discussed. The chapter starts with a general introduction of the disease, its symptoms and causes in Section 2.1. Then the way PD is diagnosed and treated is discussed in Section 2.2 and 2.3, respectively. Finally the relation between PD and fine-motor skills is discussed in Section 2.4, and the chapter is concluded in Section 2.5.

2.1. Introduction to Parkinson's disease

Parkinson's Disease is a brain disorder, characterised by the cell death of many of the neurons in the Substantia nigra, a small part of the midbrain [9] (see Figure 2.1.[1]) These neurons are responsible for the creation of dopamine, and their deaths lead to a shortage of this neurotransmitter. Dopamine is vital for the coordination and effective planning of movements [10], and lack of it causes the classical motor symptoms associated with the disease [1]. With more than 10 million people living with PD worldwide[2], it is the second most common neurodegenerative disorder (after Alzheimer's Disease). Yet much about the disease is unknown; we do not know what causes it, and there is no cure [2].

In many patients PD first manifests as a slight tremor in one of the hands, but other early symptoms include difficulty with walking, rigidity and bradykinesia (slowness of movement) [1]. These motor symptoms are the most well known aspect about the disease. It was in fact James Parkinson who first published a detailed description of these symptoms in his An Essay on the Shaking Palsy, back in 1817 [11]. Since then, much research has been put into describing and measuring these symptoms. As PD progresses, these motor symptoms worsen, and in the later stages of the disease a patient typically experiences postural instability, speech

[1] Modified from: `https://upload.wikimedia.org/wikipedia/commons/9/9e/Basal_ganglia_circuits.svg`, retrieved on 29 September 2020

[2] `https://www.parkinson.org/Understanding-Parkinsons/Statistics`, retrieved on 8 November 2020

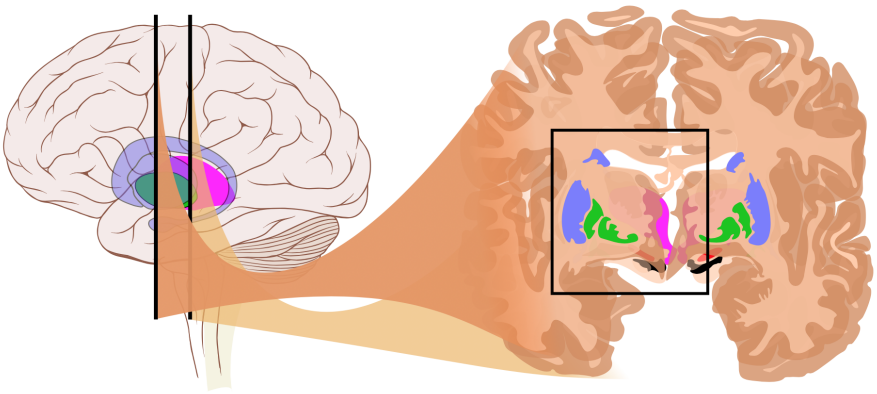


Figure 2.1: The location of the Substantia nigra within the brain, given in black.[1]

dysfunction, and difficulty swallowing [1, 4]. Unfortunately, Parkinson's Disease also comes with neuropsychiatric symptoms [3, 12]: dementia caused by PD is prevalent under patients (up to 78% [13]). This PD dementia is furthermore associated with depression, anxiety, apathy and hallucinations. Additionally, patients often develop sleep disorders, bladder dysfunction, and several sensory symptoms.

Although the exact cause of PD has yet to be determined, research indicates that a combination of environmental and genetic factors is working together to cause the disease [1]. The greatest risk factor for developing PD is age: incidence rises exponentially after the age of 60 [14]. Furthermore, more males than females suffer from the disease (approximately 3:2 [1]). Other risk factors are of environmental nature. Among others, exposure to pesticides and prior head injury increase the risk of contracting PD, while smoking and coffee consumption lower the risk [14]. Overall, many risk factors and their effects are largely unknown, and the epidemiology of PD is poorly understood.

2.2. Diagnosis

Parkinson's Disease is hard to explicitly identify: there is no test that will give a definite diagnosis. Instead, clinical criteria are used [1, 3]: both the presence of the cardinal motor symptoms (bradykinesia, tremor, rigidity, and postural instability), as well as the absence of certain symptoms that would point to a different diagnosis are used as indicators for a possible case of PD. Furthermore, the patient's response to dopaminergic treatments is evaluated [3].

One of the difficulties of diagnosing PD is that the syndrome Parkinsonism, the condition that causes many of the movement disorders seen in PD, can have other causes, such as medications (drug-induced parkinsonism), small strokes (vascular parkinsonism), or other neurodegenerative diseases that intially present in a similar way (e.g. progressive supranuclear palsy) [4]. There are a couple important differences between PD and Parkinsonism from other causes. PD is a progressive disease, meaning it gets worse over time. This is not necessarily the case for other forms of Parkinsonism. Secondly, the symptoms of PD can somewhat be treated with dopaminergic medications, while these medications have no or even detrimental effect when used for other causes[3]. When diagnosing PD, the presence of red flags suggesting other causes of the motor symptoms should be considered carefully. The US National Institute of Neurological Disorders and Stroke (NINDS) defines criteria for Possible PD, Probable PD, and Definite PD, depending on how many of the PD cardinal motor symptoms are present, and the severity of the red flags [15]. Features suggesting alternative diagnoses can be found using neuroimaging and lab tests, but unfortunately no method has been developed yet that incorporates these tools for positively diagnosing PD. The definite diagnosis for PD can only be given after the presence of biological indicators has been observed through autopsy.

In order to overcome these problems a lot of research is performed into developing new methods for diagnosing Parkinson's Disease. Many of these methods involve analysis of early motor symptoms. These will be discussed in Section 2.4. A second area of research is evaluating speech patterns. Little (2009) collected voice measurements from healthy persons and PD patients, and developed a new way to measure dysphonia (disorders of the voice), that is "robust to many uncontrollable confounding effects including noisy acoustic environments and normal, healthy variations in voice frequency" (p. 1015) [16]. Using a machine learning algorithm to classify these new measures of voice data into either coming from a healthy person or a PD patient, a correct classification performance of 91.4% was reached. Das (2010) used the data collected by Little to compare the performance of different machine learning algorithms, and achieved a 92.9% accuracy using neural networks [17]. Other studies using similar methods reached similar levels of accuracy [18, 19].

Machine learning algorithms are also used in conjunction with gait analysis. In Abdulhay (2018) [20], measurements of vertical ground reaction force during two minutes of walking on level ground are used to evaluate different parts of the gait cycle (stance time, swing time), as well as the stride period and the foot strike profile. Applying this data to a Medium Gaussian SVM machine learning algorithm, a 94% classification accuracy was reached. Other studies combining gait analysis data performed similarly [21]. Unfortunately, both speech and gait analysis methods are still very much in development, and not yet applicable in clinical practice.

[3] `https://davisphinneyfoundation.org/parkinsons-vs-parkinsonism/`, retrieved on 25 October 2020

2.3. Progression and treatment

Parkinson's Disease is progressive and chronic, meaning that the symptoms worsen over time and no cure is available. This means treatment of PD is focused on treating the symptoms, and the used treatment approach depends on the disease stage [1].

The main treatment for the PD motor symptoms is dopaminergic function enhancing drugs, such as Levodopa, dopamine agonists, and monoamine oxidase type B inhibitors. These drugs do not alter the course of the disease, and thus treatment can be started at any time. However, a trade-off has to be made between symptom relief and the side-effects these medications have [1, 22]. The Levodopa related side effects are especially relevant for this study, as these influence the motor performance of patients that take the medication.

Levodopa is an amino-acid common in humans, and it is the precursor to dopamine. Since PD patients are deficient in dopamine, which is the cause of the motor symptoms, artificially introducing Levodopa, which gets transformed into dopamine, helps to alleviate some of these symptoms [23], mainly bradykinesia and rigidity. Unfortunately, long-term treatment with the drug requires higher and higher doses, leading to a high chance (40-50% in the first five years of chronic treatment, 70-80% after 10 years [7]) of developing Levodopa-Induced Dyskinesia (LID). LID involves dyskinesia (involuntary muscle movements) and motor fluctuations [23, 24], and can cause problems in the lives of patients. Although no way has been found to administer Levodopa long-term without it resulting in LID, most patients prefer this over the PD motor symptoms [25].

Since PD is a progressive disease the treatment options are different based on the stage the disease is in. Subjective rating scales are used in order to monitor the progression and plan treatment. The most-commonly used and best tested rating scale is the Movement Disorder Society's Unified Parkinson's Disease Rating Scale (MDS-UPDRS) [26]. It considers both the symptoms and the experiences of daily living, and makes a distinction between motor and non-motor symptoms [27]. The MDS-UPDRS is divided into four parts, that are given in Table 2.1. Other scales are used as well, and the most well known of these is the Hoehn and Yahr Rating Scale [4]. It is much simpler than the MDS-UPDRS, but due to that its use is more limited. It does not cover all motor-symptoms, and none of the non-motor symptoms [28].

Table 2.1: The four parts of the MDS-UPDRS, with an example question topic for each of the parts. Part 1 & 2 are filled in by the patient, whereas part 3 & 4 are filled in by the examiner [29].

Part	Topic	Example question topic
I	Non-motor experiences of daily living	Depressed mood
II	Motor experiences of daily living	Eating tasks
III	Motor examination	Gait
IV	Motor complications	Time spent with dyskinesia

In general, there is no best rating scale, it depends on the patients and their circumstances. However, there are a few points one has to keep in mind. Rating scales are subjective, and the outcome depends on the specific patient, but also on the examiner. Not only that, the time a rating scale interview is done also matters. Patients experience good and bad days, and symptoms can even vary over the course of the day. Furthermore, depending on the medication given to the patient LID might be present. These limitations indicate the need for more objective methods [30]. Research is being done into human behaviour coding in video recordings [26], wearing sensors to evaluate motor performance [31, 32], and even using smartphones to evaluate various aspects of PD [30, 33]. Unfortunately, this research is still in the early stages, and there are still many challenges that have to solved before home-based monitoring systems can become commonplace.

2.4. The relation between Parkinson's disease and fine-motor performance

For this study, the main point of attention is PD's influence on fine motor skills. Parkinson's disease affects the substantia negra in the basal ganglia of the brain (see Figure 2.1). These areas are involved in the process of cognitive-motor decisions, which involve decision making and voluntary movement generation [34]. This can be seen by the orange lines in Figure 2.2 [34]. Damage in this area of the brain causes bradykinesia [35], difficulty in executing sequential movements [36], and decreased ability to coordinate and synchronise movements [37], all of which decrease the patients ability to perform fine motor functions.

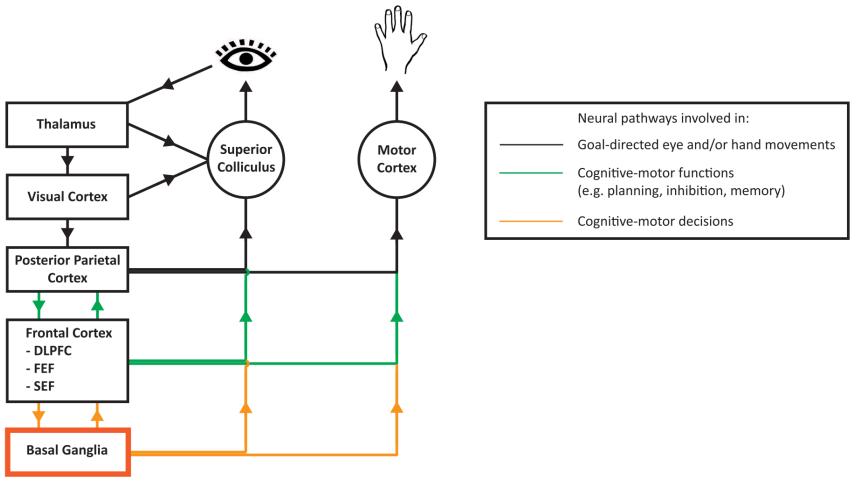


Figure 2.2: Schematic depiction of the neural pathways involved in eye-hand movement generation [34]

A lot of research has been done on this decrease in ability. Flowers (1978) found that PD patients have difficulties in their control during continuous movements, and for simple multisine signal tracking tasks perform slower than healthy persons of the same age [38, 39]. Furthermore, research found that PD patients are deficient in the strategy of movement; it seems they have trouble predictively controlling their movements due to a lack of a dynamic 'internal model' [40]. This reduction in prediction capabilities was also observed by Jones [41]. The effect of dopaminergic medications (such as Levodopa) on the fine-motor skills of PD patients varies. Some problems, such as the tendency to overshoot a target, are reduced when using medication (the damping ratio is increased) [42]. Furthermore, it was found that the natural frequency of movements tends to become more similar to healthy persons. In contrary, many patients limit the range (amplitude) of their movements; this is not changed when using medication [43].

De Boer (2015) evaluated fine-motor performance using "precise measurements of eye- and hand parameters during visuomotor coordination tasks with varying cognitive load" (p. 150) [34]. In this research it was found that the movements of early-stage Parkinson's patients were slower when reaching for targets. In order to further investigate the dynamics of fine-motor performance in patients suffering from neurodegenerative disorders, a collaborative research project was set up between the department of Neuroscience of the Erasmus Medical Center and the faculty of Aerospace Engineering of Delft University of Technology. This research focuses on using cybernetics to quantify the decline in fine-motor performance caused by neurodegenerative disorders. De Vries (2016), building on the work of Flowers (1978), Jones (1989), and Hufschmidt (1995) [40–42], used a pursuit tracking task to show that PD patients have a lower control gain, but increased damping and higher control variability [6]. Lugtenborg (2020) developed a trend analysis method for quantifying the decline in motor performance for PD patients [7]. Using a general linear regression model she was able to successfully detect trends in tracking task data. However, nonlinear trends were not explored in this work, and they might provide a better fit. That is where this research comes in.

2.5. Conclusions

This chapter gave an overview of Parkinson's Disease. It has become clear that PD is a highly individual disease; every patient experiences it differently, and the symptoms differ wildly. There is no comprehensive and objective clinical method for diagnosing the disease, nor for monitoring its progression, though clearly there is a need for such a method. This research will further develop one approach for solving this problem. Earlier research proved that a cybernetic tracking task can be used for measuring the impact PD has on the fine-motor skills of patients. Subsequently, a way to detect linear trends in longitudinal patient data was developed. However, the most optimal way for detecting trends (and thus tracking progression) is not yet known. This research will develop a way of detecting nonlinear trends in the data, and compares this against the methods developed earlier. State-of-the-art in detecting nonlinear trends is by making use of machine learning techniques [8], and those will be used in this research as well.

3

Tracking tasks and the data set

In this chapter the problem at hand for this research will be analysed, by looking at the tracking task and data set that will be used. It starts with a description of the tracking task in Section 3.1. Due to the Covid-19 pandemic gathering data from real patients was undesired, and therefore a procedure for simulating artificial data is explained in Section 3.2. Then, in Section 3.3 a preliminary data simulation is discussed, and the chapter is concluded in Section 3.4 with a summary of the problem that will be solved using machine learning.

3.1. Tracking task for evaluating fine-motor control

As discussed in Chapter 2, tracking tasks can be used to track fine-motor performance, and earlier research has been done into this area. De Vries (2016) and Lugtenborg (2020) defined a particular tracking task, that is capable of detecting changes due to PD motor symptoms, and this task will also be used in this research [6, 7]. A condensed overview of the task, defined using control theory principles, will be given in this section, and a more elaborate view can be found in the earlier research.

3.1.1. Types of control

The control behaviour of a human controller (HC) differs depending on the task. In their 1960 paper, Krendel and McRuer described three different control classifications: [44]

- With Compensatory control the only input for the HC is the error signal between their output and the target signal.
- For Pursuit control the HC gets the actual signals of the output and target as input, and can use that to derive the error themselves.
- Finally, with Precognitive control the HC knows the target and necessary input completely.

Different displays are used for the different control classifications, from which the HC gathers their information. Figure 3.1a gives an example of a compensatory display and Figure 3.1b of a pursuit display [45]. Figure 3.1c shows a preview display [45]. This is a more advanced display for pursuit control, where apart from the output and target also the future path of the target signal is given. Regarding precognitive control, this is only possible when the target signal is super predictable, and this is thus independent of the type of display used. Earlier work determined that for this research, where the focus lies on elderly people and PD patients, a pursuit display would be most optimal [6, 7].

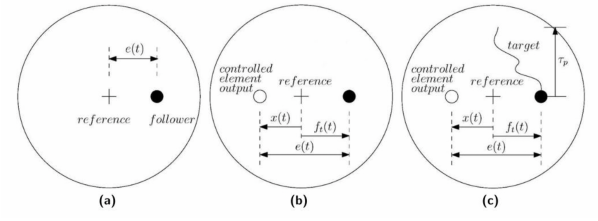


Figure 3.1: An example of a compensatory (a), pursuit (b), and preview display (c) [45]. A preview display attempts to provide the HC with as much information as possible by providing the future path of the target signal.

3.1.2. Block diagram

The block diagram corresponding to a pursuit tracking task is given in Figure 3.2 [45]. The HC is represented by controller, and the element that needs to be controlled by H_{ce}, which is acted upon by the control signal ($u(t)$) from the controller and the external disturbance ($f_d(t)$). The controller takes three information sources as input, the controlled element output ($y(t)$), the target signal ($f_t(t)$), and the error signal ($e(t)$), as well as the as well as the nonlinear remnant, $n(t)$, present when modelling humans.

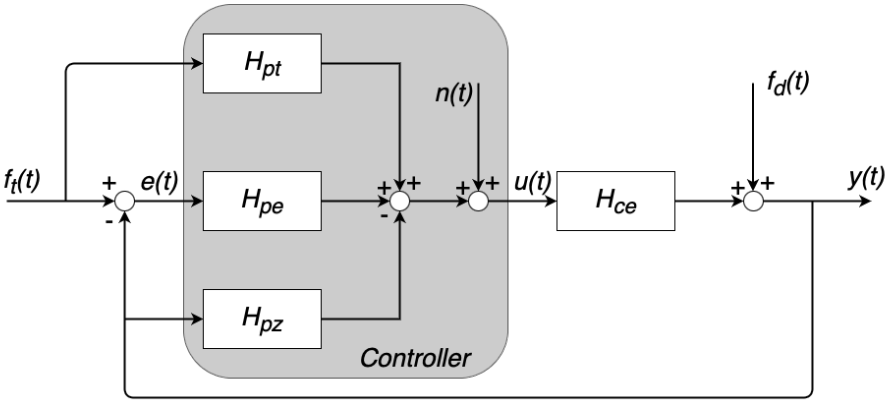


Figure 3.2: Block diagram of a pursuit tracking task, adapted from [45].

The system given in Figure 3.2 is overdetermined, meaning that it can be simplified. Often this is done by removing the $y(t)$ feedback [7], meaning that the HC only uses the error and target signal. A further simplification can be made by looking at the controlled element. It is defined by single integrator dynamics, as given in Equation 3.1, where its gain, K_c, is taken to be equal to 1.

$$H_{ce} = \frac{K_c}{s} \tag{3.1}$$

When controlling single integrator dynamics in a pursuit task, the HC does not use the target input function [46], and the model can thus be reduced to what is effectively a compensatory control task, as shown in Figure 3.3 [45], where the controlled element dynamics are as given in Equation 3.1. The disturbance forcing function has become redundant due to the single-input-single-output nature of the system [7], and the controller dynamics and target forcing function will be described next.

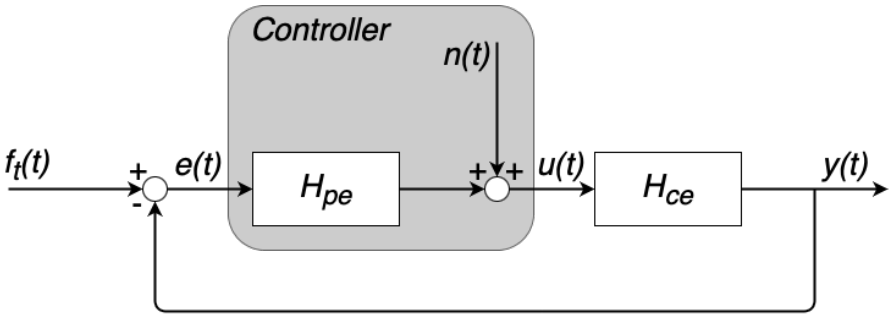


Figure 3.3: Block diagram of a compensatory tracking task, adapted from [45].

3.1.3. Controller

The controller represents the dynamics of the human operator. The dynamics follow from the extended crossover model, defined by McRuer and Hex (1967) as a combination of a linear model and a nonlinear remnant [47]. The linear part consists of the dynamics of the neuromuscular system, a pure delay term, and a pilot equalisation model, taken to be a pure gain term for a single-integrator control task. It is given in Equation 3.2, . The contribution of the nonlinear remnant can be neglected by using a properly designed forcing function, with a high signal-to-noise ratio (SNR) at the excitation frequencies [48]. Such a forcing function is defined in the next section.

$$H_{pe}(s) = K_p e^{-s\tau} \underbrace{\frac{\omega_{nms}^2}{s^2 + 2\zeta_{nms}\omega_{nms}s + \omega_{nms}^2}}_{\text{Neuromuscular dynamics}} \tag{3.2}$$

3.1.4. Forcing function

The forcing function takes the form of a quasi-random multisine signal, with k different sines, as given in Equation 3.3. N_f is the number of sines, and each sine has amplitude A_f, frequency ω_f, and phase ϕ_f.

$$f_t(t) = \sum_{k=1}^{N_f} A_{f_k} \sin\left(\omega_{f_k} t + \phi_{f_k}\right) \tag{3.3}$$

The frequencies are obtained by multiplying the base frequency, ω_m, by a prime integer, n_f, as in Equation 3.4, where T_m is the measuring time (taken by De Vries as 40.96 seconds [6]).

$$\omega_f = n_f * \omega_m = n_f * \frac{2\pi}{T_m} \qquad n_f = \{4,7,13,19,29,37,43,53,79,109,157\} \tag{3.4}$$

The amplitudes come from Equation 3.5, scaled by Equation 3.6 in order to set the standard deviation to 1.

$$A_{f_k} = \left| \frac{\left(1 + 0.1 j\omega_{f_k}\right)^2}{\left(1 + 0.8 j\omega_{f_k}\right)^2} \right| \tag{3.5}$$

$$A_{f_k} := A_{f_k} * \sqrt{\frac{1}{\sigma_{A_f}}} \tag{3.6}$$

Finally, the phase is taken at random with a couple constraints defined by De Vries [6], and given in Equation 3.7.

$$\phi_f = \{7.239, 0.506, 7.860, 8.184, 9.012, 6.141, 6.776, 6.265, 4.432, 2.672, 8.009\} \tag{3.7}$$

3.2. Data set

The second part of this chapter concerns the data set that will be used in this research. In order to train and validate a machine learning model a sufficiently large data set is needed; the more data the more accurate the model. Secondly, as will be explained in Section 4.2, models trained using a small data set are more prone to over-fitting. However, due to current circumstances[1], it is undesirable to gather new data from human test subjects. Therefore another solution has to be found. Here, previous research can help. Data gathered by Lugtenborg (2020) [7], will be used as a reference for creating an artificial dataset of both healthy persons, as well as PD patients. This dataset will subsequently be used to train the machine learning models.

3.2.1. Measured data

The basis of the data used for this research is the data gathered by testing human subjects in earlier research [7]. This data takes the form of panel data: different observations for different entities at different points in time [49]. The data set consists of trial data of 25 healthy participants. Each participant completed 50 trials, with each trial consisting of performing the tracking task explained in Section 3.1. Performing the tracking tasks works as depicted in Figure 3.4 [7]: the target (black circle) moves across the screen, and the participant is tasked with reducing the error between the target and the system output (blue dot), by moving the blue dot with their finger, as depicted in Figure 3.5 [7]. Each experiment is repeated for the other hand.

From the experiments the time trace of the control and output signals are recorded. An example of the output signal is given in Figure 3.6, where it is shown together with the target signal [7]. From this, and the known time trace of the forcing function, system identification techniques can be used to solve for the unknown controller parameters in the human controller model described in subsection 3.1.3: K_p, τ, ζ_{nms}, and ω_{nms} [47]. These parameters are the footprint of a human's control performance, and De Vries (2016) showed that there is a significant difference in their values when comparing healthy subjects to PD patients,

[1] This research is being carried out during the height of the Covid-19 pandemic.

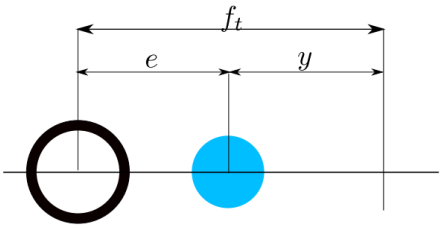


Figure 3.4: A schematic overview of the tracking task performed by the participants. [7]

Figure 3.5: The experiment set-up. Participants manipulate the blue dot via a touchscreen. [7]

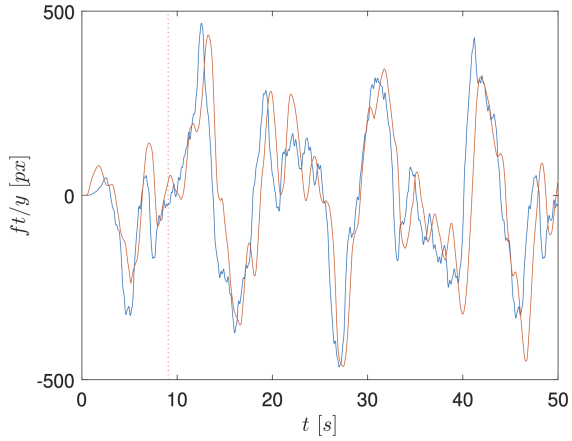


Figure 3.6: An example of the time trace of the target (blue) and output signal (red). [7]

as explained in Chapter 2 [6]. An example of the values of each of the parameters can be found in Figure 3.10 [7].

3.2.2. Improving the data set

As explained earlier the data is obtained from healthy participants and therefore no significant trend is expected in the data, as there is no decline in fine-motor skills (just day-to-day variations). In order to progress with this research a method has to be setup to simulate this decline.

This simulation will be done in two steps. First, 25 additional data points are constructed using bootstrapping. With bootstrapping one draws random samples with replacement from the data [50]. Unfortunately, one of the core assumptions of standard bootstrapping methods is that the data is independent and identically distributed (i.i.d.). With time series data this is not the case, as each data point depends on previous data points. Drawing random samples from the data ignores this dependency. Therefore, another method has been proposed: the block bootstrap. This method, first introduced by Künsch (1989) [51], divides the data into overlapping block of length l, determined by Equation 3.8, where N is the length of the time series. The block consists of l entries of the time series, in order, with the first block starting at the first entry of the time series, the second block at the second entry, etc. To prevent the first few entries of the time series from occuring less than the others, the data is wrapped around in a circle, meaning the last block consists of the entries $\{y_n, y_0, y_1, ..., y_{l-1}\}$. Instead of sampling single entries entire blocks are sampled, preserving the dependent structure within the separate blocks.

$$l = \lceil N^{1/3} \rceil \tag{3.8}$$

According to Künsch, observations are nearly uncorrelated when they are far enough separated in time, indicating the blocks can be sampled at random. Much research is still being done into bootstrapping time series, and one problem persists: the block bootstrap tends to make the dependency present in the time series weaker [52]. In order to overcome this problem, Politis and Romano (1994) proposed the stationary

Table 3.1: Range of parameter changes for simulated PD symptoms [7]

	Mild Symptoms	Severe Symptoms
K_p	$-0.5 < \delta < 0$	$-0.9 < \delta < 0$
τ	$0 < \delta < 0.02$	$0 < \delta < 0.07$
ζ_{nms}	$0 < \delta < 0.2$	$0 < \delta < 0.31$
ω_{nms}	$0 < \delta < 3$	$0 < \delta < 5$

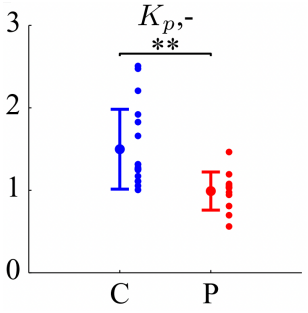


Figure 3.7: Comparison of K_p values for healthy people and PD patients. [6]

bootstrap [53]. Here, sampling is still done in blocks, but the block length is no longer fixed. Instead it is a random variable L, with its probability distribution equal to the geometric distribution with parameter p, as given in Equation 3.9. In order to select p the desired average block length, λ can be used, since $\lambda = p^{-1}$. λ itself can be determined in a similar way to l in the block bootstrap, by using Equation 3.8.

$$P(L = j) = (1-p)^{j-1} p \tag{3.9}$$

The second step in simulating the required trend in the data is artificially inducing a decline in motor skills by manipulating the constructed parameters in such a way that it simulates the decline of motor skills in PD patients. This is done by adding a δ to each of the bootstrapped data points. This δ is selected at random from a predefined range, that can be found in Table 3.1. The range is constructed by Lugtenborg (2020) [7], based on experiments with PD patients carried out by De Vries (2016) [6]. An example of the results of these experiments can be found in Figure 3.7. δ is constructed subjectively, with mild and severe symptoms corresponding to the average and extreme difference in values, respectively. The random drawing of 25 different δ results in a variation in symptom severity per trial, for each specific case. This variation accounts for the real life differences in control of PD patients on good and bad days, as explained in Chapter 2 [1]. In the ideal case the trend introduced into the data set in this way matches the real life trend that exists within PD patient data.

3.3. Data simulation

As explained in the previous section, not only will the data measured by Lugtenborg be used as input for the trend analysis, additional data points will be simulated using bootstrapping [7]. Simulating data allows for increasing the size of the dataset, but another benefit is that the data set is 'clean', meaning the prediction models can be more accurately tested and assessed. In this section the simulation of the data through a Python script will be explained, and the initial results will be shown and compared to the measured data.

3.3.1. Performance parameters

First two new performance parameters will be introduced [7]. These performance parameters give an indication of how well the HC in controlling the controlled element. The performance parameters are the tracking performance, RMS_e, given by Equation 3.10 and the control activity, RMS_u, given by Equation 3.11. In both equations, RMS() indicates taking the root mean squared value of the signal. The former gives an indication of how well the test subject is performing the control task, and the latter parameter indicates the extend to which the participant is actively controlling the system; a high value indicates high control activity.

$$\text{RMS}_e = \text{RMS}(e)/\text{RMS}(f_t) \tag{3.10}$$

$$\text{RMS}_u = \text{RMS}(e)/\text{RMS}(u) \tag{3.11}$$

The value of the performance parameters is determined by running the control system depicted in Figure 3.2, and obtaining its time response. The different signals, $e(t)$, $u(t)$, and $y(t)$ come from running the system. The system is run in Python, with as input the forcing function described in subsection 3.1.4. In order to run the system the delay term has to be linearised, and this is done using a third-order Padé approximation [54], as given in Equation 3.12.

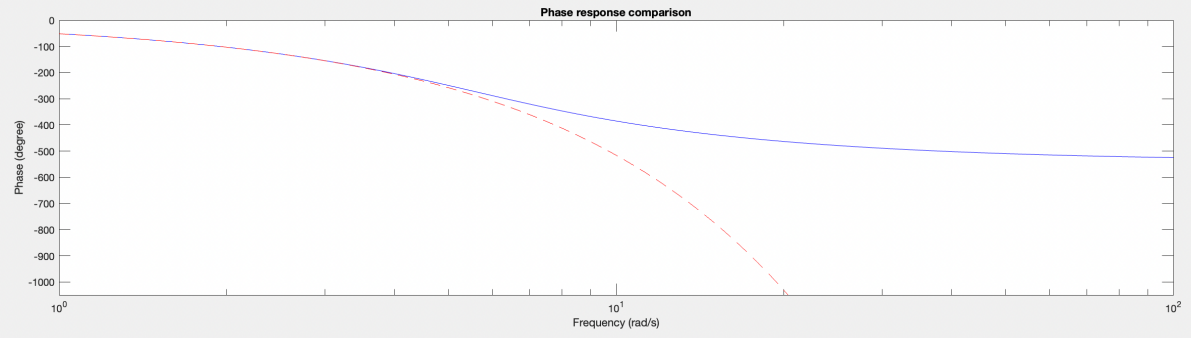


Figure 3.8: Phase response of the third order Padé approximation of $e^{-0.9s}$. The blue line is the Padé approximation of the pure delay, given by the red line. As can be seen, the approximation starts to diverge at around x = 7, where x denotes the frequency, w [rad/s].

$$e^{-s\tau} \approx \frac{-\tau^3 s^3 + 12\tau^2 s^2 - 60\tau s + 120}{\tau^3 s^3 + 12\tau^2 s^2 + 60\tau s + 120} \tag{3.12}$$

Through trial and error it was found that a third-order approximation was sufficient. At the extreme values of τ ($\tau = 0.9$ s) it matches the phase response of a pure delay, as can be seen in Figure 3.8. Beyond $w \approx 7$ rad/s it starts to diverge significantly, however, this is also where the magnitude of the response starts the decrease, lessening the impact the diversion has on the final simulation result.

3.3.2. Preliminary data simulation

In order to prepare for further research a preliminary data simulation script was developed to test the simulation process. Random samples are drawn from a normal distribution, with mean and standard deviation taken from the data of Lugtenborg [7], for each of the four human controller parameters. Then, the 25 additional data points are obtained through block bootstrapping, and the delta is introduced to simulate PD symptoms. The RMS_e and RMS_u where simulated through the process described in the previous section. The results are given in Figure 3.9, and the comparative results of Lugtenborg's research are given in Figure 3.10 [7]. In both figures the vertical line gives the separation between "healthy" trials, and trials where PD symptoms were simulated. From a visual inspection of the results it can be concluded that the simulated data is comparable. The major difference is in the RMS_e and RMS_u, indicating that in future research some settings of the control system simulation will have to be checked. The data simulation results will be used in Chapter 5 to test the preliminary implementation of the chosen machine learning algorithms, and in further research to create a clean data set upon which the various algorithms can be tested.

3.4. Conclusions

This chapter explained the nature of the problem that will be solved in the rest of this thesis. It started by explaining the tracking task that was used in earlier research for evaluating PD patients. The results were analysed and through system identification techniques 6 control parameters were identified: K_p, τ, ζ_{nms}, ω_{nms}, RMS_e, and RMS_u [7]. But unfortunately this data was gathered using healthy subjects, and due to the Covid-19 pandemic no additional testing can be done with humans. Therefore, a method was setup to simulate the tracking task data, which was artificially altered to reflect values corresponding to those of PD patients.

The data that will be used in the next steps takes the form of panel data: 6 parameters, for 25 participants, in 75 trials. The parameters (or features as they will be called later) are dependent upon each other; they result from the same measured time traces of the tracking tasks. In earlier research trend analysis on this data was performed using linear methods, and each metric evaluated separately [7]. However, non-linear trends might be a better fit, and taking all metrics as input might give more accurate results. Finding non-linear trends in the data set of multiple dependent features will be the topic of the rest of this research.

For solving this problem machine learning techniques will be used. Machine learning is highly suitable to solving these problems: it is most likely to provide accurate predictions (when compared to ordinary statistical methods), it is highly adaptable and scalable, and it allows for capturing more complex relationships [55]. This makes it the ideal technique for application in this research.

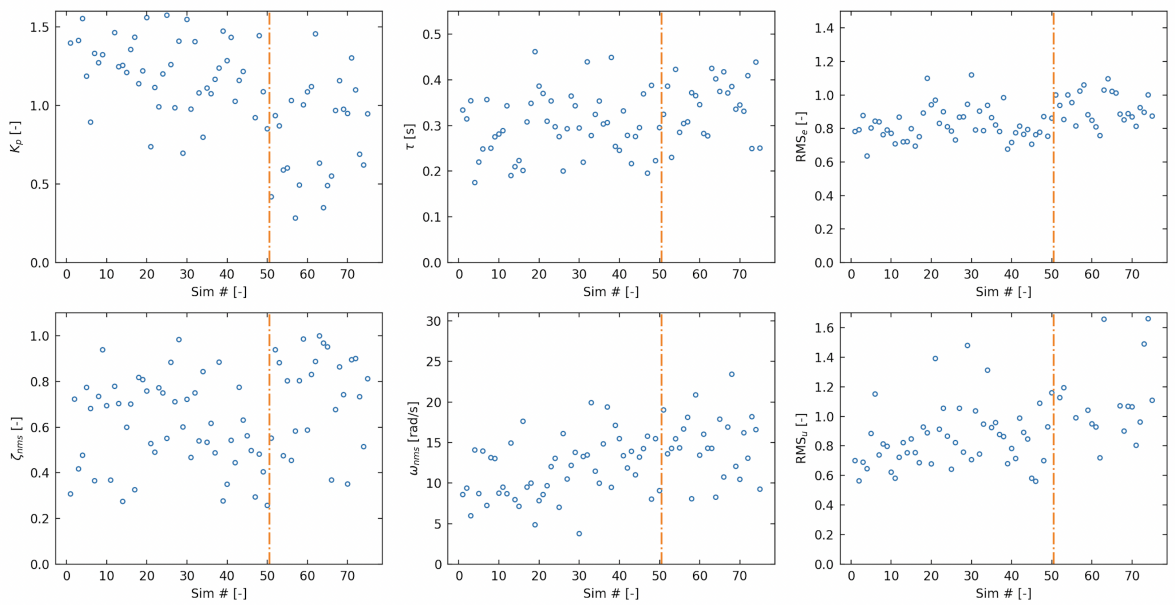


Figure 3.9: Results of the preliminary data simulation. The dotted line indicates the separation between the "normal" values, and the values where artificial PD is introduced. The parameters (μ, σ) for the normal distribution are taken from Lugtenborg (Table VIII, p. 19) [7], and are the following: K$_p$ = (1.25, 0.25), τ = (0.29, 0.07), ζ_{nms} = (0.564, 0.22), ω_{nms} = (11.9, 3.69). These are the values for Participant 25, D.

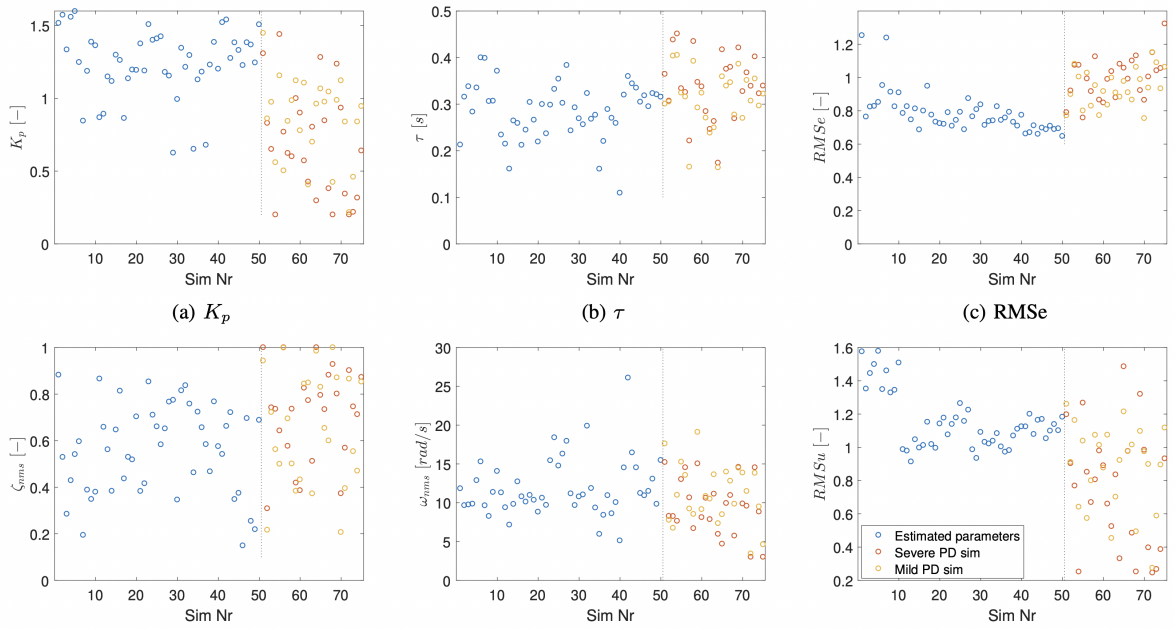


Figure 3.10: Results for Participant 25, D of Lugtenborg's research [7].

4

Machine Learning

Machine learning is the field of computer science in which computers use large amounts of data to independently learn to perform tasks, without being programmed how to do them. Since this research involves applying machine learning techniques to quantifying the cybernetics of Parkinson's Disease a quick overview of the field is given in this chapter. First, a general introduction to machine learning is given in Section 4.1, and data processing is discussed in Section 4.2. Following this introduction into the field a closer look is taken at the various algorithms that can be considered for this research. In Section 4.3 a number of these are explained (linear regression, linear regression based, decision trees, support vector machines, and neural networks), and relevant examples from literature are given. Finally, the chapter is concluded in Section 4.4.

4.1. Introduction to Machine Learning

Machine learning algorithms are algorithms that improve automatically through experience. A large set of data is used to train the algorithm to deliver a particular output. Once the algorithm is sufficiently trained, it can then be applied to new data, processing input separate from the training data. This is called generalisation; successful generalisation is one of the central goals of a machine learning problem [55].

4.1.1. The machine learning process

A machine learning model is a model that is setup in such a way that it can develop its own algorithm for solving a problem [56]. Machine learning has many applications, from computer vision [57] and voice recognition [58], to, relevant for this research, medical diagnosis [8]. There are many different aspects to consider when solving a problem using machine learning, and the general process looks as given in Figure 4.1[1].

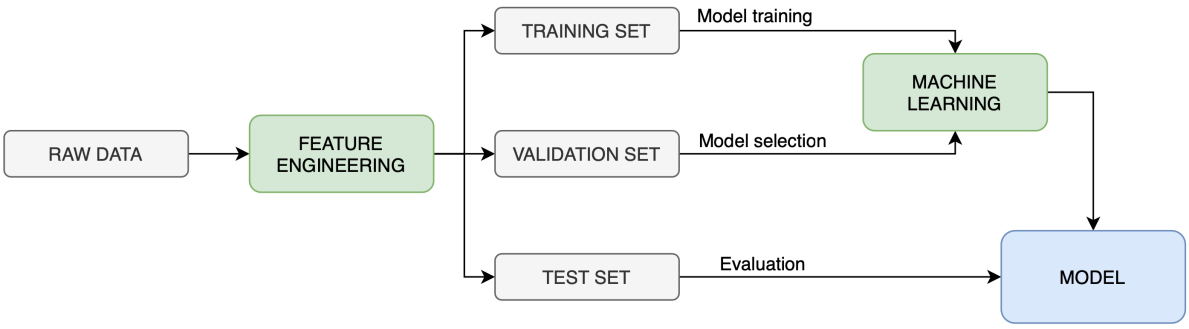


Figure 4.1: The process of creating a machine learning model[1]

It starts by converting the raw data into three different data sets, through a process called feature engineering. These three data sets each have a different purpose[2]. The training set is the actual data set that is used to train

[1] Modified from: `https://techblog.cdiscount.com/a-brief-overview-of-automatic-machine-learning-solutions-automl`, retrieved on 3 November 2020

[2] `https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7`, retrieved on 3 November 2020

the algorithm and determine its parameters. However, the internal parameters of the algorithm are not the only parameters that have to be determined. Each algorithm comes with a set of hyperparameters, these are the different settings of the algorithm, used to alter its learning process [59]. An example of this is the amount of nodes in a neural network. The validation set is used to tune these hyperparameters, through a process called model selection. Once both the internal- and hyperparameters of the algorithm are determined the algorithm is said to be "trained". However, before one can use the model to make predictions it has to be independently evaluated to make sure it is working properly. That is where the test set is used. This latter set is completely new to the model, and often carefully curated in order to best reflect the real world.

4.1.2. Types of machine learning

A lot of different machine learning algorithms exist, that each have different properties and serve different purposes [60]. Selection of the right algorithm depends on quite a few factors: what is the problem that needs to be solved, how much data can be gathered and what types, how much computer power is available, etc. Especially the first factor makes quite a difference; within the field of machine learning there are several different approaches, and each approach can be used to solve different problems. The first step in algorithm selection is therefore to determine the machine learning approach that is most applicable to solving the problem. The different approaches are [55]:

- **Supervised learning** works with labelled data; the training set consists of input vectors, along with their corresponding target vectors (the labels). The model is trained to categorise the input with a certain target vector. Within supervised learning a distinction can be made between regression and classification. For the former the output is a continuous quantity, while for the latter the output takes on a discrete set of labels.
- Opposite of supervised learning is **unsupervised learning**. The training data only consists of the input vectors, without any corresponding target values. The goal of algorithm is to take the inputs and generate structure in the data.
- In between the two approaches above lies **semi-supervised learning**, where some data is labelled and some isn't. This form of learning is useful when a large data set needs an initial labelling to cluster around, but labelling everything would be too resource intensive.
- Finally, there is the technique of **reinforcement learning**. Here the algorithm is not given optimal target vectors, but must discover them itself, through maximising a reward corresponding to how well the algorithm solved the problem. This can for example be used to train an algorithm how to play chess.

The goal of this research is to develop a way to detect nonlinear trends in longitudinal patient data. Regression algorithms are the algorithms that are used for these type of problems [55], and these will therefore be the ones considered in the rest of this research.

4.1.3. Over- and underfitting

One of the major complications to watch out for in any machine learning model is the concept of overfitting. Overfitting happens when the model fits the training data too well. The noise, inherent in any data, has been captured by the model structure, and the model is therefore no longer applicable to new data. An example of this can be found in Figure 4.2[3]. There, the green line represents an overfitted model, while the black line represents a usable model. As can clearly be seen, when using the green model to make predictions they will be way to specific, and thus often incorrect for other samples. The danger for overfitting is largest when the data set is small, especially when the number of features is close to, or even surpasses, the number of observations. Luckily, most advanced algorithms will have build-in methods to deal with overfitting.

On the other side of overfitting is the problem of underfitting. This happens when one chooses a model that is to simplistic to fully capture the complexity of the data (i.e. choosing a linear model to represent a polynomial relation). Underfitting is easier to detect than overfitting (using good performance metrics), and therefore less of a problem.

4.2. Data processing and features

The machine learning process starts with data collection. Lots of data is needed in order to properly train and validate the learning algorithm. The different data attributes fed into the machine learning model are referred to as features. One only wants a certain selection of features to be fed into the model, but often there are a lot

[3] Modified from: `https://en.wikipedia.org/wiki/Overfitting`, retrieved on 3 December 2020

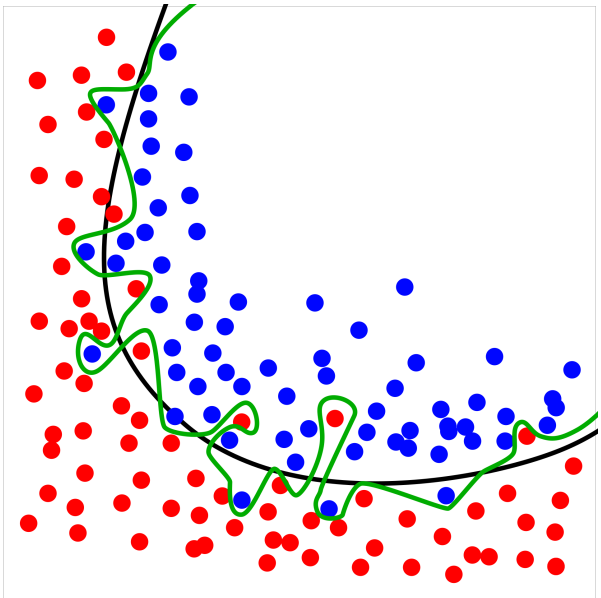


Figure 4.2: Example of a overfitted (classification) model[3]. The algorithm needs to determine a good split into the red and the blue class (as given by the black line), however, it overfits (green line).

of features present, that might or might not be relevant to the problem. That's where feature engineering and feature learning come in; selecting the right features is crucial for creating a good performing model.

Feature engineering is the process of determining the features that will be used in the machine learning model. This can either be done by selecting a subset of features that are most relevant to the model (feature selection), or by deriving new features from an initial set of measured data (feature extraction). For feature selection, three main approaches are defined [61], in addition to manually picking the features. The three main approaches are:

- **Wrapper methods** take a subset of the features, train a model using these features, and evaluate the model performance using separate validation data set. Wrapper methods usually succeed in selecting the most optimal set of features, however because the process needs to be performed for many different feature subsets they are very computationally expensive. Methods that use the wrapper approach are for example Genetic Algorithms [62].
- **Filter methods** are similar to wrapper methods. They, too, test subsets of features, but instead of evaluating them using the model, a proxy method is used. An example is Welch's t-test, which can be used to investigate the effect of selecting different features on the target variable [63].
- **Embedded methods** are feature selection methods that are inherent to a model. By constructing the model one automatically performs feature selection. An example of this is Random Forests[1].

In contrast to feature selection is the concept of feature extraction. With feature extraction an initial set of measured data is taken, and features are constructed by combining the data in different ways. Often feature extraction is performed in order to come up with a smaller set of new features that better describe the data. For feature extraction too a split can be made between manual and algorithmic feature extraction. Manual feature extraction is relatively simple, but does require knowledge in the field to do properly [64]. Examples of manual feature extraction are derivatives, feature scaling, and statistical functionals [65]. On the other hand, using algorithms for feature extraction is more complex, but often performs better. Examples of such algorithms are Principal Component Analysis, where linear combinations are combined to maximise the variance, and Linear Discriminant Analysis, which ranks these combinations of features on the basis of their separability[4].

Finally there is the concept of feature learning. Here the feature selection is done completely by the machine learning algorithm, and the only input is the raw data [66]. An example of this are Deep Learning algorithms.

Though performing feature engineering is of vital importance in creating any good working machine learning

[4] `https://elitedatascience.com/dimensionality-reduction-algorithms`, retrieved on 9 November 2020

model, it is beyond the scope of this study to go into further detail here. The selection of features relevant to this research will be discussed later in this report.

4.3. Selecting and evaluating ML algorithms

The most important choice in developing a machine learning model is the selection of the algorithm. In order to make this choice a few aspects have to be considered. These are outlined in this section. Furthermore, two performance metrics that can score the performance of regression algorithms are explained.

4.3.1. General selection factors

There are a few factors that are important to every machine learning algorithm selection. The first of these is the **type of problem** one wants to solve. This has been discussed in section 4.1, and fundamentally limits the range of algorithms to choose from (supervised/unsupervised, classification/regression, etc.).

Secondly, there is the **data set**. Things like size, structure, whether or not there is missing data, the ratio between the number of observations and the number of features all suggest different algorithms. For example, when there is a limited amount of training data, or if the amount of features is higher than the amount of observations, one should choose algorithms with low variance and/or high bias, often the more simpler algorithms (i.e. Linear Regression) [55]. Furthermore, domain knowledge can play a role here; especially in the way the selected algorithm can match an expected outcome. It would make sense, for example, to choose an algorithm capable of finding nonlinear trends when such trends are expected, though it can still be beneficial to first try linear algorithms, as they are often way easier to set up and interpret.

This last point touches on another consideration: **interpretability** [67]. In order words, how easy it is to understand the algorithm and how it comes to its conclusions. This relates to how well researchers can trust the prediction. A distinction is made between white-box, grey-box, and black-box [68]. The former is an algorithm that is easy to describe and understand, an example is linear regression, where one can easily see how the different features relate to the outcome through their weights. On the other side of the spectrum are black-box algorithms, where an input leads to an output, but the process of getting from one to the other is not understood. An example of this is Deep Neural Networks. In-between the two extremes lie grey-box models, where the process can be understood to some degree.

Interpretability is often traded off with **accuracy** (the degree to which the prediction matches the real world) [67]. For most machine learning problems, the more interpretable a model, the less accurate it is. The reasoning behind this is that highly interpretable models are less flexible, and conform less well to the data, whereas models such as deep neural networks can approximate even the most complex structures.

Other considerations all have to do with the implementation of the algorithm. **Training time** (time to learn the model) and **prediction time** (time to make a prediction based on input) are factors that should be taken into account, and depend on the purpose of the machine learning model. These factors depend on the algorithm selected, but are also highly influenced by the amount of data, and the power of the system on which the model is run.

Finally there is a subjective metric: **Ease of Use**. This includes aspects as how easy it is to code, how much information is available, how much research has been done into the various intricacies of the algorithm, and whether special data preparation is necessary.

4.3.2. Performance metrics

In order to determine how accurate the model is in making predictions a performance metric can be used. For regression there are two often used metrics. First of these is the Root Mean Square (RMS) error, which gives an indication of how much error the model makes in its predictions. It is determined using Equation 4.1 [69], where m is the number of instances in the dataset, $\mathbf{x}_i$ the i$^{\text{th}}$ feature vector in matrix $\mathbf{X}$, which contains all feature values in the dataset. Finally, h is the prediction function of the model.

$$\text{RMS}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(\mathbf{x}_i) - y_i)^2} \tag{4.1}$$

A second performance metric that can be used is this Mean Absolute Error (MAE). Where RMSE punishes larger errors more than minor errors, MAE punishes errors linearly, meaning it is more interpretable and less sensitive to outliers. The MAE is determined using Equation 4.2 [69].

$$\text{MAE}(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m}|h(\mathbf{x}_i) - y_i| \tag{4.2}$$

4.4. A survey of ML Algorithms

Many different ML algorithms exist, and each works in a different way and has its own strengths and weaknesses. As stated in Section 4.1, for this research a supervised regression algorithm will be used, of which an overview of the most popular is given in this section. Special attention is paid to those algorithms used in medical literature, and where relevant examples will be noted and commented upon.

4.4.1. Linear regression

Linear regression is the simplest and fastest form of regression. It works by fitting a straight line through the data, which consists of a scalar response, and a scalar or vector of explanatory variables. Even though linear regression is a simple method, it can still be quite powerful, and it often gives a good baseline for further analysis. This can for example been seen in [70], where it is compared against more advanced algorithms. Interestingly it was found that simple linear regression performed as well as more advanced linear regression algorithms when applied to nonlinear data, suggesting that for providing a baseline for nonlinear analysis the former is preferred. Linear regression has two main disadvantages: it is unusable for fitting nonlinear trends, and it is prone to overfitting and multicollinearity. The former can only be overcome by choosing different algorithms, but to deal with the latter disadvantage extensions of the simple algorithm have been developed, all performing regularisation (artificially penalising model coefficients). Some of these are:

- **Least-angle regression (LARS)**. LARS starts with no variables in the model, and step-wise adds the variables with the most explanatory power (least-angle to the residual) [71].
- **Least Absolute Shrinkage and Selection Operator (LASSO)**. LASSO works by adding a penalty term to the regression cost function, equal to the magnitude of the coefficients. This allows for coefficients to go to zero, removing them from the model [55] (de facto performing feature selection). LASSO is sometimes combined with least-angle regression.
- **Ridge Regression** is developed to deal with multicollinearity. Ridge regression is linear regression with an added penalty term to the cost function, equal to the square of the magnitude of the coefficients. It lowers their value, but does not push them to zero. This is useful when all coefficients are of importance [55].
- The final extension that will be discussed is **Elastic Net**, which combines LASSO and Ridge Regression. The balance between the two has to be carefully tuned, but if the right combination is found it removes the problems with either algorithm.

In a comparison study on linear regressors done by Bayestehtashk et al (2013) it was found that ridge regression was the best performing regularisation technique for voice data [72], and research by Lugtenborg (2020) successfully a general linear regression model to tracking task data [7].

4.4.2. Linear regression based algorithms

There are several algorithms that are in essence just linear regression, but get treated as separate algorithms. The first of these is polynomial regression. Instead of fitting a straight line, a nth order polynomial is fitted to the data, using least squares methods. The different x^n are treated as features, and therefore this method is in fact another form of linear regression. The main disadvantage with polynomial regression is that it is very prone to overfitting, and therefore, unless one is very sure of a polynomial relationship, will only work within the domain of the test data. Another algorithm of this family is **Multivariate adaptive regression spline**. With this algorithm, instead of fitting a single linear relation to the data, hinge functions are used to break the regression up into several linear parts, that together can model nonlinear relations. Both polynomial regression and multivariate adaptive regression spline are seldom used in the area of prediction. Finally, there is **Theil-Sen regression**, Developed to deal with variables that have a non-constant variance, Theil-Sen regression fits a straight line by taking the median of slope of all lines through pairs of points [73]. Theil-Sen regression is compared to other linear and nonlinear algorithms by Salmanpour (2020), where it performed quite well in predicting the UPDRS score in year 4 based on certain parameters in year 0 and 1 [74]. Unfortunately, Theil-Sen regression is only capable of fitting linear relations.

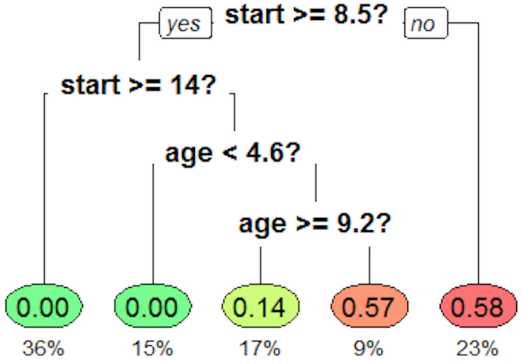


Figure 4.3: An example of a decision tree[5]

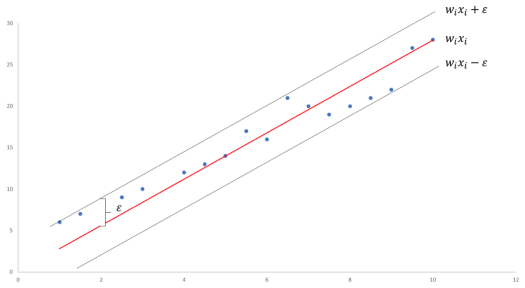


Figure 4.4: Illustrative example of SVR. The boundary is defined by the allowed error ϵ, and the slack is given as ζ[7]

4.4.3. Decision trees

A second form of linear regression is to make use of decision trees. These are a set of hierarchical decisions which eventually give a final result [75]. An example of this can be seen in Figure 4.3[5]. The main advantage of tree based models is that they are highly interpretable, it is easy to understand how a set of decisions leads to the outcome. There are also major disadvantages however; the model is highly sensitive to the training data (meaning that when trained on different data an entirely different tree will result), and the split made at each level is a hard split, leading to functions with discontinuities at the split boundaries. Proper construction of the three model based on the data set is a difficult problem to solve by hand, but luckily many different algorithms have been developed. Examples of these are called ID3 and C4.5, but the most used one in regression analysis is called CART (Classification and Regression Trees) is most often used in regression analysis [76]. CART uses binary trees, where each node leads to only two branches. The example in Figure 4.3 is of a CART model. CART has been developed by Breiman et al. in 1984 [75], and has subsequently been used in many different studies to solve regression problems. For example, Tsanas et al. (2010) [70] compared the algorithm to linear regression in predicting UPDRS score based on voice data, and found that it performed satisfactory. CART is also widely applied in other fields, such as for prediction of panel data in precipitation forecasting [77].

Decision trees are easy to interpret, and perform well with nonlinear problems. However, single trees have a high tendency for overfitting, because they memorise the training data. This is especially the case with noisy data. As with linear regression, extensions have been developed to deal with this problem. These methods are so called ensemble methods, as they combine multiple trees, and come in two variants:

- **Random Forest** is an example of a bootstrap aggregation (bagging) method [78]. Subsets of the data are created using bootstrapping methods (see section 3.2). Decision trees are trained on each of the subsets, and their predictions are averaged to arrive at a single prediction. Bagging methods handle high-dimensional data very well, and are good at dealing with missing data. However, because the final prediction is an average they are less accurate.
- The second method is boosting. Here the learners (trees) are learned sequentially. Early learners are trained on the data and analysed for errors. Subsequent trees are fitted to take the error into account, with the goal of improving the accuracy. Examples of boosting methods are **AdaBoost** [79] and **XGBoost** [80]. Boosting methods are more accurate than bagging, but they do require careful tuning of hyperparameters.

4.4.4. Support Vector Regression

Support Vector Machines are algorithms originally developed for classification problems, but they can be applied to regression as well (SVR). The most simple form of SVR finds the best fitting straight line, similar to linear regression, but in SVR this line is called a hyperplane[6]. Contrary to linear regression, SVR does not seek to minimise the errors, instead allowing a certain degree of error to be present in the model. SVR tries

[5] Modified from: `https://upload.wikimedia.org/wikipedia/commons/2/25/Cart_tree_kyphosis.png`, retrieved on 2 December 2020

[6] Formally, a hyperplane is defined as "a subspace whose dimension is one less than that of its ambient space", from: `https://en.wikipedia.org/wiki/Hyperplane`, retrieved on 3 December 2020

to fit the line in such a way that all points fall within the margin of error, minimising the slack between the boundary line and any outlying points. The SVR hyperplane is visualised in Figure 4.4[7].

Not all models can be represented by a linear hyperplane. But luckily SVR can still be applied. By first mapping the features to a higher space, where linear SVR is possible, then define a hyperplane in that space, and subsequently map this hyperplane back into the original space, obtaining a linear decision boundary is possible. This is where so called kernel tricks come in. The kernel function is specific to each feature mapping, and represents an efficient way to perform the earlier explained process[8]. There are three main kernels used, and one generally tries the least complex first, before moving on to more complex kernels.

- **Linear kernel**: as the name suggests, a linear kernel fits a straight line through the data. It is the most simple form of SVR, but often performs worse than simple linear regression. The linear kernel cannot be used when the relation between the data is not linear.
- The **Polynomial kernel** is used for mapping the data to a higher order polynomial.
- Finally, the **Radial Basis Function (RBF) kernel** is used to map the data into infinite dimensions. It is difficult to visualise how this works, but the result is the RBF works on the principle of similarity: a point gets rated based on the points closest to it.

The main advantage of SVR is that it is a really fast way to capture and predict highly non-linear phenomena, but it is also very susceptible to outliers. In literature, support vector machines are most often used for classification tasks, but nonetheless some good results were achieved in regression. For example, Eskidere et al (2012) [81] compared two SVR and two neural network algorithms against each other in predicting UPDRS data from speech measurements, and found that 'Least-Squares Support Vector Regression' (LS-SVR) [82] provided the lowest prediction errors.

4.4.5. Neural Networks

Neural networks are the most popular and most well known type of machine learning algorithm, and are modelled after the brain [55]. A neural network consists of different types of nodes, and each node receives inputs from other nodes. These inputs are weighted, and all inputs are summed together in the node, and serve as input for the node activation function ($f()$ in Figure 4.5). This activation function can be thought of as kind of mathematical gate between the input and the output of the node. This gate can take on different kinds of forms, from a simple step function turning the output on or off to a complex transformation mapping the input into the correct output signals. The learning aspect of a neural network is in the weights, which determine how strong the connection is between the various nodes, and thus what the final output will be. The nodes are put together in a network, with an input and output layer, and one or more hidden layers [83]. This is illustrated in Figure 4.6[9].

There are a lot of different types of neural network architectures, and the most used ones (looking only at regression problems) will be outlined here.

- A **Feedforward** neural network is the simplest form of a neural network. In fact, the network given in Figure 4.6 is an example of such a network. In a feedforward neural network information only moves in one, 'forward', direction and there is no feedback present. Simple feedforward neural networks are easy to setup and train, and are capable of dealing with lots of noise in a data set.
  - A special type of feedforward neural network is the **Radial Basis Function network**. Here, the hidden layer activation functions consist of radial basis functions [55]. This means that the input is weighted based on its distance to a certain centre. This centre needs to be determined through training.
  - The simplest way of training a neural network is through random variation of the weights. Unfortunately, since there is no structure this is an incredibly inefficient way. That is where **backpropagation** comes in [83]. Backpropagation is the most used algorithm for training a feedforward neural network, and it's significantly more efficient. The algorithm works by determining how much each output node needs to change its output for the right answer to come out. It then backpropagates this error through the network, adjusting the required weights as necessary.
- **Recurrent Neural Networks** (RNN's) are more advanced algorithms. In RNN's, the sequential input is processed one element at a time. From each step information is retained for the next step through a so

[7] Modified from: `https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2`, retrieved on 3 December 2020

[8] Specifically, the kernel function is a very efficient way to compute a dot-product in a feature space, possibly of a very high dimension.

[9] Modified from: `https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/`, retrieved on 3 December 2020

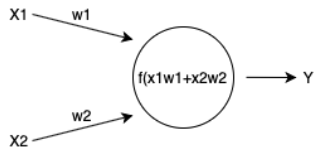


Figure 4.5: Schematic depiction of a single node in a neural network

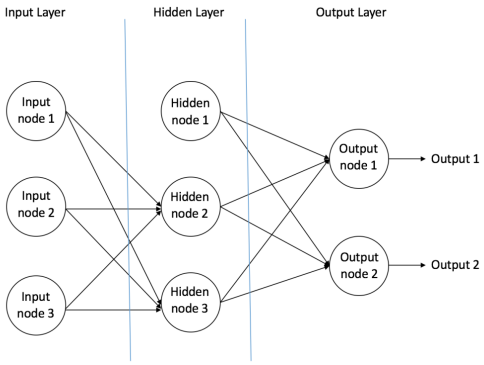


Figure 4.6: Example of a simple neural network architecture[9]. Not all nodes are connected; where there is no connection the weight factor is equal to zero.

called hidden state, and this information gets combined with a new input into a new hidden state. At the end of the chain is the final output of the model. The output is thus not only based on the current input, but also on all previous inputs. The main advantage of RNN's is that they often perform superior to standard feedforward neural networks when working with sequential data [83], for example when predicting time series. An schematic depiction of a single neuron of an RNN is given in Figure 4.7[10], which illustrates how the RNN retains information from each step.
Unfortunately, RNN's are hard to train; they suffer from the vanishing gradient problem, meaning that the backpropagated error, used for training, decays over time [84]. This leads to RNN's having short term memory: the earlier layers hardly 'learn'. Two specific forms of RNN's have been developed to deal with this short term memory problem: Long Short Term Memory (LSTM) Networks, and Gated Recurrent Unit (GRU) Networks. Both work with gates; internal mechanisms that combined allow the networks to store information for a longer time.

- The first of these is the **LSTM**, first developed by Hochreiter and Schmidhuber (1997) [85]. The basis of an LSTM cell is the LSTM state, which can be stored for a longer time, thus forming a memory cell of sorts. Through the gates information gets added or removed from the cell state. An LSTM contains three types of gates: forget gate, which decides what information from the previous hidden state to keep and what to forget, the input gate, which combines the result of the forget gate and the input into the new cell state, and the output gate, which decides what the new hidden state will be. The LSTM cell and its gates are illustrated in Figure 4.8.[11]
- The second form of RNN is **GRU**, developed by Cho et al. (2014) [86]. It simpler than LSTM, containing only two gates and no cell state. Instead, the previous hidden state gets modified directly through the reset gate, which takes the input and previous hidden state, and determines how much information to retain from the previous hidden state, resulting in a candidate hidden state. This candidate state then passes through the update gate, which decides what the final hidden state will be. An example of a GRU cell is given in Figure 4.9.[12]

- **Convolutional Neural Networks** (CNN's) are the final variant of neural networks relevant to this problem. Though more often associated with image recognition, CNN's can also be used for time series analysis [87]. CNN's are neural networks with one or more convolutional layers. These are similar to normal layers, in that they have an activation function that transforms the input into the output. However, in convolutional layers the transformation is a convolutional operation. This means that one or more filters are applied to the input, increasing certain features and decreasing others. The specifics of the filter are determined when training the CNN. By using this method of filtering, the model is able to learn filters that 'search' the input for specific patterns. One advantage of CNN's is that they are quite efficient in training and predicting [55]. An example of a typical CNN architecture can be found in Figure 4.10.[13]

[10] https://upload.wikimedia.org/wikipedia/commons/b/b5/Recurrent_neural_network_unfold.svg, retrieved 30 April 2021
[11] https://en.wikipedia.org/wiki/File:Long_Short-Term_Memory.svg, retrieved 30 April 2021
[12] https://en.wikipedia.org/wiki/File:Gated_Recurrent_Unit.svg, retrieved 30 April 2021
[13] https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png, retrieved 3 April 2021

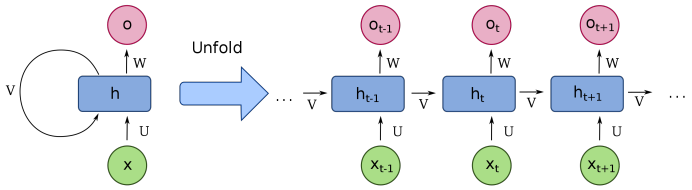


Figure 4.7: Overview of a recurrent neural network recurrent neural network, which shows how information is retained throughout the time steps.[10]

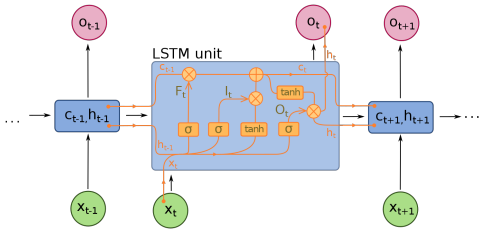


Figure 4.8: An LSTM cell. The gates are indicated by their first letter (F for forget gate, etc.).[11]

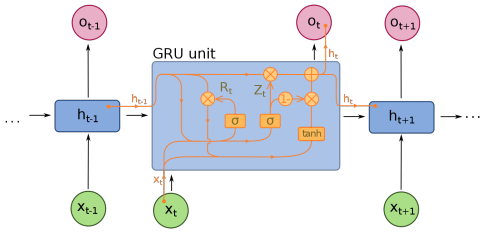


Figure 4.9: A GRU cell. The reset gate is indicated by R_t, and the update gate by Z_t.[12]

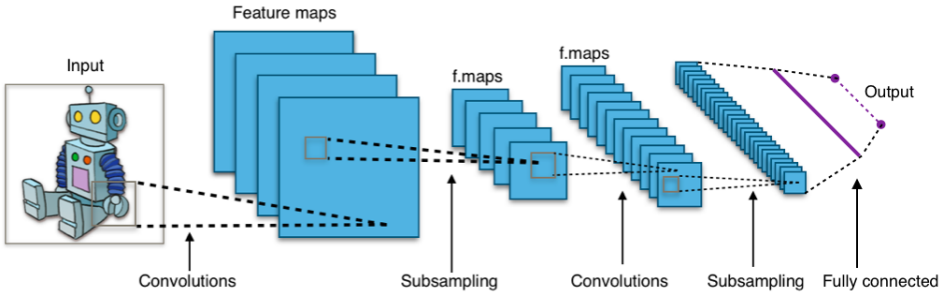


Figure 4.10: An example of a typical CNN architecture.[13] In the image it can be seen how each step is filtered and only part of the input is carried over to the next layer.

Neural networks in relevant literature

Due to their popularity, there are a lot of examples of applications of neural networks within relevant literature. Eskidere et al (2012) used a standard multilayer feedforward neural network, as well a a variant of the radial basis function network to predict UPDRS score from speech measurements [81]. Though in this particular case other algorithms performed better (LS-SVR), the neural networks still achieved sufficient accuracies. Salmanpour et al. (2020) compared 11 different algorithms for predicting the UPDRS score in year 4 based on certain parameters in year 0 and 1 [74], among which a radial basis function neural network, a normal feedforward neural network, and a recurrent neural network. Of the three, RNN performed best. Recently the focus has started to shift from RNN's to CNN's for time series analysis: Borovykh et al. (2018) use CNN's to forecast time series based on financial data, and thereby argue that CNN's perform better than RNN's when forecasting time series with limited data [88], and Hewamalage et al. agree with this in their 2021 overview paper on using RNN's for time series forecasting [89]. Nonetheless, RNN's are still widely used in time series prediction, especially in neurology: Wang et al. (2018) used an RNN for predicting the progression of Alzheimer's Disease [90], and Che et al. (2017) used an RNN for evaluating the similarities between longitudinal patient data [91].

4.5. Conclusions and selected algorithms

This chapter gave an introduction to machine learning. It outlined the general process and explained some concepts that are important for every machine learning problem. It has become clear that for this research a supervised regression approach is most suitable, and therefore an overview of the most popular algorithms

for regression analysis (linear regression, decision trees, support vector regression, and neural networks) has been provided. Machine learning is an incredibly wide field, and many more algorithms exist. Examples of these are Gaussian Process Regression and K-nearest Neighbours Regression. However, compared to the described algorithms in this chapter they are sparsely used, and therefore not considered for this research. Of the four groups of algorithms, only the latter three (Decision Trees, Support Vector Regression, and Neural Networks) support nonlinear regression. These are summarised in Table 4.1, along with a few key points that help decide which algorithms will be used in further research.

From each of the three groups an exemplar algorithm will be chosen for use in further analysis. Starting with decision trees, by far the most used algorithm in literature is CART [75]. Compared to other tree based algorithms it is simple, less susceptible to outliers, and it can use the same variables different times in separate parts of the tree, allowing for uncovering complex interdependencies [92]. For these reasons it is the decision tree algorithm that will be used in this research. Secondly, for support vector regression there is the specific algorithm, Least-Squares Support Vector Regression [82], used in some literature. As it is simpler than normal support vector regression and performs similarly [93], it is chosen for future research. As kernel the RBF Kernel will be used, as it is the most versatile [94]. Finally, for neural network regression there are a bit more options that can be chosen from. However, in most research RNN's are suggested for use with time series data, as they have been especially designed for that purpose [69]. Within RNN the GRU Network is the considered the new state-of-the-art, and it is simpler compared to LSTM [91]. It is therefore the final algorithm selected for further research.

Table 4.1: Summary of the different algorithms considered for this research

Group	**Algorithm**	**Comments**
Decision trees	CART	- Can handle missing values - Simple - Less susceptible to outliers - Can use the same variables different times in separate parts of the tree
	ID3	- Can only handle categorical data
	C4.5	- Can't handle missing values - More susceptible to outliers - Can use the same variables different times in separate parts of the tree
Support vector regression	SVR	- Standard version of SVR
	LS-SVR	- Improved version of SVR - Uses linear instead of quadratic programming - Easier to understand and implement
	Linear kernel	- Can only do linear relationships
	Polynomial kernel	- Can only do polynomial relationships
	RBF kernel	- Can do complex non-linear relations - Gives good results with an unknown relation between variables
Neural networks	Feedforward	- Difficulty with capturing sequential information in the input data, and therefore less suitable for dealing with sequential data
	RBF Network	- Difficulty with capturing sequential information in the input data, and therefore less suitable for dealing with sequential data
	RNN - LSTM	- Designed for handling sequential data
	RNN - GRU	- Designed for handling sequential data - Simpler architecture compared to LSTM, leading to shorter computational time
	CNN	- Designed for handling spatial data

5

Candidate algorithms

Thus far this report outlined the background of the research that will be conducted, explained the problem that will be solved using machine learning, and gave a high level overview of the field of machine learning, thereby selecting three algorithms that will be used in the rest of the research. Each of these algorithms will be implemented, and their performance will be evaluated in order to select the best one for the problem at hand. In this chapter the three candidate algorithms will be explained: CART in Section 5.1, LS-SVR in Section 5.2, and GRU Network in Section 5.3. Along with this explanation, a preliminary version of each algorithm will be implemented. The objective of this is two-fold. First, a preliminary implementation can show that well performing regression is possible using the algorithm, and second, by applying the models to a simple dataset an initial trade-off can be made, whereby the algorithm most likely to be successful in the context of this research can be identified. This will be done in Section 5.4.

5.1. Candidate algorithm - CART

The first algorithm that will be explained is CART Decision Trees [75]. CART is a specific algorithm that builds a decision tree based on a series of Boolean decisions: whether or not to split a node into two daughter nodes. Every tree starts with a root node, in which all variables are present. By evaluating every possible split for all the variables present at that node, the algorithm decides on which variable it is best to split. It then creates two daughter nodes, and each of the daughter nodes is itself split into two daughter nodes, and so on. If a node is not split further it is referred to as a terminal node. Each terminal node has a corresponding value, which, when you follow the tree from the top down, is the value that is predicted for this input.

An example of a CART Decision Tree is given in Figure 5.1. Every node, τ, is either given by a circle for intermediate nodes (1, 2, 3, and 7), or by a square for terminal nodes (4, 5, 6, 8, and 9), and each terminal node has a prediction value, $Y(\tau)$. In decision trees nodes have ancestors and/or descendants. For example, in Figure 5.1 all nodes are descendants of node 1 (the root node of the tree). Furthermore, node 6, 7, 8, and 9 are all descendants of node 3, but not of node 2. Ancestry goes the other way around: node 2 and 1 are ancestors of node 4, but node 3 isn't.

In order to grow a decision tree the following elements have to be determined [75]:

1. How to assign a predictor value to each terminal node
2. How to split each intermediate node into two daughter nodes
3. How to decide whether or not a node is terminal

The rest of this section will explain how this is done.

5.1.1. Resubstitution estimate

In order to answer these questions first the concept of resubstitution estimate has to be introduced, which is an estimate of the absolute prediction error of the fitted model. It is determined by taking the mean of the squared residuals when using the fitted model to predict each of the known output values from the dataset, as given in Equation 5.1 [95]. Here, R^{re} is the resubstitution estimate of the predictor function, $\hat{\mu}$, and y_i and $\hat{y}_i$ are the known and predicted value belonging to input $\mathbf{x}_i$ ($\hat{y}_i = \hat{\mu}(\mathbf{x}_i)$), respectively.

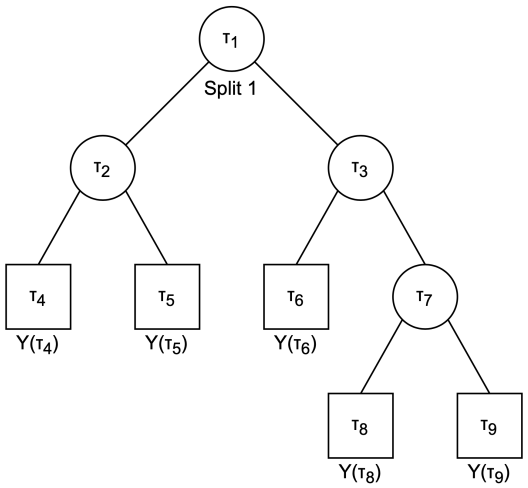


Figure 5.1: An example of a CART tree

$$R^{re}(\hat{\mu}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5.1}$$

5.1.2. Assigning a predictor value to terminal nodes

The first element that will be treated is assigning a value to a terminal node. For every prediction we want the resubstitution estimate to be as small as possible, thus the value, $Y(\tau)$, of any node τ needs to be taken in such a way that $R^{re}(\hat{\mu})$ is minimised. Breiman proved that "The value of $Y(\tau)$ that minimises $R^{re}(\hat{\mu})$ is the average y for all cases $(\boldsymbol{x}_i,\ y_i)$ falling into τ" ([75], p. 230). In other words, the value of a node is taken as the average value of the training cases, for which following the structure of the decision tree leads to that particular terminal node. It is given by Equation 5.2, where $n(\tau)$ is the total number of cases in τ.

$$Y(\tau) = \frac{1}{n}\sum_{\mathbf{x}_i \in \tau}^{n} y_i \tag{5.2}$$

5.1.3. Redefining the resubstitution estimate

Using the rule for assigning a predictor value to each node, Breiman introduces a new way for determining the resubstitution estimate, better reflecting the tree structure [75]. Instead of looking at the entire predictor function, $\hat{\mu}$, we now look at each node individually, and sum the individual resubstitution estimates. This can be done by taking the insight from the previous section, namely the assigned predictor value for the nodes. We then get a resubstitution estimate at a node, given by Equation 5.3, and the tree resubstitution estimate is subsequently given by Equation 5.4, with T indicating the set of nodes that together form the tree.

$$R^{re}(\tau) = \frac{1}{n}\sum_{\mathbf{x}_i \in \tau}^{n}(y_i - Y(\tau))^2 \tag{5.3}$$

$$R^{re}(T) = \sum_{\tau \in T} R^{re}(\tau) \tag{5.4}$$

One further redefinition can be made, by taking the (biased) sample variance of all training values falling within a node, as seen in Equation 5.5, where s^2 denotes the sample variance.

$$s^2(\tau) = \frac{1}{n}\sum_{\mathbf{x}_i \in \tau}^{n}(y_i - Y(\tau))^2 \tag{5.5}$$

When combining that with the proportion of total observations falling within node τ, $p(\tau) = \frac{n(\tau)}{n}$, the resubstitution estimate of a single node can be rewritten as in Equation 5.6.

$$R^{re}(\tau) = p(\tau)s^2(\tau) \tag{5.6}$$

5.1.4. Splitting strategy

This new resubstitution estimate will be used to determine the splitting strategy. According to Breiman, the best split is the split that provides the biggest reduction in the value of $R^{re}(T)$ [75]. If we define a reduction in $R^{re}(\tau)$ due to a split into τ_L and τ_R as in Equation 5.7, the best split at a node τ is then the node that maximises the reduction, $\Delta R^{re}(\tau)$.

$$\Delta R^{re}(\tau) = R^{re}(\tau) - R^{re}(\tau_L) - R^{re}(\tau_R) \tag{5.7}$$

Finding τ_L and τ_R as to maximise $\Delta R^{re}(\tau)$ is the same as minimising $R^{re}(\tau_L)+R^{re}(\tau_R)$, which, per Equation 5.5 is equal to finding the minimal value of their weighted variance, as given in Equation 5.8 [95].

$$\min_{\tau_L,\tau_R} \left\{ p(\tau_L)s^2(\tau_L) + p(\tau_R)s^2(\tau_R) \right\} \tag{5.8}$$

5.1.5. Determining which nodes will be terminal

The only question that still needs to answered is how we can determine which nodes will be terminal, and which ones aren't. One option would be to grow a tree until no more splits can be made. However, as Breiman observed [75] (p. 61), there is an optimal tree size: trees that are too large or too small lead to high errors, so a middle ground has to be found. He proposes a method that doesn't involve selecting the tree size up front. Instead, a tree is grown that is much too large, and subsequently pruned in the "right way". Doing this results into many different subtrees, from which the right sized subtree is chosen as the final tree.

5.1.6. Pruning

The pruning process explained in this section was developed by Breiman [75], and it starts by growing a large tree, T_{max}. This is done by letting the splitting process continue as long as possible, typically until each node contains less than five observations, but sometimes even further. This T_{max} is then selectively pruned upwards, which produces a sequence of increasingly smaller subtrees, eventually collapsing to T_0, which consists solely of the root node. However, even for a small sized T_{max} many subtrees exist, meaning a "selective" pruning procedure is necessary, where the subtrees continuously decrease in size, and each subtree is the "best" in it's size range.

In order to determine the best subtree the error-complexity measure will be introduced. It starts by defining a complexity parameter, $\alpha \geq 0$. The error-complexity measure, $R_\alpha(T)$, is then given by Equation 5.9. The first term represents the cost of the tree, and the second term its complexity, where α can be thought of as a cost penalty for complexity, and $|\tilde{T}|$ is the number of terminal nodes in the subtree T (5 in the example of Figure 5.1).

$$R_\alpha(T) = R^{re}(\tau) + \alpha|\tilde{T}| \tag{5.9}$$

For each α we then choose the subtree $T(\alpha)$ of T_{max} that minimises $R_\alpha(T)$. If α is small the penalty for having a large number of terminal nodes is small, and the minimising subtree, $T(\alpha)$, will be large, and if α is sufficiently large $T(\alpha)$ will consist of the root node only. Even tough α is continuous, there is only a finite number of subtrees. Thus what happens is that if $T_1 = T(\alpha_1)$ is the minimising tree for a given value of α, then it continues being the minimising tree as α increases, until a certain jump point ($\alpha = \alpha_2$) is reached, upon which $T_2 = T(\alpha_2)$ becomes the minimising tree, until the next jump point. This process produces a finite sequence of subtrees T_1, T_2, T_3, ..., T_{max}.

Now the only problem left to solve is how to determine $T(\alpha)$, as evaluating every possible subtree to find the minimiser of $R_\alpha(T)$ is computationally expensive. Luckily, a more efficient process has been developed. This process starts by constructing T_1 from T_{max}. T_1 is the subtree belonging to $\alpha_1 = 0$. Constructing T_1 is done as follows:

1. Take any two terminal nodes in T_{max}, τ_L and τ_R, resulting from a split of their immediate ancestor node τ.
2. If $R^{re}(\tau) = R^{re}(\tau_L) + R^{re}(\tau_R)$, prune off τ_L and τ_R.
3. Repeat until no more pruning is possible. The resulting tree is T_1.

Once T_1 has been found we can continue with the rest of the algorithm. Let τ be any nonterminal node of T_1, and T_τ the subtree whose root node is τ, with $\tilde{T}_\tau$ its set of terminal nodes. Then:

1. Starting with T_1, define a function $g(\tau)$ as in Equation 5.10.

$$g(\tau) = \begin{cases} \frac{R^{re}(\tau) - R^{re}(T_\tau)}{|\tilde{T}_\tau| - 1} & , \tau \notin \tilde{T}_1 \\ +\infty & , \tau \in \tilde{T}_1 \end{cases} \tag{5.10}$$

2. Then define the weakest link node, $\bar{\tau}_1$, in T_1, as the node such that:

$$g_1(\bar{\tau}_1) = \min_{\tau \in T_1} g(\tau)$$

   When α increases, $\bar{\tau}_1$ is the first node for which its error-complexity measure, $R_\alpha(\tau_1)$, is equal to the error-complexity measure of its subtree, $R_\alpha(T_\tau)$. This means that the node without its descendants becomes preferable to the subtree below it.
3. The value of α at which this occurs becomes α_2, and this is given by $\alpha_2 = g_1(\bar{\tau}_1)$.
4. Prune away the descendants of $\bar{\tau}_1$ such that $\bar{\tau}_1$ becomes a terminal node. We now have T_2. Now we can find the weakest link in T_2 instead of T_1, using the same process. If there is a tie for which node is the weakest link, all nodes in the tie become new terminal nodes by pruning their descendants.

The result of the process described in this section is a sequence of subtrees of decreasing size:

$$T_{max} = T_0 \rightarrow T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow \cdots \rightarrow T_M$$

and a corresponding increasing sequence of complexity parameters:

$$0 = \alpha_1 < \alpha_2 < \alpha_3 < \cdots < \alpha_M$$

5.1.7. Selecting the best pruned subtree

The result of the previous subsection is a selection of subtrees, T_k of different sizes, with $T_k = T(\alpha_k)$. But which one is the right size, and thus the "best"? For that we depend upon getting a good estimate of the prediction error, but unfortunately the resubstitution estimate that is used earlier is biased, and will select the largest tree [75]. Therefore a new, "honest", estimate, $R(T_k)$, of the prediction error is needed, and the best subtree can be defined as the subtree with the lowest $R(T_k)$. This is done using V-fold cross-validation (CV/V).

In CV/V the entire set of samples, $\mathscr{L}$, is divided into (typically 5 or 10) auxiliary subsamples ($\mathscr{L}_1$, $\mathscr{L}_2$, …, $\mathscr{L}_V$), such that these subsamples $\mathscr{L}_v$ are of approximately the same size. Then let the vth learning sample be $\mathscr{L}^{(v)} = \mathscr{L} - \mathscr{L}_v$, and its corresponding test sample is $\mathscr{T}^{(v)} = \mathscr{L}_v$. Repeat the tree growing and pruning procedure using each $\mathscr{L}^(v)$. This results in the trees $T^{(v)}(\alpha)$, which are the minimal error-complexity trees for parameter value α. Then also perform the tree growing procedure using all of $\mathscr{L}$, giving the sequences $\{T_k\}$ and $\{\alpha_k\}$.

In order to select the best pruned tree from $\{T_k\}$, the V-fold CV estimate of the prediction error is introduced, given by Equation 5.11.

$$R^{CV/V}(T_k) = \frac{1}{n} \sum_{v=1}^{V} \sum_{(\mathbf{x}_i, y_i) \in \mathscr{T}^{(v)}}^{n} (y_i - \hat{\mu}_k^{(v)}(\mathbf{x}_i))^2 \tag{5.11}$$

The estimate is obtained letting the prediction function, $\hat{\mu}_k^{(v)}$, associated with subtree $T^{(v)}(\alpha'_k)$, predict an outcome corresponding to input $\mathbf{x}_i$ in $\mathscr{T}^{(v)}$, where α'_k is given by Equation 5.12. What shouldn't be forgotten here is that although α is continuous, $T^{(v)}(\alpha'_k)$ is equal to $T_k^{(v)}$ for $\alpha_k < \alpha < \alpha_{k+1}$. The prediction outcome for each $\mathbf{x}_i$ is then compared to the known output y_i.

$$\alpha'_k = \sqrt{\alpha_k \alpha_{k+1}} \tag{5.12}$$

To determine which subtree, T_k, is the right size, first $R^{CV/V}(T_k)$ is calculated for every k. The optimal subtree, T_*, is then chosen as the smallest subtree for which Equation 5.13 holds, where SE denotes the standard deviation for $R^{CV/V}(T_k)$.

$$R^{CV/V}(T_*) \leq \min_k R^{CV/V}(T_k) + \mathrm{SE}(R^{CV/V}(T_k)) \tag{5.13}$$

The chosen subtree is considered the best, and will subsequently be used to make predictions on new data.

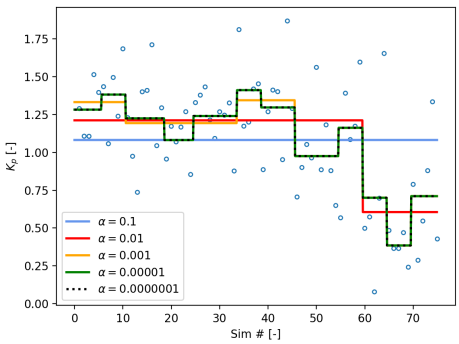


Figure 5.2: Results of the preliminary CART implementation, using different values of α.

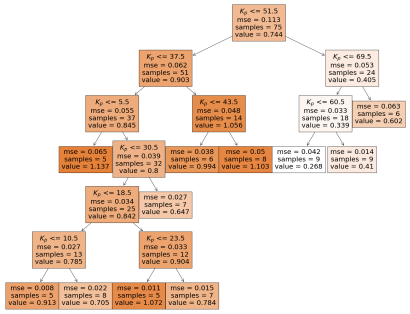


Figure 5.3: The tree grown using $\alpha = 0.0001$. Each node contains the split value of the feature, the mean squared error of the observations in the node, the amount of observations within the node ("samples"), and the value of the node.

5.1.8. CART conclusions and preliminary implementation

CART is a quite straight forward algorithm. It constructs a number of trees by continually splitting the nodes until a stopping condition has been reached. Subsequently it prunes these large trees, and it selects the best pruned tree. Most of these steps require little input; the algorithm decides for itself how to split and what the best tree is. To test the algorithm a preliminary implementation has been made in Python, using the scikit-learn library [96, 97]. This library contains implementations of many machine learning algorithms, among which a CART regression implementation. The scikit-learn implementation is similar to the algorithm explained above, with the only difference being in how the best pruned subtree is selected. Scikit-learn does not do this automatically, instead giving two options for determining tree size:

- The first option is to set the maximum tree depth in advance. With this option no pruning will take place
- The second option does involve pruning, but the 'best' is not selected. Instead the user is asked to give a value for the complexity parameter (α), and the minimum number of observations in a terminal node, and thus choosing the tree size.

The results of the preliminary CART implementation can be found in Figure 5.2, where it is tested on the simulated K_p data (see Section 3.4). The algorithm regresses the dependent variable K_p against the input Sim # (de facto time). Different values for α are used, and the effect can clearly be seen: the smaller α, the flatter the approximation, and if α gets larger the regression matches better. However, at a certain value of α this effect stops, since at that point the tree is as big as it will ever be. An example of a constructed tree, corresponding to $\alpha = 0.0001$, can be found in Figure 5.3. For the minimum required values in a terminal node a value of five was chosen. The effect of varying this parameter will be explored in the next research steps.

5.2. Candidate algorithm - LS-SVR

The second algorithm that will be explained is Least-Squares Support Vector Regression (LS-SVR). In support vector regression (SVR) the goal is to fit a line through the data. This line is called a hyperplane, and the way the line is fit through the data is by taking an allowed error, and letting as many data points as possible fall within this error. In order to solve the regression problem for a nonlinear trend the data can be mapped to a higher dimension, by using kernel tricks. This algorithm was first developed by Vapnik [98], and makes use of quadratic programming. Least-Squares Support Vector Regression is an alternative to standard SVR developed by Suykens and Vandewalle [99]. It makes use of linear programming, instead of quadratic programming, and is therefore both easier to understand and easier to implement. This section explains the algorithm and its preliminary implementation.

5.2.1. The hyperplane and optimisation function

The basic idea of LS-SVR is to find the optimal hyperplane; that is, to fit a function, $f(x)$, through the training data that lies close to as many data points as possible. Where in ordinary least squares one fits this line by minimising the sum of the squared errors, in support vector regression the objective is to minimise the

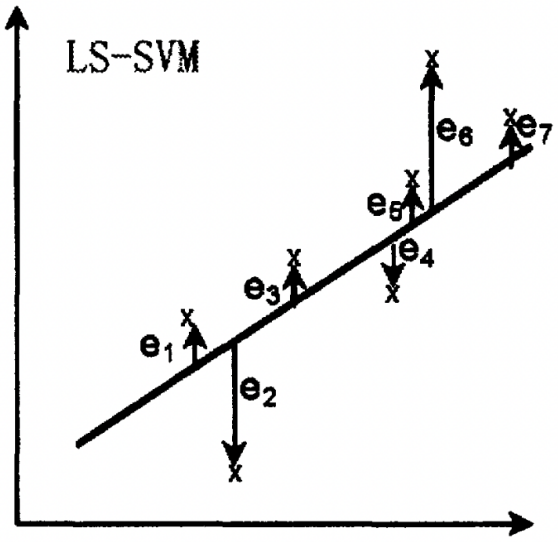


Figure 5.4: The LS-SVR hyperplane and the error terms, adapted from [94]

coefficients instead, handling the error in the constraints of an optimisation problem. This allows for a certain slack in the error; we can select a certain error that is deemed acceptable.

Given the data set (D) given in in Equation 5.14, the approximating function $f(\mathbf{x})$ is given by Equation 5.15, where $\mathbf{w}$ is the coefficient vector, and b the bias term [94].

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}, \quad \mathbf{x}_k \in \mathbb{R}^n, y_k \in \mathbb{R} \tag{5.14}$$

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{5.15}$$

However, the function given in Equation 5.15 is a linear function. LS-SVR can be used for non-linear regression as well. In order to do this the input has to be mapped to a 'high-dimensional feature space', where linear regression can be performed, by using some non-linear function ϕ. The new approximation function corresponding to this approach is given in Equation 5.16.

$$f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x}) + b \tag{5.16}$$

The goal of LS-SVR was formulated into an optimisation problem by Suykens, as given in Equation 5.17 [100]. The objective function consists of two terms, the minimisation of the coefficients, $\mathbf{w}$, and the minimisation of the errors, e_k. These errors are illustrated in Figure 5.4. The trade-off between the two parts of the objective function can be made by altering the regularisation constant, γ, where a lower gamma leads to less emphasis being placed on minimising the errors, thus less overfitting. The objective function is subject to one constraint, namely that the prediction has to equal the actual value y_k corresponding to input $\mathbf{x}_k$, excluding the allowed error.

$$\begin{aligned} \text{minimise} \quad & \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + \frac{1}{2}\gamma \sum_{k=1}^{N} e_k^2 \\ \text{subject to} \quad & y_k = \mathbf{w} \cdot \phi(\mathbf{x}_k) + b + e_k, \quad i = 1, ..., N \end{aligned} \tag{5.17}$$

5.2.2. Solving the optimisation problem

In order to efficiently solve the optimisation problem explained in the previous section ('primal' from now on) a Lagrangian function can be constructed ('dual') [55], which has as saddle point at the solution in both the primal and dual. In the case of LS-SVR, this Lagrangian is given by Equation 5.18, where $\alpha_k \in \mathbb{R}$ are the Lagrange multipliers [101].

$$L = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + \frac{1}{2}\gamma\sum_{k=1}^{N} e_k^2 - \sum_{k=1}^{N}\alpha_k\{\mathbf{w}\cdot\phi(\mathbf{x}_k) + b + e_k - y_k\} \tag{5.18}$$

Finding the saddle point, and thus the optimality conditions, from the Lagrangian leads to the linear system given in Equation 5.19. More information of this derivation can be found in Wang (2005) or Suykens et al. (2002) [94, 100].

$$\begin{bmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & \mathbf{\Omega} + \gamma^{-1}\mathbf{I} \end{bmatrix}\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix} \tag{5.19}$$

In this equation, $\mathbf{y}$ is the column vector of training values, $\mathbf{1}$ is a column vector of ones, and $\boldsymbol{\alpha}$ is the column vector of Lagrange multipliers. $\mathbf{\Omega}$ denotes the dot product: $\mathbf{\Omega} = \phi(\mathbf{x}_k)\cdot\phi(\mathbf{x}_l), \quad k,l = 1,...,N$.

Solving the system in Equation 5.19 for b and α allows for construction of the estimator function, as given in Equation 5.20 [100].

$$f(\mathbf{x}) = \sum_{k=1}^{N}\alpha_k\phi(\mathbf{x})\cdot\phi(\mathbf{x}_k) + b \tag{5.20}$$

5.2.3. Kernels

As stated earlier, in order to perform non-linear regression the input can be mapped to a higher dimension by using some non-linear function ϕ, and by performing linear regression in that higher dimension the result in the original dimension will be non-linear. There is, however, one problem with this approach. Computing the dot product in the higher dimension becomes unfeasible due to its high computational expense. Luckily there is a solution to this problem: the algorithm only depends on the value of the dot product of $\phi(\mathbf{x})$, not on the value of $\phi(\mathbf{x})$ itself. This means a 'kernel trick' can be employed to compute the outcome of the dot product, as in Equation 5.21.

$$\mathbf{\Omega} = \phi(\mathbf{x})\cdot\phi(\mathbf{x}_k) = K(\mathbf{x},\mathbf{x}_k) \tag{5.21}$$

A kernel trick takes advantage of the Kernel function, K, which is a function in the primal space with the same outcome as the dot product in the higher feature space. The easier-to-compute Kernel function can thus be used as replacement for the dot product, as given in Equation 5.22.

$$f(\mathbf{x}) = \sum_{k=1}^{N}\alpha_k K(\mathbf{x},\mathbf{x}_k) + b \tag{5.22}$$

These Kernels have been defined by literature, and different variants exist that each have different properties [102]. For the initial implementation of LS-SVR the so-called Radial Basis Function (RBF) Kernel will be used. The RBF Kernel is given by Equation 5.23.

$$K(\mathbf{x},\mathbf{x}_k) = e^{-\frac{||\mathbf{x}-\mathbf{x}_k||^2}{2\sigma^2}} \tag{5.23}$$

The parameter σ is a kernel parameter that determines how "flexible" the prediction is. A larger value means that the prediction is flatter, but therefore a worse fit for the data. The right value has to be determined as to fit the data as best as possible, while avoiding overfitting.

5.2.4. LS-SVR conclusions and preliminary implementation

Compared to CART, LS-SVR is more difficult to understand, but easier to implement. As with CART, a preliminary implementation has been made in Python, using the scikit-learn library. For this implementation several values for the hyperparameters, γ and σ, have been used, and the results can be found Figure 5.5, tested on the same data as the CART algorithm. The effect of increasing or decreasing the hyper parameters can clearly be seen: increasing σ limits the overfitting, while increasing γ allows the curve to be less flat.

5.3. Candidate algorithm - GRU Network

The final algorithm is the Gated Recurrent Unit Neural Network algorithm (GRU). GRU is a specialised form of a recurrent neural network, and was first developed by Cho et al. (2014) [103]. It uses sophisticated units with gates to solve some of the problems inherent in RNN's (see Section 4.4.5.). GRU Networks are often used for time series forecasting, but with a slight modification can be used for general regression as well.

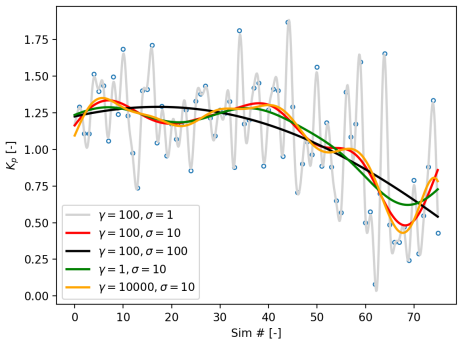


Figure 5.5: Results of the preliminary LS-SVR implementation, using different values of γ and σ.

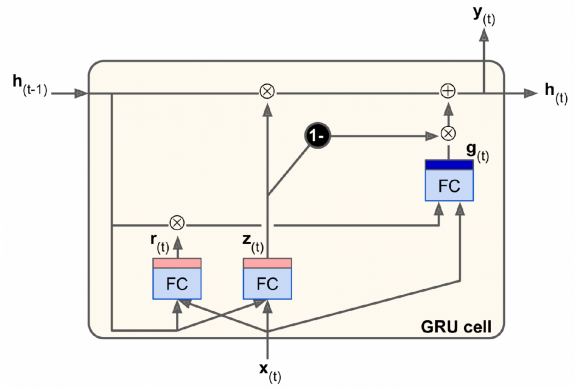


Figure 5.6: A single GRU unit. [69]

5.3.1. The GRU unit

In a GRU layer the neurons are GRU units, that look as in Figure 5.6 [69]. The core of a GRU unit is the state vector, $\mathbf{h}_t$, which serves as a memory. In order to arrive at the current state, the current input vector, $\mathbf{x}_t$, and the previous state vector, $\mathbf{h}_{t-1}$, are fed through three different fully connected layers.[1] These layers are the main layer, which outputs the candidate state, $\mathbf{g}_t$, the update gate, $\mathbf{z}_t$, and the reset gate, $\mathbf{r}_t$. The state vector is determined by Equation 5.24, where $\otimes$ denotes element wise multiplication between two vectors.

$$\mathbf{h}_t = \mathbf{z}_t \otimes \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \otimes \mathbf{g}_t \tag{5.24}$$

The state vector is a combination of the previous state and the candidate state, and the combination is weighted by the update gate, which is given by Equation 5.25. Here, $\sigma(\cdot)$ denotes the sigmoid function (which maps the output into a range of [0,1]). W_{xz} and W_{hz} are weight matrices for the connection of the input and previous state to the gate respectively, and $\mathbf{b}_z$ is the bias term.

$$\mathbf{z}_t = \sigma(W_{xz}^T \mathbf{x}_t + W_{hz}^T \mathbf{h}_{t-1} + \mathbf{b}_z) \tag{5.25}$$

The role of the candidate state is to combine the current input and previous state in order to be able to add new information to the state vector. It is determined using Equation 5.26.

$$\mathbf{g}_t = \tanh(W_{xg}^T \mathbf{x}_t + W_{hg}^T (\mathbf{r}_t \otimes \mathbf{h}_{t-1}) + \mathbf{b}_g) \tag{5.26}$$

In this equation, tanh denotes the hyperbolic tangent function, (which maps the input into a range of [-1,1]), and W and b are again weight matrices and the bias term, respectively. $\mathbf{r}_t$ is the reset gate, and it determines which part of the previous state gets fed into the the candidate state. If r_t is close to zero (in a so called off-state), the reset gate allows the unit to "forget" the previous output, and the candidate output only depends on the input vector [84]. The reset gate is computed by Equation 5.27.

$$\mathbf{r}_t = \sigma(W_{xr}^T \mathbf{x}_t + W_{hr}^T \mathbf{h}_{t-1} + \mathbf{b}_r) \tag{5.27}$$

5.3.2. Constructing a GRU Network

As with any neural network, a GRU network consists of multiple layers of neurons. At least three are needed: an input layer, a GRU layer (the hidden layer), and an output layer. More hidden layers can be added depending on the desired properties of the network.

5.3.3. GRU conclusions and preliminary implementation

As with any neural networks, GRU requires much tinkering and optimisation. This will be done in future stages of this research. For now a preliminary implementation has been made, with one GRU layer, with either 4 or 32 neurons. The results of this, when tested on the K_p data can be found in Figure 5.7. As can be seen, results differ between runs, even when keeping the settings constant. The reasoning for this will also be

[1] Each layer is a de facto fully connected network, where the input gets multiplied by a weight vector, and the output gets filtered through a sigmoid or tangent function.

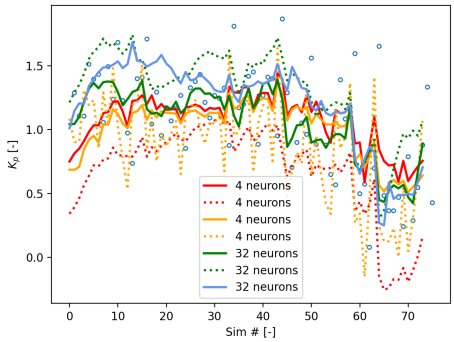


Figure 5.7: Results of the preliminary GRU implementation. As can be seen from the purposefully messy graph, each run gives different results.

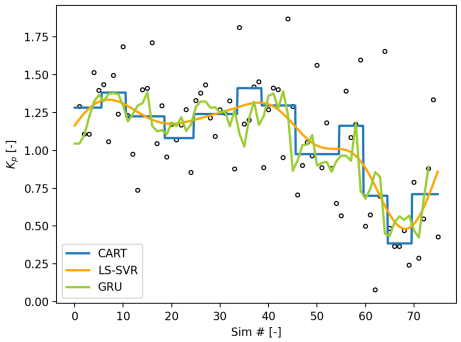


Figure 5.8: Comparison of the results of the three algorithms.

explored in subsequent research. When training a neural network, an amount of epochs has to be specified. An single epoch means training the network with all data once. Too few epochs leads to under-, and too many epochs to overfitting, so an optimal number has to be found. This will also be explored in subsequent research.

5.4. Algorithm trade-off

In this section the three candidate algorithms (CART, LS-SVR and GRU) will be evaluated on various metrics. In order to this, one implementation will be chosen, corresponding to specific values for the hyperparameters. This can be found in Table 5.1. The results of this evaluation are summarised in Table 5.2.

5.4.1. Evaluation metrics

The algorithms will be evaluated according to certain metrics, an overview of which is given in this section. A more detailed explanation can be found in Section 4.3.

The first metrics have to do with the performance of the algorithm. These are the (Root) Mean Squared Error (RMSE) and the Mean Absolute Error (MAE). These are measures that compare the predicted regression curve to a test data set, and determine the prediction error in various ways [104]. Both will be scored quantitatively by calculating their value.

The next two metrics are more subjective: interpretability and Ease of Use. The former says something about how easy it is to understand how the algorithm comes to its results, and is especially important when widely implementing the algorithm, as researchers and medical personnel have to trust the outcome. The latter is purely subjective, and includes such aspects as how much information is available and how easy it is to implement. Both these metrics will be scored on the following scale: [–, -, 0, +, ++], where '-' indicates bad performances, and '+' good performance on the metric.

Finally we have the evaluation metrics that have to do with the implementation of the algorithm: training time and prediction time. These metrics are highly dependent on the size of the data and the hardware on which the algorithms are run, with supercomputers performing better than ordinary laptops. They are therefore often lumped together in one parameter, called computational complexity. The more computational complex an algorithm, the more time it takes to run when using the same computer and dataset [69].

5.4.2. Hyperparameters

Each algorithm comes with different hyperparameters, whose values determine the characteristics of the learning process. In the normal machine learning work flow, the hyperparameters are tuned after a model has been chosen (see Section 4.1.1.). However, in order to make the selection between the three candidate models (CART, LS-SVR, and GRU) in this research a preliminary implementation is evaluated, for which hyperparameter value have to be selected. That is done through visual inspection, using the results in Figure 5.2, Figure 5.5, and Figure 5.7. The hyperparameters and their values for the algorithms are explained in Table 5.1, and the results of the regression using these values are given in Figure 5.8. For some hyperparameters

the effect of changing their values on the regression in this dataset is not yet known, but this will be further investigated in the next stages of this research.

Table 5.1: Hyperparameters of the chosen algorithms, and their values used in the implementations for the trade-off.

Algorithm	Hyperparameter	Value	Comments
CART	α	0.00001	Increasing α leads to a flatter regression curve, but limits overfitting
	Min. observations in terminal node	5	Effect as of yet unknown
LS-SVR	γ	100	Decreasing γ leads to a flatter regression curve
	σ	10	Increasing σ limits overfitting
GRU Network	Neurons in the GRU layer	4	Effect as of yet unknown
	Number of training epochs	100	Effect as of yet unknown

5.4.3. Comparing the algorithms

In the earlier part of this chapter the three algorithms were explained, a long with their implementation, used to compare the algorithms. The input data for each run is the K_p data as explained in Section 5.1.8. The scoring for each of the metrics can be found in Table 5.2.

RMSE and MAE

For determining the RMSE and MAE Equation 4.1 and Equation 4.2 are used. A low score is better than a high score.

Interpretability and Ease of Use

These metrics are scored subjectively. First CART, which is incredibly interpretable. One can exactly follow the tree, and thus the steps the algorithm is taking to score an input. It is also easy to use: implementation is done quickly, and there are only two hyperparameters to tune. LS-SVR is harder to interpret, but it has one advantage: the weights given to the features directly correspond with how important they are in the model, and thus they say something about what the model has learned. Regarding Ease of Use, for LS-SVR two hyperparameters have to be tuned as well, and running the model is simply doing matrix operations. Finally, for GRU interpretability is not good: neural networks are considered black boxes, and it is difficult to understand how the algorithm comes to a prediction. It's Ease of Use is also low: many decisions have to be made about network architecture and training strategy.

Computational Complexity

As explained in section 5.4.1, training time and prediction time are often put together into the computational complexity. Unfortunately, for neural networks computational complexity is hard to determine, and no reliable estimate has been found for the GRU Network used in this research. It is therefore not possible to have an objective measure for this metric. When running the preliminary implementation it has been observed that GRU is significantly slower than CART and LS-SVR, for this small dataset. However, this dataset will likely be larger for future research.

For this trade-off, from literature can be concluded that the CART algorithm scales logarithmic when the training set increases [69, 97], and LS-SVR scales linearly [69, 105]. It is assumed that GRU will scale worse than these two [69]. Because of their subjective evaluation these metrics are scored on the +/- scale.

Table 5.2: Scores for each of the algorithm on the five evaluation metrics.

Algorithm	RMSE	MAE	Interpretability	Ease of Use	Computational complexity
CART	0.27	0.20	++	+	+
LS-SVR	0.30	0.23	0	++	0
GRU	0.25	0.19	- -	-	-

5.5. Conclusions and most promising algorithm

From this literature study three preliminary algorithms have been selected: CART, LS-SVR, and GRU. Each algorithm has been explained, and a preliminary implementation has been made, regressing the K_p data from a single run. A comparison of the three algorithms can be found in Figure 5.8 and Table 5.2. From these results a few notes can be made. First off all, while GRU Network performs better than the other two in the RMSE and MAE metric, the values are close (within 0.05 difference). For the other three metrics (interpretability, Ease of Use, and computational complexity), GRU scores significantly lower, while CART and LS-SVR are comparable.

Since GRU scores low on the latter three metrics, it is discarded as potential algorithm for this research. Regarding CART and LS-SVR, because of their comparable scores the selection has to be made in a different way. Therefore, the focus for the rest of this research will lay on developing LS-SVR, as this has the most interest of the researchers. CART will be used as a back-up option, or potentially as further comparison material.

6

Conclusions and Future research

This chapter serves as a conclusion of this literature study. It comes in the form of a plan for future research, and is divided into 2 sections. Section 6.1 looks back at the research questions given in the introduction, and provides answers to these questions where possible. Section 6.2 explains the steps that will be undertaken for the rest of this thesis.

6.1. Review of the research questions

The introduction gave the research objective and main research question, as well as several sub-questions. Many of these sub-questions can be answered after the work done for this report, and this will be done in this section.

Sub-question 1a: How does the motor behaviour of PD patients change over time?

PD symptoms get worse over time. Due to these symptoms the motor behaviour of PD patients changes: the control gain becomes lower, the neuromuscular damping gets higher, and control variability increases. In general it can be said that PD patients perform worse at tracking tasks the more severe the symptoms.

Sub-question 1b: What data needs to be simulated, and what are the steps that need to be taken in order to simulate this change in tracking task motor behaviour?

The tracking task set-up for this research line involves tracking a target across a touch screen, and the time trace of the control and output signals is recorded. Four cybernetic parameters and two performance parameters can be obtained from this time trace: K_p, τ, ζ_{nms}, and ω_{nms}, and RMS_e and RMS_u. Data needs to be simulated for 75 of these trials, with the patient being healthy for the first 50 trials, and experiencing PD symptoms for the last 25. In order to simulate this, the following steps have to be taken:

1. Simulate a range of healthy trials by drawing the values of the cybernetic parameters from a normal distribution with mean and standard deviation taken from experimental data from earlier research.
2. Simulate the tracking task control system in order to determine the performance parameters.
3. Simulate latter 25 trials by artificially changing the parameter values to represent the effect of PD symptoms.

Sub-question 1c: How can the change of motor behaviour be accurately represented in the simulated data, and how can a variability be introduced?

Experimental data from earlier research can be used to change the cybernetic parameter values in order to simulate a decline in motor behaviour. However, this decline is now random and apart from the split between the 50 "healthy" and 25 "symptom" trials no other trend is introduced. This variable decline will be further investigated in the subsequent stage of this research.

Sub-question 1d: Which bootstrapping method best preserves the dependence present in the participant data?

Due to the data for the rest of this thesis being pure simulated data, selecting the best bootstrapping method is of less importance. By any further need for bootstrapping the Block Bootstrap will be used.

Sub-question 2: What criteria are relevant for evaluating the performance of the chosen models?

Five evaluation metrics are used for evaluating model performance. These are:

- Root Mean Square Error
- Mean Absolute Error
- Interpretability
- Ease of Use
- Computational Complexity

Sub-question 3a: How do linear methods perform when applied to this data set?

Linear trend detection methods were able to successfully detect a decline in motor skills when applied to the dataset with no variety in this decline, being able to distinguish between healthy subjects and subjects with early-stage PD. It is unknown how these methods would perform when the decline varies, but this will be investigated in further research.

Sub-question 3b: Which nonlinear methods are available, and which of these is most fitting for solving the problem in this research?

Many methods are available, divided into three categories: tree-based models, support vector machines, and neural networks. The rest of this research will focus on a support vector machine: Least Squares Support Vector Regression (LS-SVR), with a tree-based model (CART) as backup.

Sub-question 3c: How does the chosen nonlinear model compare to the earlier used linear methods when applied to varying levels of motor symptom change?

This question can not yet be answered and will be further investigated in the subsequent stage of this research.

Sub-question 3d: Does the model have to be unique for each individual, or can a training dataset be used that combines data from multiple individuals?

This question can not yet be answered and will be further investigated in the subsequent stage of this research.

6.2. Further research steps

In the rest of this thesis the unanswered research questions will be investigated. That will be done in the steps below, schematically represented in Figure 6.1.

- **Improve and extend model implementation:** the current implementation of LS-SVR in Python is rudimentary. This needs to be improved and extended, in order to successfully be analysed in the rest of the thesis.
- **Improve data simulation and introduce decline variability:** the current dataset is simulated without decline variability. The current simulation goes from zero symptoms to symptoms, with no gradual change. This can be improved and will be added in this step.
- **Train model and tune hyperparameters:** The model needs to be trained in order to make predictions. This is done using a training dataset. During this training the hyperparameters need to be tuned in order to get the best performing model possible.
- **Sensitivity analysis:** perform sensitivity analysis on the model.
- **Analysis of results:** analysis of results need and comparison to earlier results (Lugtenborg [7]).
- **Model validation:** the model needs to be validated to see if it corresponds to reality.
- **Report and present:** the earlier steps will be repeated if significant changes to the model are made, and if needed a separate algorithm will be tested as well. After this has been done the results and resulting model need to be reported and presented.

6.3. Desired outcome of this research

The steps explained in Section 6.2 have the goal of creating a model that is capable of detecting a change in tracking task control performance due to PD symptoms. It is expected that the nonlinear LS-SVR model will perform better than earlier tested linear methods. The model should require as little calibration as possible, ideally being generalised.

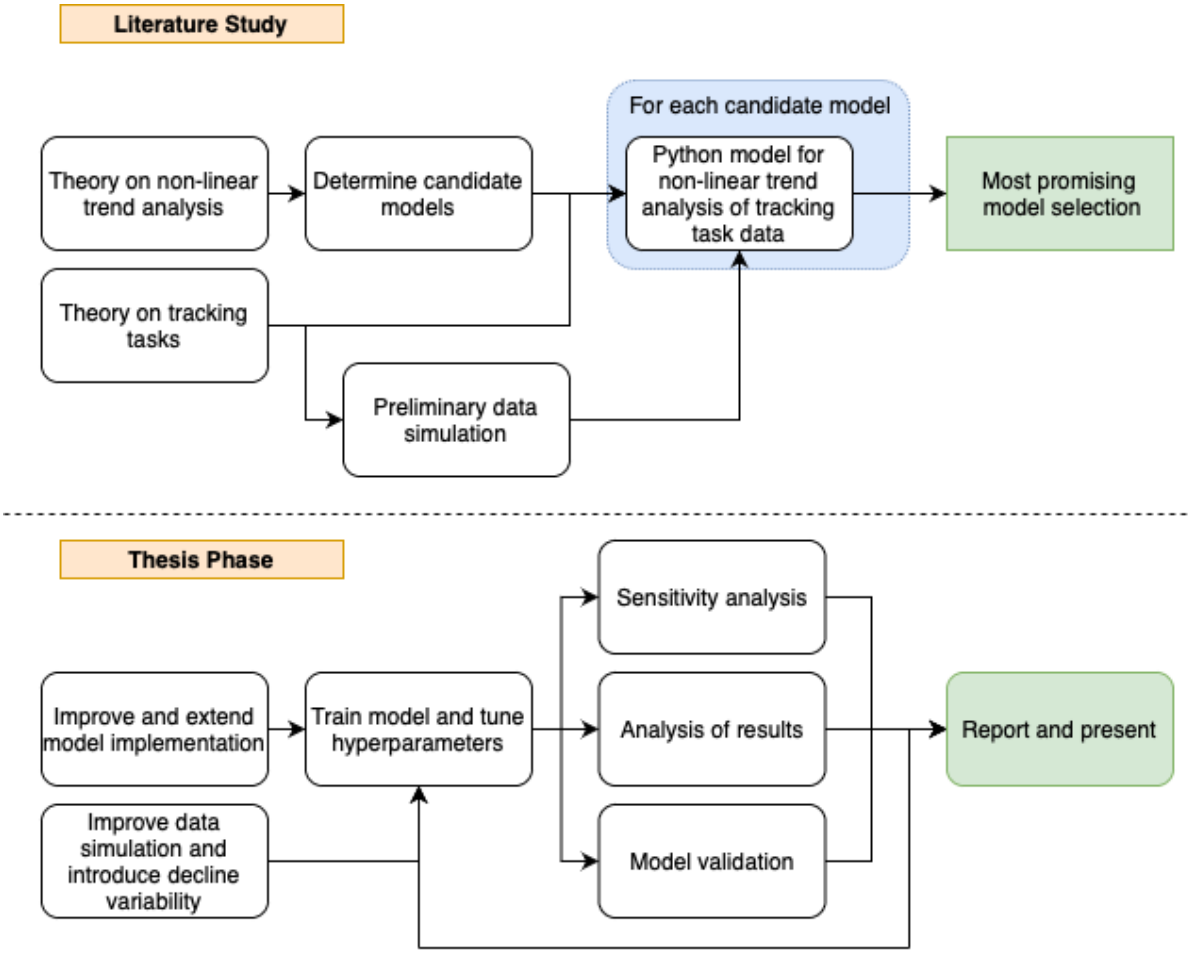


Figure 6.1: Schematic overview of the steps to be taken in the rest of this thesis.

Bibliography

[1] L.V. Kalia and A.E. Lang. Parkinson's disease. The Lancet, 386(9996):896–912, 2015.

[2] J. Palfreman. Hersenstormen: De ziekte van Parkinson en de raadselen van het brein. Uitgeverij Balans, (I. Pieters, trans.). Amsterdam, The Netherlands, 2016.

[3] J. Jankovic. Parkinson's disease: Clinical features and diagnosis. Journal of Neurology, Neurosurgery and Psychiatry, 79(4):368–376, 2008.

[4] M.M. Hoehn and M.D. Yahr. Parkinsonism: onset, progression, and mortality. Neurology, 50(2):318, 1998.

[5] M. Mulder, D.M. Pool, D.A. Abbink, et al. Manual Control Cybernetics: State-of-the-Art and Current Trends. IEEE Transactions on Human-Machine Systems, 48(5):468–485, 2018.

[6] R.J. de Vries. Using Human Operator Modeling for Quantifying Loss of Motor Skills due to Parkinson's Disease. Delft University of Technology, Faculty of Aerospace Engineering, MSc Thesis. Delft, The Netherlands, 2016.

[7] L.A. Lugtenborg. Identifying Behavioural Changes due to Parkinson's Disease Progression in Motor Performance Data. Delft University of Technology, Faculty of Aerospace Engineering, MSc Thesis. Delft, The Netherlands, 2020.

[8] M.A. Myszczynska, P.N. Ojamies, A.M.B. Lacoste, et al. Applications of machine learning to diagnosis and treatment of neurodegenerative diseases. Nature Reviews Neurology, 16(8):440–456, 2020.

[9] N. Eriksen, A.K. Stark, and B. Pakkenberg. Age and Parkinson's Disease-Related Neuronal Death in the Substantia Nigra Pars Compacta. In G. Di Giovanni, V. Di Matteo, and E. Esposito, editors, Birth, life and death of dopaminergic neurons in the substantia nigra, chapter 16, pages 203–213. Springer, New York, USA, 2009.

[10] S. Gepshtein, X. Li, J. Snider, et al. Dopamine Function and the Efficiency of Human Movement. Journal of Cognitive Neuroscience, 26(3):645–657, 2014.

[11] J. Parkinson. An essay on the shaking palsy. Journal of Neuropsychiatry and Clinical Neurosciences, 14 (2):223–236, 1817.

[12] K.R. Chaudhuri, P. Martinez-Martin, P. Odin, et al. Handbook of Non-Motor Symptoms in Parkinson's Disease. Springer Healthcare, London, UK, 2011.

[13] S.N. Gomperts. Lewy Body Dementias: Dementia With Lewy Bodies and Parkinson Disease Dementia. Continuum (Minneap Minn), 22(2 Dementia):435–463, April 2016.

[14] L.M.L. Lau and M.M.B. Breteler. Epidemiology of Parkinson's disease. The Lancet. Neurology, 5(6): 525–535, 2006.

[15] D.J. Gelb, E. Oliver, and S. Gilman. Diagnostic Criteria for Parkinson Disease. Archives of Neurology, 56 (1):33–39, 1999.

[16] M.A. Little, P.E. McSharry, E.J. Hunter, et al. Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. IEEE Transactions on Biomedical Engineering, 56(4):1015–1022, 2009.

[17] R. Das. A comparison of multiple classification methods for diagnosis of Parkinson disease. Expert Systems with Applications, 37(2):1568–1572, 2010.

[18] J.S. Almeida, P.P. Rebouças Filho, T. Carneiro, et al. Detecting Parkinson's disease with sustained phonation and speech signals using machine learning techniques. Pattern Recognition Letters, 125:55–62, 2019.

[19] Z.K. Senturk. Early diagnosis of Parkinson's disease using machine learning algorithms. Medical Hypotheses, 138, 2020.

[20] E. Abdulhay, N. Arunkumar, K. Narasimhan, et al. Gait and tremor investigation using machine learning techniques for the diagnosis of Parkinson disease. Future Generation Computer Systems, 83:366–373, 2018.

[21] C. Cho, W. Chao, S. Lin, et al. A vision-based analysis system for gait recognition in patients with Parkinson's disease. Expert Systems with Applications, 36:7033–7039, 2009.

[22] N. Singh, V. Pillay, and Y.E. Choonara. Advances in the treatment of Parkinson's disease. Progress in Neurobiology, 81(1):29–44, 2007.

[23] P. Zis, K.R. Chaudhuri, and M. Samuel. Phenomenology of Levodopa-Induced Dyskinesia. In S. Fox and J. Brotchie, editors, Levodopa-Induced Dyskinesia in Parkinson's Disease, chapter 1, pages 1–16. Springer-Verlag, London, UK, 2014.

[24] A.J. Espay, F. Morgante, A. Merola, et al. Levodopa-Induced Dyskinesia in Parkinson Disease: Current and Evolving Concepts. Annals of Neurology, 84(6):797–811, 2018.

[25] S. Fox and J. Brotchie. Final Thoughts: Summary and Future Therapeutic Strategies in Levodopa-Induced Dyskinesia. In S. Fox and J. Brotchie, editors, Levodopa-Induced Dyskinesia in Parkinson's Disease, chapter 19, pages 355–359. Springer-Verlag, London, UK, 2014.

[26] C. Brooks, G. Edena, A. Changa, et al. Quantification of discrete behavioral components of the MDS-UPDRS. Journal of Clinical Neuroscience, 61:174–179, 2019.

[27] C.G. Goetz, B.C. Tilley, S.R. Shaftman, et al. Movement Disorder Society-Sponsored Revision of the Unified Parkinson's Disease Rating Scale (MDS-UPDRS): Scale presentation and clinimetric testing results. Movement Disorders, 23(15):2129–2170, 2008.

[28] C. Ramaker, J. Marinus, A.M. Stiggelbout, et al. Systematic Evaluation of Rating Scales for Impairment and Disability in Parkinson's Disease. Movement Disorders, 17(5):867–876, 2002.

[29] C.G. Goetz, J. Kulisevsky, S. Fahn, et al. MDS-UPDRS: The MDS-sponsored Revision of the Unified Parkinson's Disease Rating Scale. International Parkinson and Movement Disorder Society, Milwaukee, WI, USA, 2019.

[30] A.J. Espay, P. Bonato, F.B. Nahab, et al. Technology in Parkinson's Disease: Challenges and Opportunities Alberto. Movement Disorders, 31(9):1272–1282, 2016.

[31] S. Patel, K. Lorincz, R. Hughes, et al. Monitoring Motor Fluctuations in Patients With Parkinson's Disease Using Wearable Sensors. IEEE Transactions on Information Technology in Biomedicine, 13(6): 864–873, 2009.

[32] A. Sánchez-Ferro, M. Elshehabi, C. Godinho, et al. New Methods for the Assessment of Parkinson's Disease (2005 to 2015): A Systematic Review. Movement Disorders, 31(9):1283–1292, 2016.

[33] A. Zhan, S. Mohan, C. Tarolli, et al. Using Smartphones and Machine Learning to Quantify Parkinson Disease Severity: The Mobile Parkinson Disease Score. JAMA Neurology, 75(7):876–880, 2018.

[34] C. de Boer. Visuomotor integration in neurodegenerative brains. Erasmus University Rotterdam, Neuroscience Department, PHD Thesis. Rotterdam, The Netherlands, 2015.

[35] P. Brown and C.D. Marsden. Bradykinesia and Impairment of EEG Desynchronization in Parkinson's Disease. Movement Disorders, 14(3):423–429, 1999.

[36] R. Benecke, J.C. Rothwell, J.P.R. Dick, et al. Disturbance of sequential movements in patients with Parkinson's Disease. Brain, 110:361–379, 1987.

[37] H. Poizner, A.G. Feldman, M.F. Levin, et al. The timing of arm-trunk coordination is deficient and vision-dependent in Parkinson's patients during reaching movements. Exp Brain Res, 133:279–292, 2000.

[38] K. Flowers. Lack of prediction in the motor behaviour of parkinsonism. Brain, 101(1):35–52, 1978.

[39] K. Flowers. Visual 'Closed-Loop' and 'Open-Loop' Characteristics of Voluntary Movement in Patients With Parkinsonism and Intention Tremor. Brain, 101(1):19–34, 1978.

[40] K. Flowers. Some frequency response characteristics of parkinsonism on pursuit tracking. Brain, 101 (1):19–34, 1978.

[41] R.D. Jones and I.M. Donaldson. Tracking tasks and the study of predictive motor planning in Parkinson's disease. Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings, 11 pt 3:1055–1056, 1989.

[42] A. Hufschmidt and C. Lucking. Abnormalities of tracking behavior in Parkinson's Disease. Movement Disorders, 10(3):267–276, 1995.

[43] M. Oishi, P. Talebifard, and M. McKeown. Assessing manual pursuit tracking in Parkinson's disease via linear dynamical systems. Annals of Biomedical Engineering, 39(8):2263–2273, 2011.

[44] E.S. Krendel and D.T. McRuer. A servomechanisms approach to skill development. Journal of the Franklin Institute, 269(1):24–42, 1960.

[45] K. van der El, D. Pool, H. Damveld, et al. An empirical human controller model for preview tracking tasks. IEEETransactions on Cybernetics, 46(10):2609–2621, 2015.

[46] J. Wasicko, D.T. McRuer, and R.E. Magdaleno. Human Pilot Dynamic Response in Single-Loop Systems with Compensatory and Pursuit Displays. Air Force Flight Dynamics Laboratory, Technical Report. WPAFB, OH, USA, 1966.

[47] D. McRuer, D. Pool, and H. Jex. A review of quasi-linear pilot models. IEEE Transactions on Human Factors in Electronics., HFE-8(3):231–249, 1967.

[48] M. van Paassen and M. Mulder. Identification of human operator control behaviour in multiple-loop tracking tasks. IFAC Proceedings Volumes, 31(26):455–460, 2017.

[49] M. Tsionas. Panel data econometrics: theory. (First). Academic Press, London, UK, 2019.

[50] W. Härdle, J. Horowitz, and J. Kreiss. Bootstrap Methods for Time Series. International Statisical Review, 71(2):435–459, 2003.

[51] H. Kunsch. The jackknife and the bootstrap for general stationary observations. Annals of Statistics, 17:1217–1241, 1989.

[52] M. R. Chernick. Bootstrap Methods: A Practioner's Guide. John Wiley & Sons, Inc., New York, USA, 1999.

[53] M. R. Politis and J.P. Romano. The Stationary Bootstrap. Journal of the American Statistical Association, 89:1303–1313, 1994.

[54] V. Hanta and A. Procházka. Rational Approximation of Time Delay. Institute of Chemical Technology in Prague, Department of Computing and Control Engineering, Czech Republic, n.d.

[55] C.M. Bishop. Pattern Recognition and Machine Learning. Springer Science+Business Media, New York, NY, USA, 2006.

[56] Ethem Alpaydin. Introduction to Machine Learning. MIT, (Fourth ed.). Cambridge, MA, USA, 2020.

[57] A. Esteva, K. Chou, S. Yeung, et al. Deep learning-enabled medical computer vision. npj Digit. Med., 4 (5), 2021.

[58] E. Anthes. Alexa, do I have COVID-19? Nature, 586(7827):22–25, 2020.

[59] Sebastian Raschka. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. University of Wisconsin–Madison, Department of Statistics, Madison, WI, USA, 2018.

[60] Jie Ding, Vahid Tarokh, and Yuhong Yang. Model Selection Techniques: An Overview. IEEE Signal Processing Magazine, 35(6), 2018.

[61] I. Guyon and A. Elisseeff. An Introduction to Variable and Feature Selection. Journal of Machine Learning Research, 3:1157–1182, 2003.

[62] S. Shah and A. Kusiak. Data mining and genetic algorithm based gene/SNP selection. Artificial Intelligence in Medicine, 31:183–196, 2004.

[63] Y. Zhang, Z. Dong, P. Phillips, et al. Detection of subjects and brain regions related to Alzheimer's disease using 3D MRI scans based on eigenbrain and machine learning. Frontiers in Computational Neuroscience, 9, 2015.

[64] S. Krishnan and Y. Athavale. Trends in biomedical signal feature extraction. Biomedical Signal Processing and Control, 43:41–63, 2018.

[65] R. Versteeg. Classifying Human Control Behaviour by Artificial Intelligence. Delft University of Technology, Faculty of Aerospace Engineering, MSc Thesis. Delft, The Netherlands, 2019.

[66] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8):1798–1828, 2013.

[67] I. Lee and Y. Shin. Machine learning for enterprises: Applications, algorithm selection, and challenges. Business Horizons, 63:157–170, 2020.

[68] P. Fung, M. Zaidan, H. Timonen, et al. Evaluation of white-box versus black-box machine learning models in estimating ambient black carbon concentration. Journal of Aerosol Science, 2020.

[69] A. Géron. Hands-on machine learning with scikit-learn, keras, and tensorflow: concepts, tools, and techniques to build intelligent systems (Second). O'Reilly Media, Inc., Sebastopol, CA, USA, 2019.

[70] A. Tsanas, M. Little, P. McSharry, et al. Accurate Telemonitoring of Parkinson's Disease Progression by Noninvasive Speech Tests. IEEE Transactions on Biomedical Engineering, 57(4):884–893, 2010.

[71] B. Efron, T. Hastie, I. Johnstone, et al. Least Angle Regression. The Annals of Statistics, 32(2):407–499, 2004.

[72] A. Bayestehtashk, M. Asgari, I. Shafran, et al. Fully automated assessment of the severity of Parkinson's disease from speech. Computer Speech and Language, 29:172–185, 2015.

[73] R. Fernandes and S. Leblanc. Parametric (modified least squares) and non-parametric (theil-sen) linear regressions for predicting biophysical parameters in the presence of measurement errors. Remote Sensing of Environment, 95(3):303–316, 2005.

[74] M. Salmanpour, M. Shamsaei, A. Saberi, et al. Machine learning methods for optimal prediction of motor outcome in Parkinson's disease. Physica Medica, 69:233–240, 2020.

[75] L. Breiman. Classification and regression trees (Ser. Wadsworth statistics/probability series). Wadsworth International Group, 1984.

[76] T. Hastie, J. Friedman, and R. Tibshirani. The elements of statistical learning: data mining, inference, and prediction. Springer Series in Statistics, Springer Science+Business Media, New York, NY, USA, 2009.

[77] B. Choubin, G. Zehtabian, A. Azareh, et al. Precipitation forecasting using classification and regression trees (cart) model: a comparative study of different approaches. Environmental Earth Sciences, 77(8): 1–13, 2018.

[78] L. Breiman. Random Forests. Machine Learning, 45:5–32, 2001.

[79] J. Friedman. Greedy function approximation: a gradient boosting machine. Annals of Statistics, 29(5): 1189–1232, 2001.

[80] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[81] Ömer Eskidere, Figen Ertaş, and Cemal Hanilçi. A comparison of regression methods for remote tracking of Parkinson's disease progression. Expert Systems with Applications, 39(5):5523–5528, 2012.

[82] S. R. Gunn. Support vector machines for classification and regression. Technical Report, Department of Electronics and Computer Science. University of Southampton, Southampton, UK, 1998.

[83] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. Nature, 521:436–444, 2015.

[84] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. CoRR, abs/1412.3555, 2014.

[85] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735–1780, 1997.

[86] K. Cho, B Van Merrienboer, C. Gulcehre, et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. CoRR, abs/1406.1078, 2014.

[87] Y. LeCun and Y. Bengio. Convolutional networks for images, speech and time series. The handbook of brain theory and neural networks, 3361(10):1995, 1995.

[88] A. Borovykh, S. Bohte, and C.W. Oosterlee. Conditional time series forecasting with convolutional neural networks. arXiv preprint, arXiv:1703.04691, 2018.

[89] H. Hewamalage, C. Bergmeir, and K. Bandara. Recurrent Neural Networks for Time Series Forecasting: Current status and future directions. International Journal of Forecasting, 37(1):388–427, 2021.

[90] T. Wang, R.G. Qiu, and M Yu. Predictive modeling of the progression of Alzheimer's disease with recurrent neural networks. Sci. Rep., 8:1–12, 2018.

[91] C. Che, C. Xiao, J. Liang, et al. An rnn architecture with dynamic temporal matching for personalized predictions of parkinson's disease. In Proceedings of the 7nd SIAM International Conference on Data Mining, pages 198–206, Minneapolis, MN, USA, 2017. SIAM.

[92] R. Nisbet, G. Miner, and K. Yale. Handbook of statistical analysis and data mining applications (Second). Academic Press, an imprint of Elsevier., London, UK, 2018.

[93] E. Dilmen and S. Beyhan. A Novel Online LS-SVM Approach for Regression and Classification. IFAC, 50 (1):8642–8647, 2017.

[94] H. Wang and D. Hu. Comparison of SVM and LS-SVM for regression. In 2005 International Conference on Neural Networks and Brain, volume 1, pages 279–283, 2005.

[95] A.J. Izenman. Modern Multivariate Statistical Techniques. Regression, Classification, and Manifold Learning. Springer Series in Statistics, Springer Science+Business Media, New York, NY, USA, 2008.

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

[97] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pages 108–122, 2013.

[98] Vladimir N. Vapnik. The Nature of Statistical Learning Theory. Springer, New York, NY, USA, 1995.

[99] J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifier. Neural Processing Letters, 9:293–300, 1999.

[100] J.A.K. Suykens, T. van Gestel, J. De Brabanter, B. de Moor, and J. Vandewalle. Least Squares Support Vector Machines. World Scientific Publishing Company, Singapore, 2002.

[101] József Valyon and Gábor Horváth. A robust LS-SVM regression. Transactions on Engineering, Computing and Technology, 7:148–153, 01 2005.

[102] A.J. Smola and B. Schölkopf. A tutorial on support vector regression. Statistics and Computing, 14: 199–222, 2004.

[103] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. CoRR, abs/1409.1259, 2014.

[104] A. Zheng. Evaluating machine learning models : a beginner's guide to key concepts and pitfalls (First). O'Reilly Media, 2015.

[105] Y. Tian, X. Ju, Z. Qi, and Y. Shi. Efficient sparse least squares support vector machines for pattern classification. Computers and Mathematics with Applications, 66:1935–1947, 2013.

III

Appendices to Scientific Article

to be graded for AE5310 Thesis Control and Operations

A

Regression example

This appendix walks through a complete analysis for a single participant, as an example to help understand the method.

A.1. Data simulation

The first step is to simulate the participant. For this example the settings from Tables A.1 and A.2, with the numpy.random seed being: 12345. 50 healthy trials are simulated, and 25 symptomatic. The results of the data simulation are given in Figures A.1 till A.6

Table A.1: Parameters for the normal distribution used to simulate the healthy trials

	K_p [-]	τ [s]	ζ_{nms} [-]	ω_{nms} [rad/s]
μ	0.91	0.37	0.38	8.7
σ_p	0.26	0.14	0.19	4.3

Table A.2: Range for δ for simulating the PD symptoms

	Range for δ
K_p	$-0.9 < \delta < 0$ [-]
τ	$0 < \delta < 0.07$ [s]
ζ_{nms}	$0 < \delta < 0.31$ [-]
ω_{nms}	$0 < \delta < 5$ [rad/s]

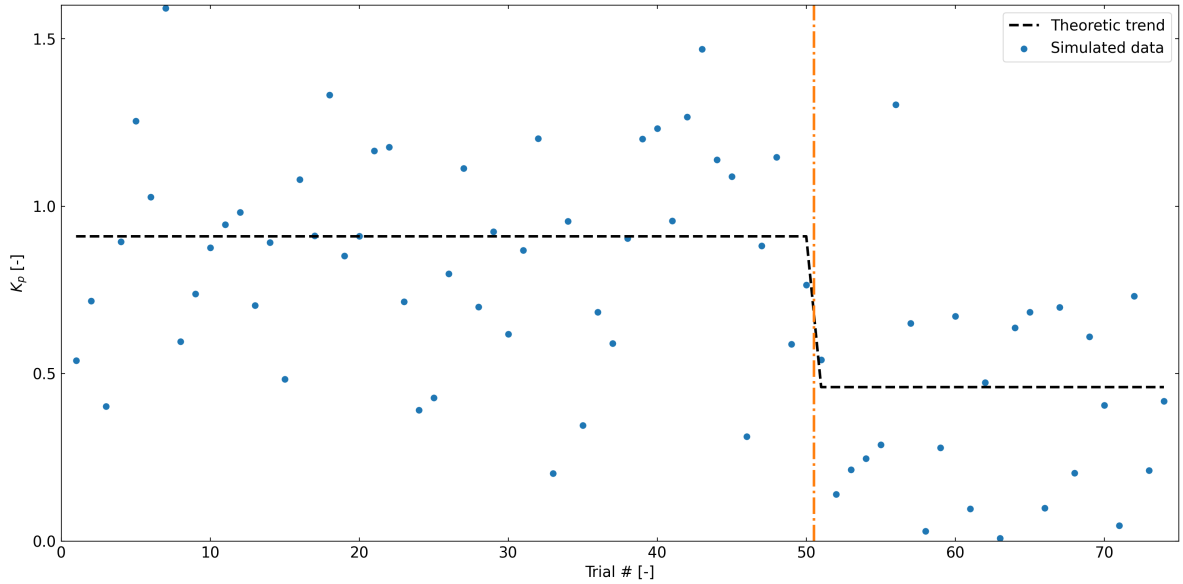


Figure A.1: Simulated data for K_p

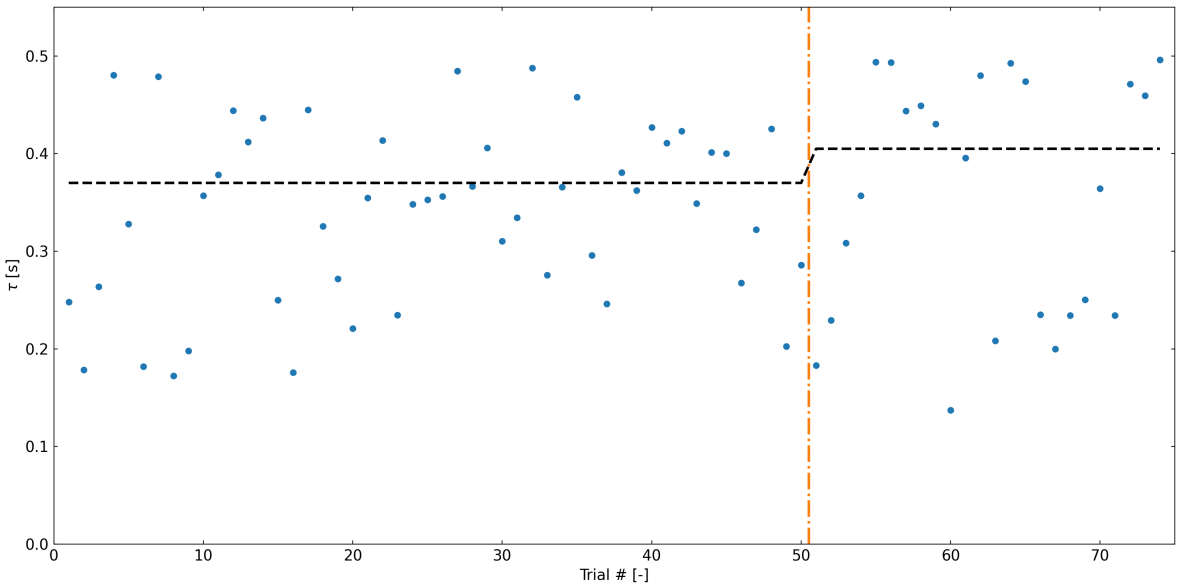


Figure A.2: Simulated data for τ

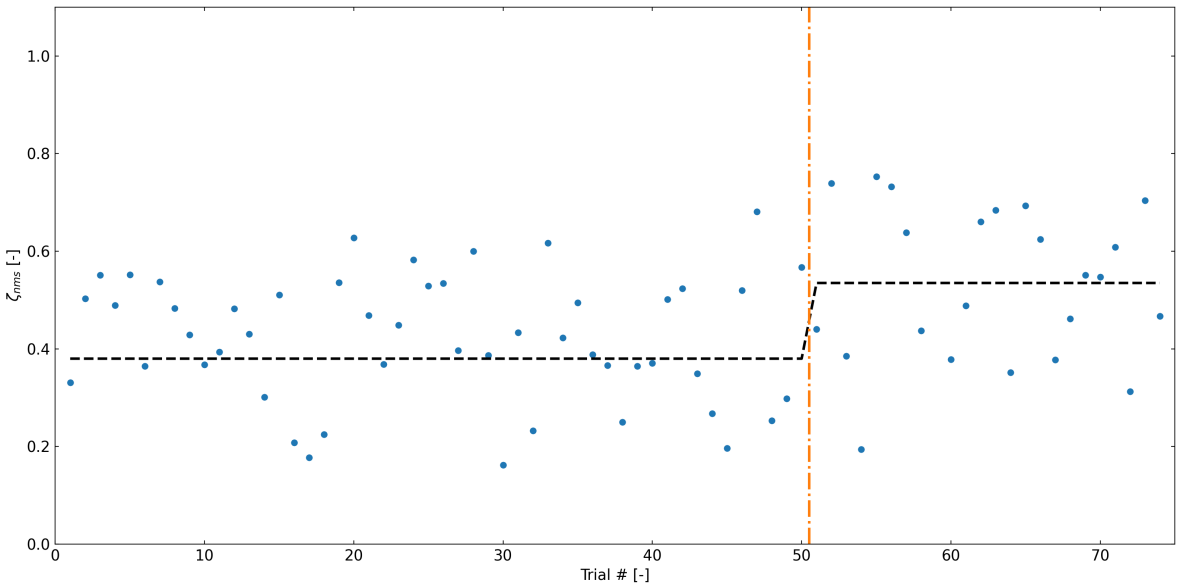


Figure A.3: Simulated data for ζ_{nms}

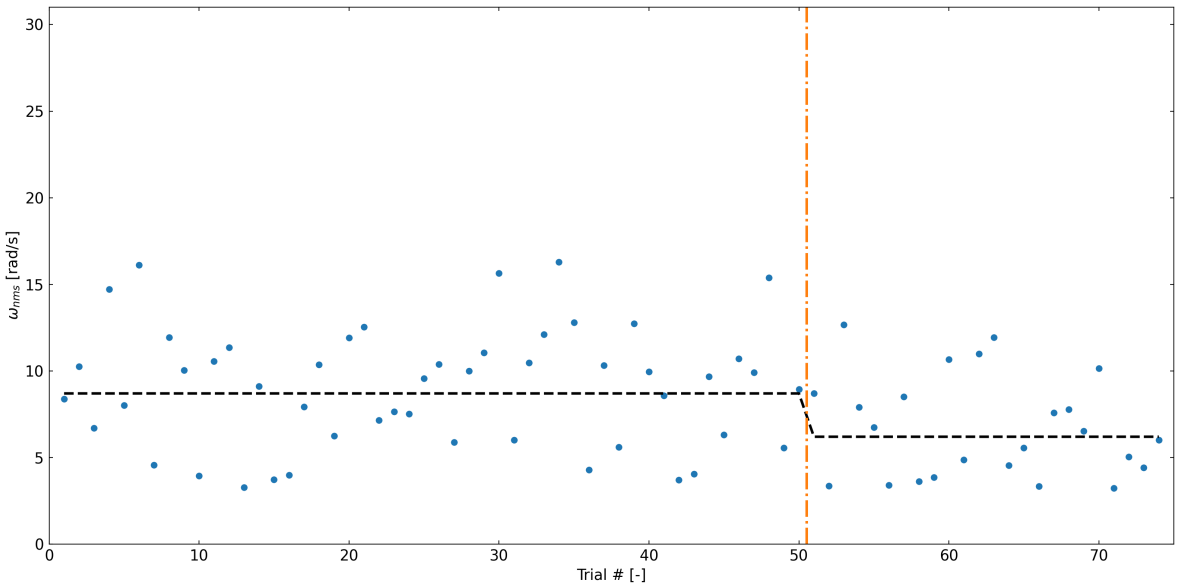


Figure A.4: Simulated data for ω_{nms}

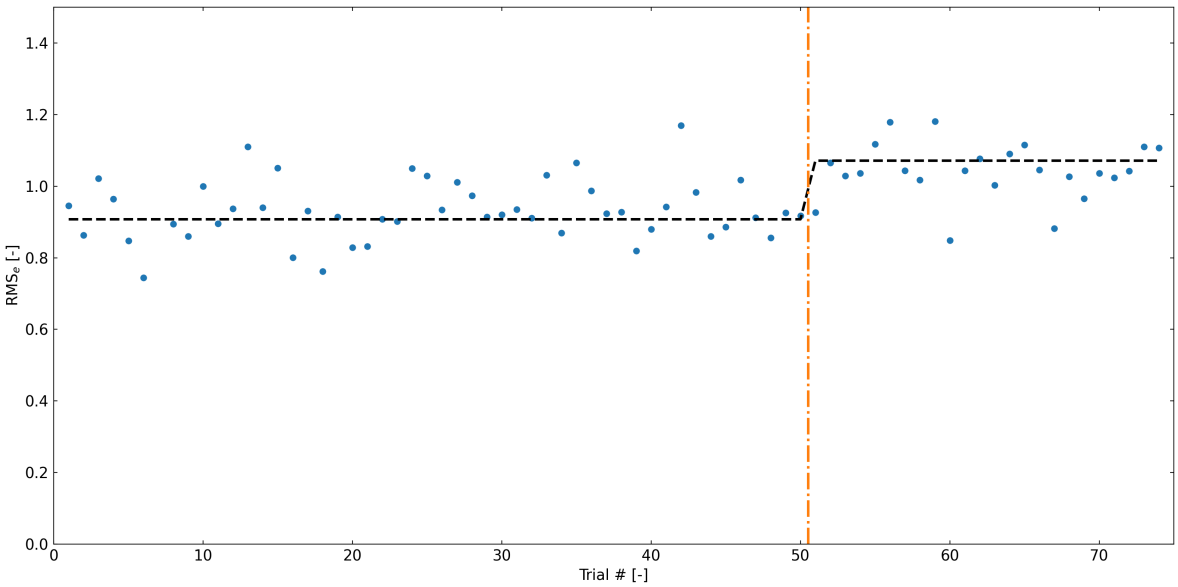


Figure A.5: Simulated data for RMS$_e$

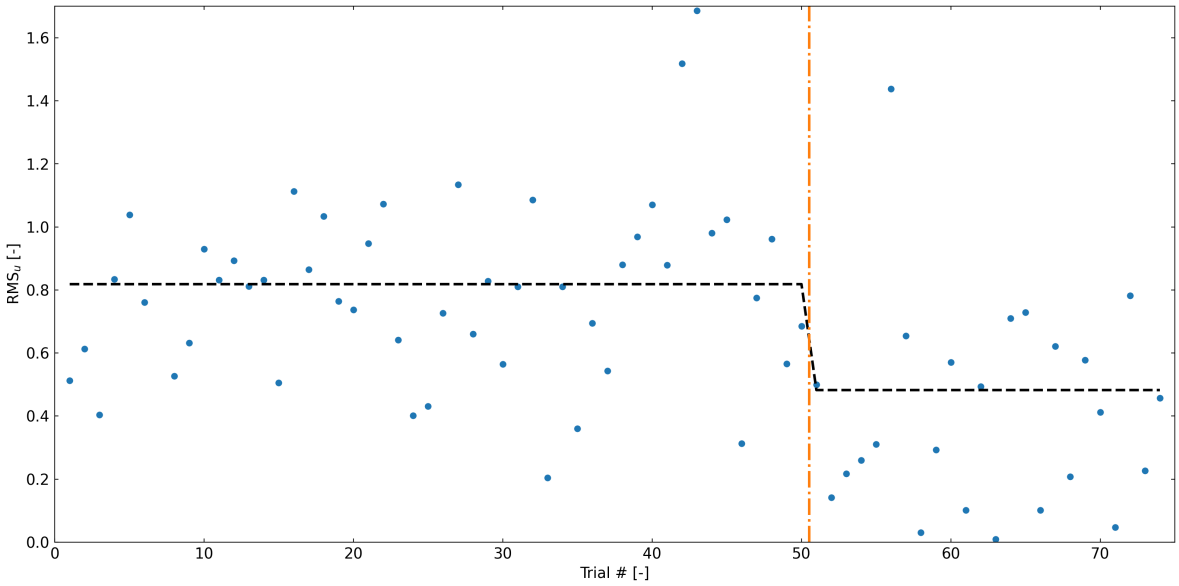


Figure A.6: Simulated data for RMS$_u$

A.2. Regression

On these simulated data the two regression models are run. For the LS-SVR model the hyperparameter settings in Table A.3 are used. The results are given in steps of 10 trials, in Figures A.7 till A.14.

Table A.3: LS-SVR hyperparameter values for the example presented in this Appendix

	K_p	τ	ζ_{nms}	ω_{nms}	RMS_e	RMS_u
γ	5	1	5	1	5	40
σ	60	50	100	100	10	100

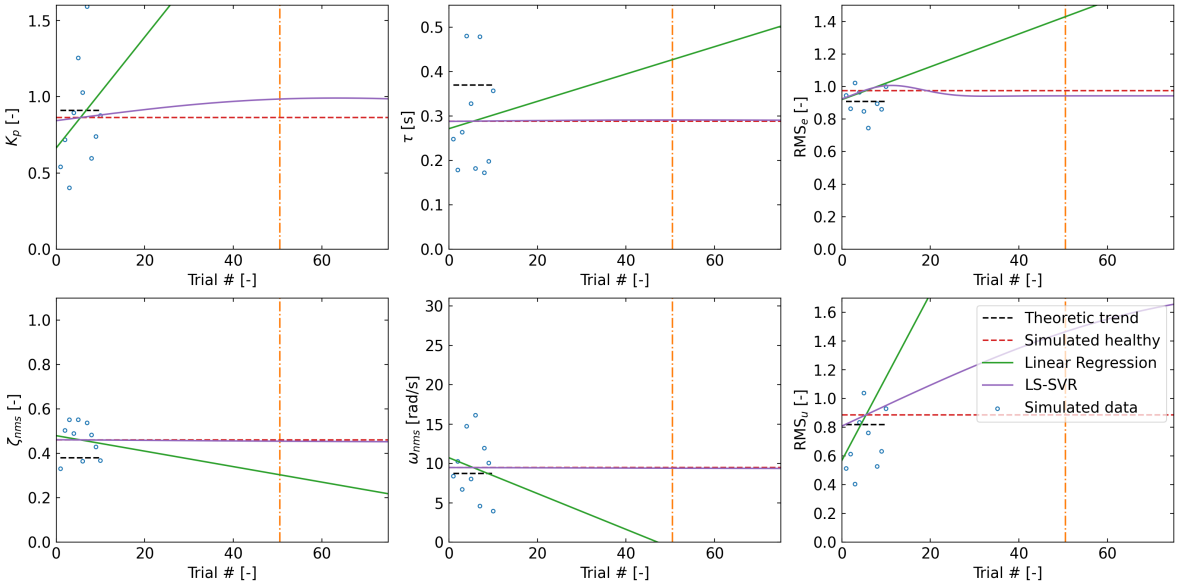


Figure A.7: Regression results for trial 10

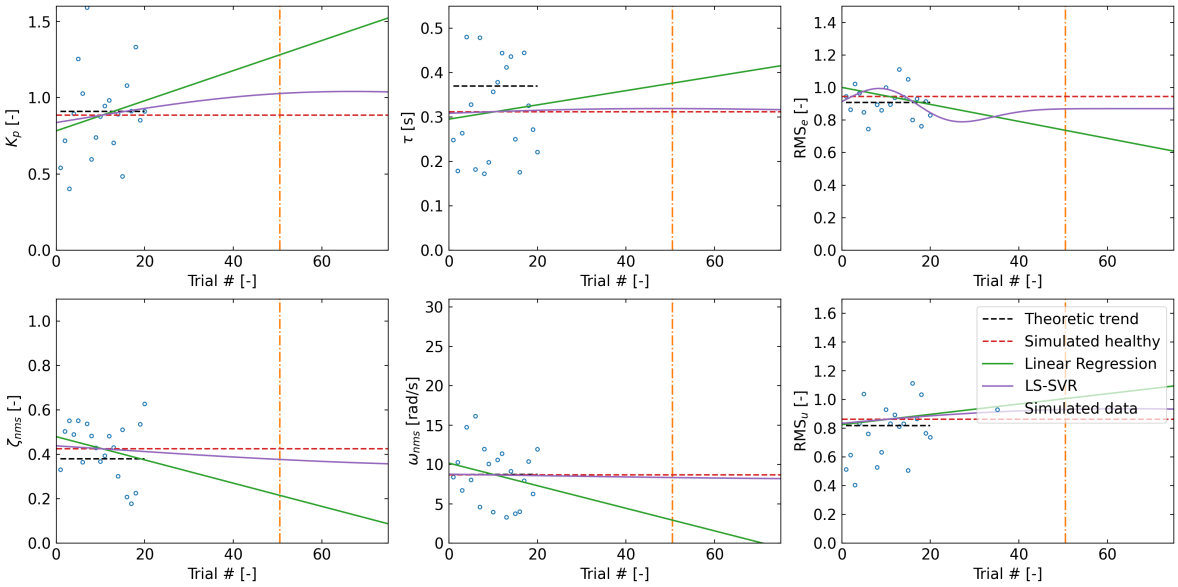


Figure A.8: Regression results for trial 20

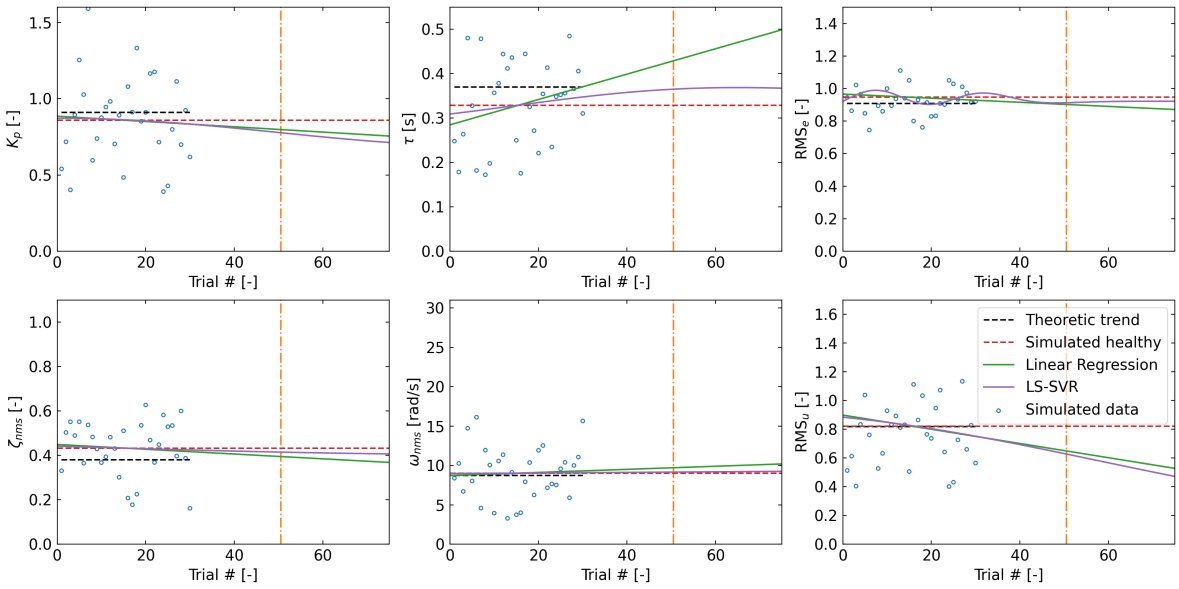


Figure A.9: Regression results for trial 30

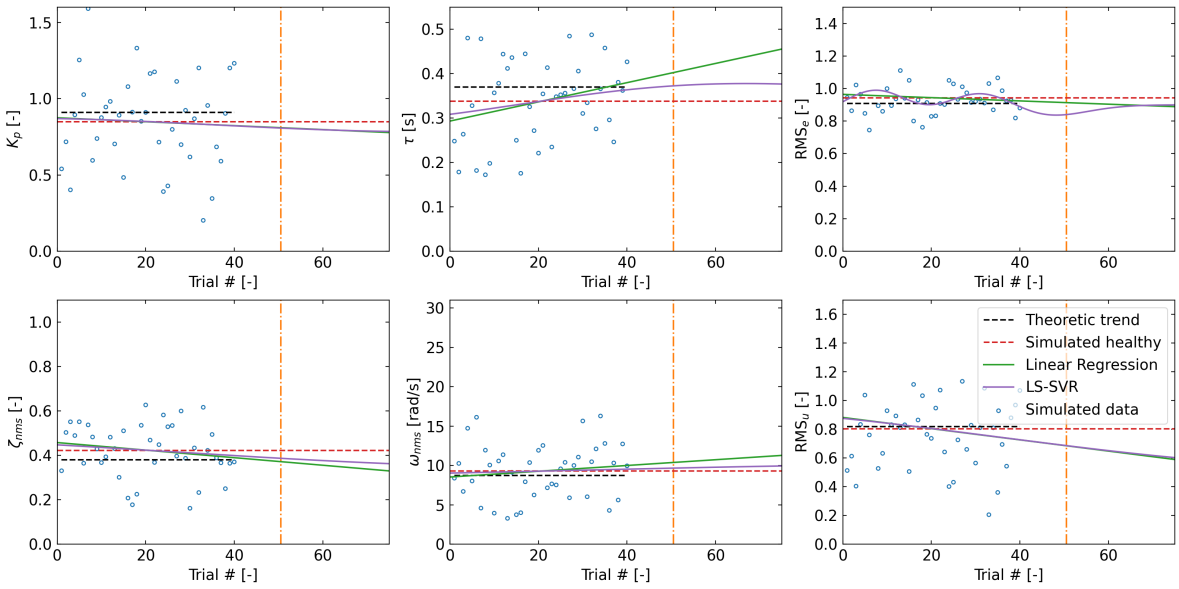


Figure A.10: Regression results for trial 40

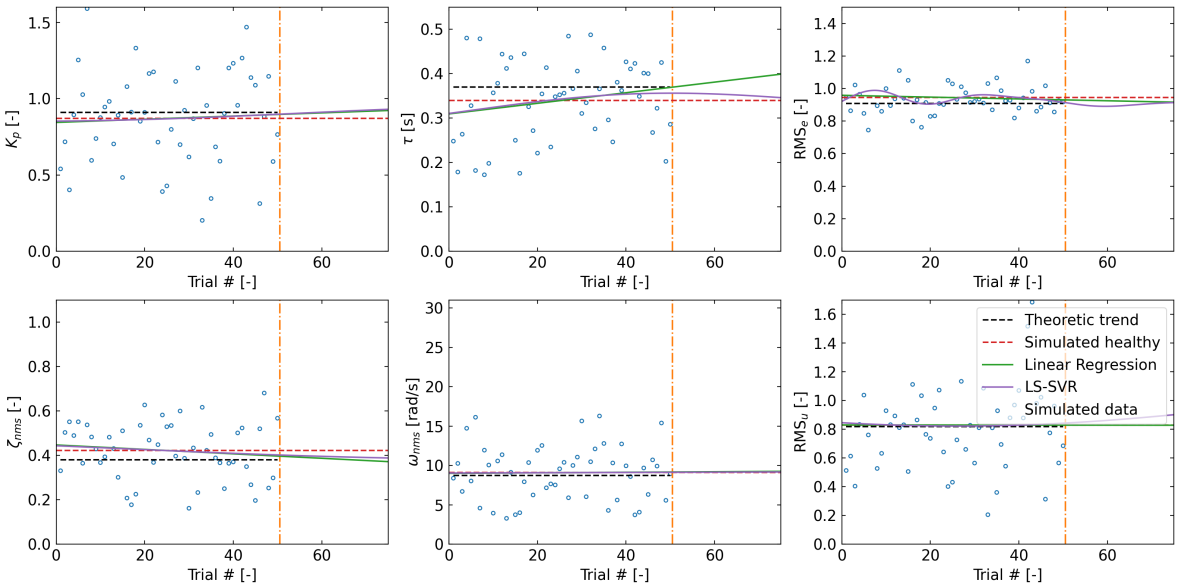


Figure A.11: Regression results for trial 50

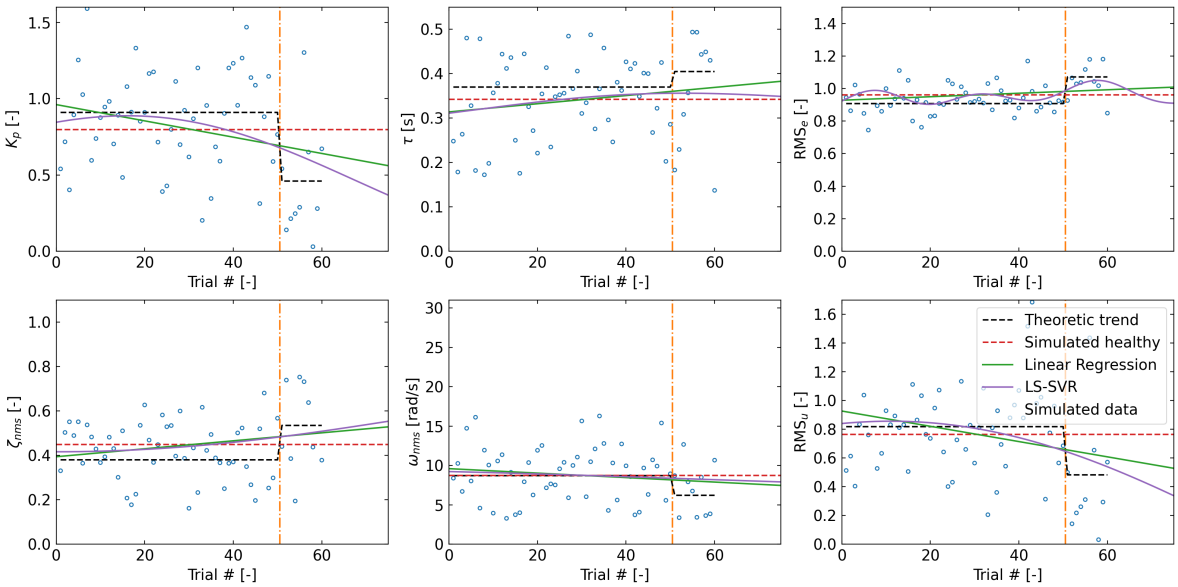


Figure A.12: Regression results for trial 60

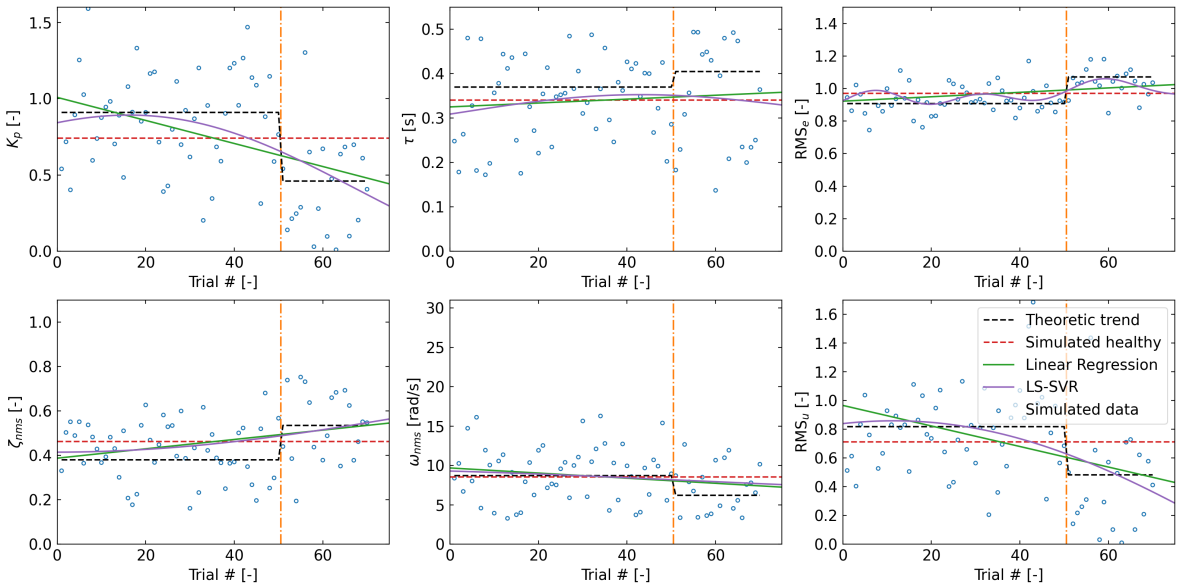


Figure A.13: Regression results for trial 70

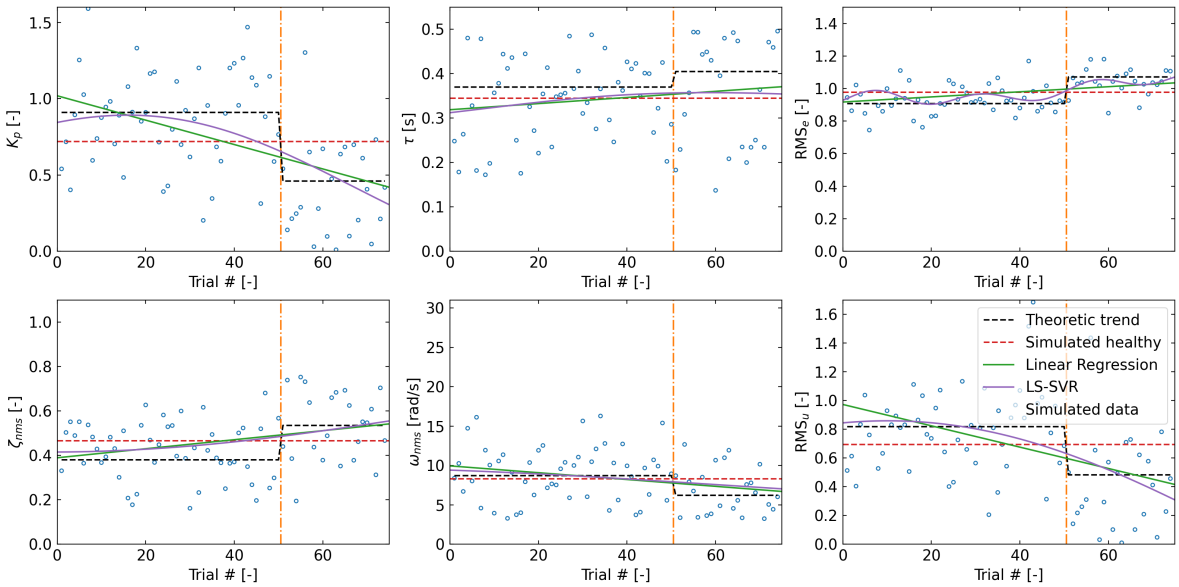


Figure A.14: Regression results for trial 74

A.3. Difference

For each trial the difference is determined. This difference can be plotted over time, as show in Figure A.15. From the difference graph a detection of a changing trend can take place, with D_{lim} and l_{det} used given in Table A.4. In the figure it can be seen that for four parameters a change is detected: K_p (at trial 56), ζ_{nms} (60), ω_{nms} (70) and RMS$_u$ (52). For τ and RMS$_e$, D does not stay above D_{lim} for at least 5 trials.

Table A.4: LS-SVR hyperparameters for the example presented in this Appendix

	K_p	τ	ζ_{nms}	ω_{nms}	RMS$_e$	RMS$_u$
D_{lim}	0.15	0.03	0.06	1.0	0.1	0.1
l_{det}	5	5	5	5	5	5

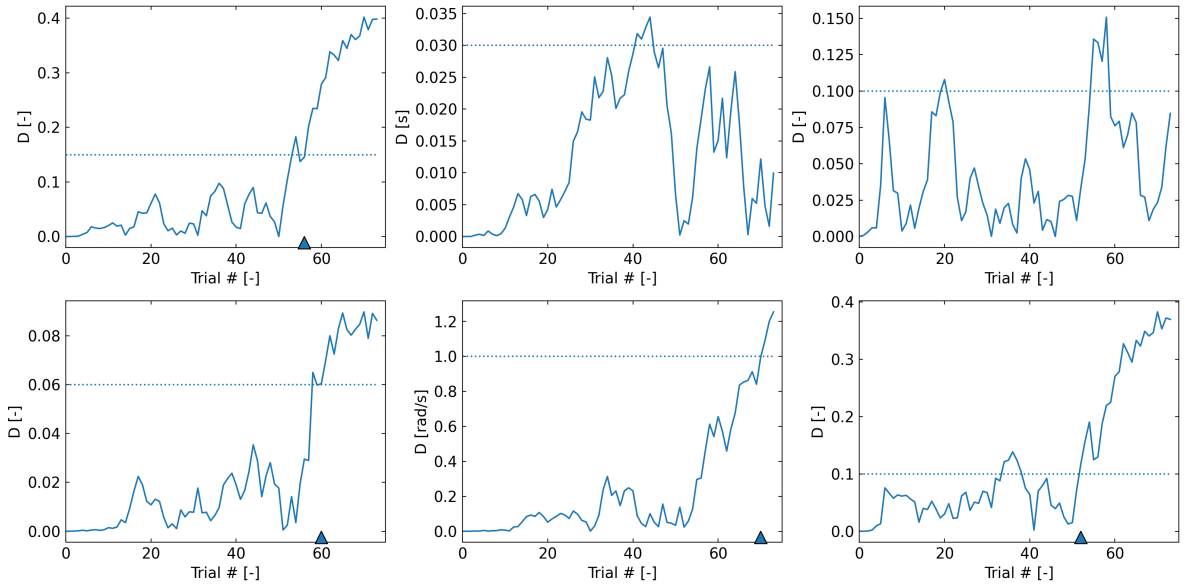


Figure A.15: Difference plots for this simulated participant

B

Future results

This appendix gives the complete results of the example participant from step 4 in the analysis: using projected future trials. For every parameter it can be seen that the larger n_{future} the earlier a detection will occur.

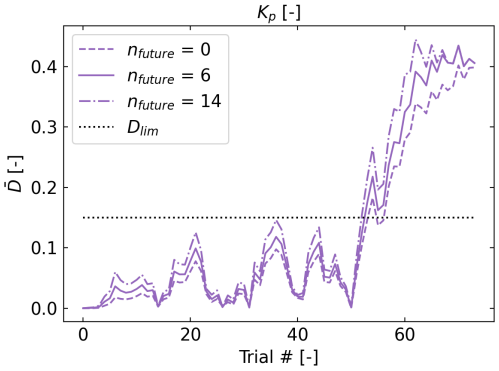


Figure B.1: Results for K_p, step 4, single participant

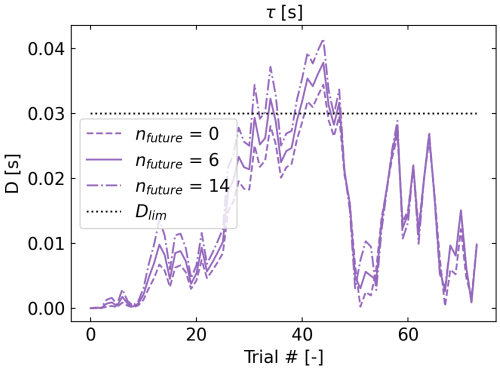


Figure B.2: Results for τ, step 4, single participant

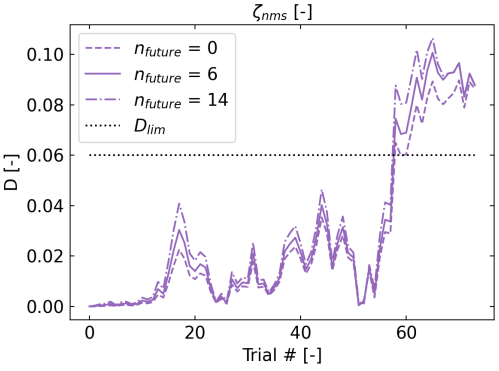


Figure B.3: Results for ζ_{nms}, step 4, single participant

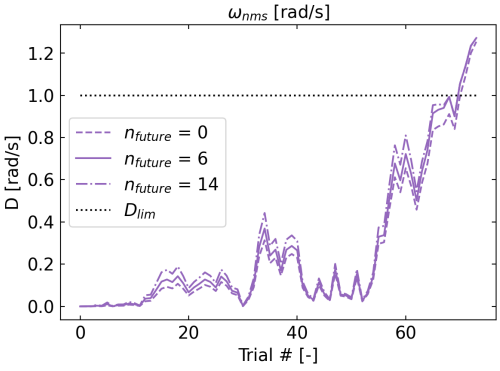


Figure B.4: Results for ω_{nms}, step 4, single participant

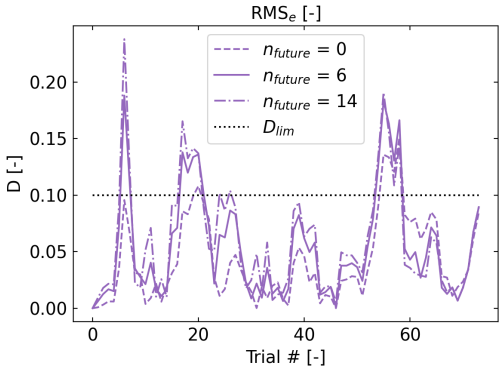


Figure B.5: Results for RMSE$_e$, step 4, single participant

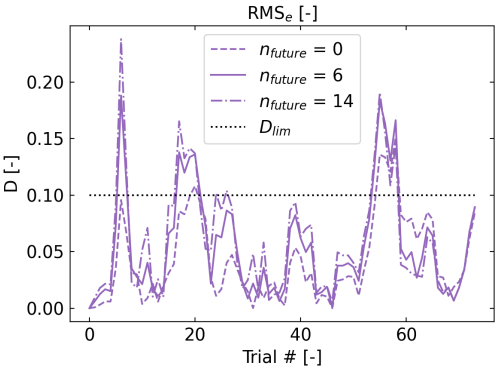


Figure B.6: Results for RMSE$_u$, step 4, single participant