2016•2017
FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
master in de industriële wetenschappen: elektronica-ICT

Masterproef
Applying machine learning algorithms on multi-sensor applications

Promotor :
Prof. dr. ir. Bart VANRUMSTE

Copromotor :
De heer Marijn LEMMENS

Promotor :
prof. dr. ir. RONALD THOELEN

Tom Kelher

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

Gezamenlijke opleiding Universiteit Hasselt en KU Leuven

2016•2017

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterproef

Applying machine learning algorithms on multi-sensor applications

Promotor :
Prof. dr. ir. Bart VANRUMSTE

Copromotor :
De heer Marijn LEMMENS

Promotor :
prof. dr. ir. RONALD THOELEN

Tom Kelher

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

universiteit hasselt | KU LEUVEN

Acknowledgements

I'd like to thank everyone who helped me with my thesis during the year. I'm especially grateful to Drs. ing. Marijn lemmens, for guiding me through my thesis and offering me all the needed support, which includes correcting my thesis, poster and abstract.

A very special gratitude goes to my promotor Prof. dr. ir. Ronald Thoelen and Mr. Thijs Vandenryt, for attending several meetings and giving me feedback on my projects.

I would also like to thank my promotor Prof. dr. ir. Bart Vanrumste for participating in brainstorm meetings with me, and giving some useful ideas.

Furthermore, I'd like to thank everyone who helped me from the IMO-research team, including but not limited to, Mr. Gilles Oudebrouckx and Mr. Frederik Vreys.

Next, I'd like to thank all the students who helped me and accompanied me during my thesis.

And finally, last but by no means least, would I like to thank my family for making my studies possible and offering me emotional support.

Table of contents

List of figures

Abstract

IMO-IMOMEC (standing for: Institute for Materials Research - Institute for Materials Research in MicroElectronics) at Hasselt researches the possibility to implement electronics in the medical sector. These electronics are accompanied by logic and machine learning. The goal of this thesis is to do a feasibility study on the implementation of machine learning in medical applications, with the necessary steps to achieve this.

This thesis studies the machine learning algorithms by using prediction- and classification algorithms. The prediction projects include a study on the growth of yeast cells and the transition between two different liquids in a pipeline, both are accomplished with the help of impedance measuring techniques. The classification project implements tomography on wounds and classifies these results afterwards based on their intensity.

The prediction projects are both realised with a double exponential-, logarithmic- and power law regression. The matlab library EIDORS made tomography possible with help of image processing, many electrode models have been tested to determine the most efficient set-up. The classification exists out of a neural network with three hidden layers that classifies the tomographic images of the wounds.

IMO-IMOMEC (staande voor: Institute for Materials Research - Institute for Materials Research in MicroElectronics) te Hasselt doet onderzoek naar het implementeren van elektronica in de medische sector. Deze elektronica wordt vergezeld door logica en machine learning. Het doel van deze masterproef is om een haalbaarheidsstudies te doen naar de implementatie van machine learning in medische toepassingen aan de hand van de nodige tussenstappen.

Deze masterproef bestudeert machine learning algoritmen door gebruik te maken van voorspellings- en classificatie algoritmen. De voorspellingsprojecten bestaan uit het bestuderen van de groei van gistcellen en de overgang van vloeistoffen in leidingen, beide zijn verwezenlijkt door middel van impedantiemetingen. Het classificatie project past tomografie op wonden toe en zal deze vervolgens classificeren op basis van hun intensiteit.

De voorspellingsprojecten zijn beiden verwezenlijkt met dubbel exponentiële-, logaritmische- en machtsregressie. De tomografie wordt verwezenlijkt met EIDORS in Matlab en beeldverwerking, verschillende electrode modellen zijn uitgetest om de meest efficiënte te bepalen. Tenslotte is er de classificatie met een neuraal netwerk van drie hidden layers die de wonden classificeert.

1 Introduction

Machine learning, or any form of artificial intelligence has a great future, and it is therefore no surprise that there are many possible ways to solve a certain problem. One way might be better than the other. In this introduction, you'll learn the basics about machine learning, how to start the process from problem to solution and how you can implement those ideas.

1.1 What is machine learning

Machine leaning gives computers the ability to learn without being exactly programmed. It has evolved from pattern recognition and is a form of artificial intelligence.
It's about creating or implementing algorithms that make predictions on data [1]. These predictions will be different for each project. Which algorithms and further information will be given in other chapters.

1.2 Machine learning task chart

The following image will give a summary of every machine learning project. It is a chart that can be followed by beginners to make sure that nothing on the way will be forgotten.

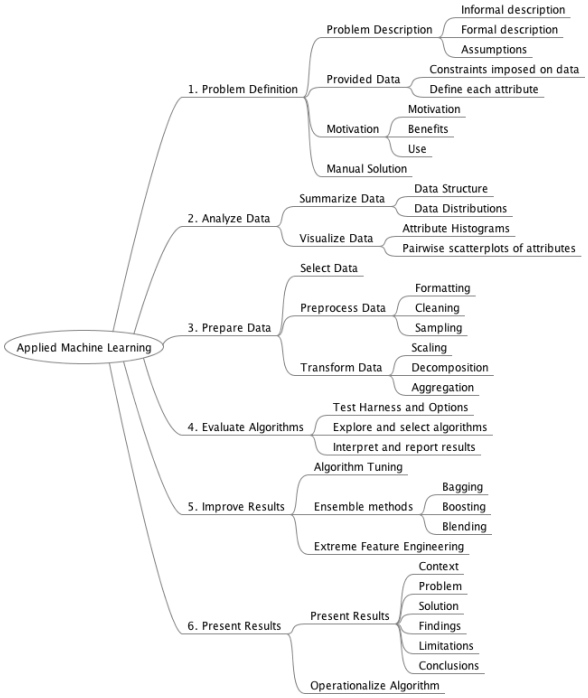

Figure 1: Machine learning flow chart, from problem to result [2]

In the chart, you can see that you first should define your problem. It's very important that you know the details of the problem and the data you'll get to help you form an optimal solution.

Secondly you need to have a look at the data. How does it spread out, is it easily classifiable by the eye? Many questions pop up, and the answers to these questions will help you determine which machine learning algorithms are suitable as solution and what kind of data they'll need.
The next step is pre-processing your data. This means that you'll have to sample and clean up redundant data, to enhance the speed of your algorithm. A very important aspect in this step is that you form the data so that it becomes suitable for your problem and algorithm. This aspect is often referred to as feature extraction or feature selection. Feature extraction is a method where you preform calculations with the given data to form one or more suitable features for your algorithm. Feature selection on the other hand is about selecting parts of the data to consider it as a separate feature. Which extraction method you'll have to implement depends strongly on the given data and on the project, itself [3].
The fifth step will be the testing of algorithms on your data. When you find a few suitable algorithms, you'll want to test them for their accuracy. The one that turns out to give the best results, will be your best choice.
After this has been done, further improving is advised. This improvement can be achieved by implementing more algorithms in your projects, but there are also other means to achieve a better accuracy. This improving of accuracy is called bagging in machine learning terms [4]. Bagging itself is an ensemble technique that takes the prediction of multiple algorithms and forms this into a more accurate prediction.
Finally, there will be a presentation of the acquired results. Why did you modify the data, why did you pick the algorithms, what is the accuracy and can this be further improved, these are a few of the questions that should be answered to ensure your client that you came up with an optimal solution.

1.3 Implementing the ideas

To implement the pre-processing and algorithms, one must code. The most popular methods are Matlab, which has a classification learner since version 2015a [5], python and R. All these options are based on scripting where python and R are the open source solutions and therefore free of charge. It's possible to combine both python and R with each other is an environment called Microsoft Azure where the basic license is also free of charge [6].

1.4 In this project

The goal of this project is, as explained in the abstract, to do a feasibility study on machine learning applied in medical applications. There are three main projects here. The first one is the regression on the impedance of liquids in pipelines. The second is a regression paired with a classification on yeast data. Finally, there is a classification of wounds based on their tomography data. The least project is divided into many subchapters, one that researches tomography, one that explains wound segmentation and finally there is a classification made with a neural network.

2 Tomography

Although there are many forms of tomography, this thesis will only focus on electrical impedance tomography and its implementation in Matlab.

2.1 What is electrical impedance tomography

Electrical impedance tomography (EIT) is a technique that uses electrical current with a frequency to examine an object. It's achieved by placing an object in an electrode model, some of these electrodes will inject current while the others measure the resulting electrical potential field. It's called impedance tomography because this method is often used to measure the dielectric constant for different frequencies, to inspect the differences between the data. When you invert the data, you'll receive the conductivity or resistivity of the object inside the electrode model. This method is not yet popular in the medical field but can be very promising [7]. A useful example would include an electrode circle placed around an arm, the technique could identify the bones and tissue and could be used to detect certain inconvenient defects of the natural human body. Another example is placing an electrode model around a wound to follow up its healing process, the electrode model will then be implemented in a bandage or other medical equipment that is used to protect the wound.

The following image displays such an electrode model. There's an object placed inside the model and the data is measured and collected. On the right side, we see the inverted data, or the conductivity/resistivity map. The red spot indicates the object.

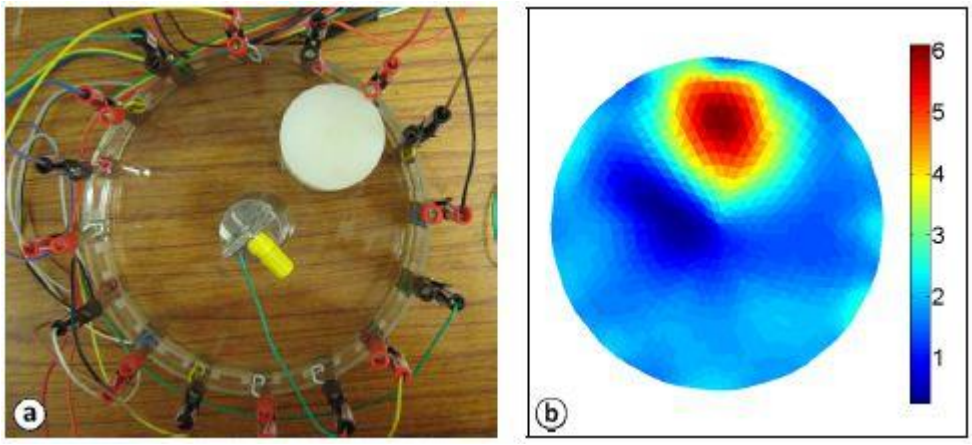

Figure 2 practical example of an EIT electrode model and the inverse results [8]

2.2 Tomography in Matlab

This thesis experiments with EIT in EIDORS, short for Electrical Impedance Tomography and Diffuse Optical Tomography Reconstruction Software. It's a free library filled with software algorithms for forward and inverse modelling for EIT and diffusion based optical tomography [9].
The settings and algorithms experimented with mainly include the injection pattern, the measurement pattern and the electrode placement, to see which influence they have on the data. These experiments are purely theoretic and simulated and differ from reality.

EIDORS works with two important structures, called the forward and inverse model.
The inverse model contains the data calculated by the forward model, or obtained by measurements, in the field 'elem_data'. It also contains the forward model to see the structure of the electrode model and the patterns used as injection and measurement. With both fields, the inverse data or conductivity/resistivity, can be calculated and displayed.
The forward model contains the structure of the measurement setup. The electrode model is defined by the node field under the electrode parameter, while the stimulation contains the stimulation- and measurement patterns. These three fields, or parameters have a big influence on the data, this will be shown in the following subchapters.

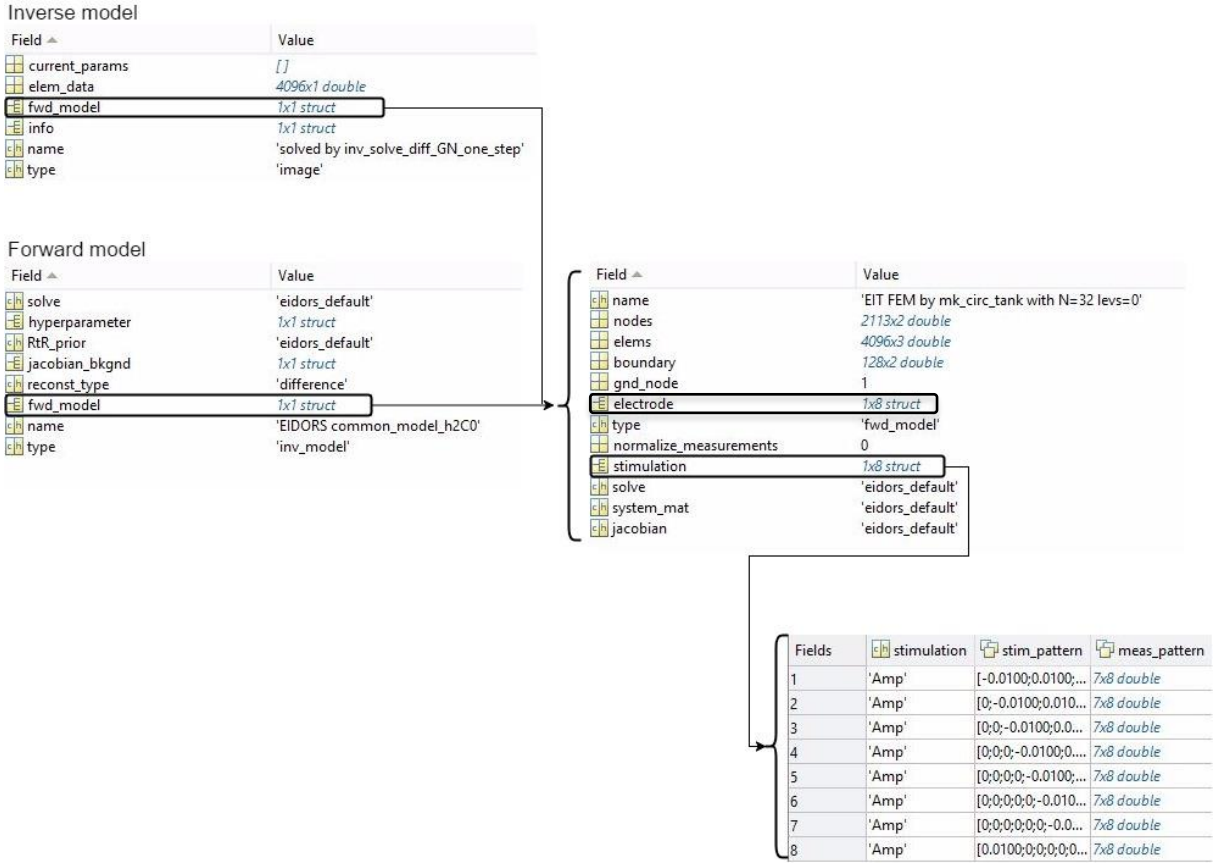

Figure 3 data structures in EIDORS: forward and inverse model

2.2.1 Influence of the injection pattern

The stimulation or injection pattern determines in which order the electrodes will send their current. The size of this field is always N x 1 in Matlab, and N contains the number of electrodes used in the model. In Figure 3, you can see that the stimulation parameter of the forward model exists out of 8 fields. The number of fields equals the number of electrodes used in this model. This means that we used 8 different injection patters, with each of them having a corresponding measurement pattern.

In the practical injection pattern, each electrode will be stimulated at a time. The image below shows such a stimulation pattern for 8 electrodes.

-0.0100	0	0	0	0	0	0	0.0100
0.0100	-0.0100	0	0	0	0	0	0
0	0.0100	-0.0100	0	0	0	0	0
0	0	0.0100	-0.0100	0	0	0	0
0	0	0	0.0100	-0.0100	0	0	0
0	0	0	0	0.0100	-0.0100	0	0
0	0	0	0	0	0.0100	-0.0100	0
0	0	0	0	0	0	0.0100	-0.0100

Figure 4: Injection pattern used in the practical setup

The next injection patterns are purely theoretic and the data will be simulated. If your data isn't simulated, then you should respect the patterns made in the practical exercise. All the examples in this subchapter and the others, are based on the same wound. This is done to make a more relevant comparison. Following image will visualise this wound.

Figure 5: (GRAPHIC) Wound used for making comparisons [10]

The image below represents two injection patterns. In the first injection pattern, the current flows between the first and last electrode, maximizing the area covered by the current, then we move on to the second electrode and the one before the last one. This will continue until 40 injections have been completed. The second one has a more structured pattern, it lets current flow between two neighbouring electrodes. First between electrode one and two, afterwards between two and three, and so on. This is also the default injection pattern in EIDORS. The best results are achieved by the first injection pattern, this is most likely due to the bigger area covered by the current, since more electrodes can measure a voltage now. The measurement patterns are the default patterns in EIDORS, every electrode gets measured except the electrodes who causes the current to flow.

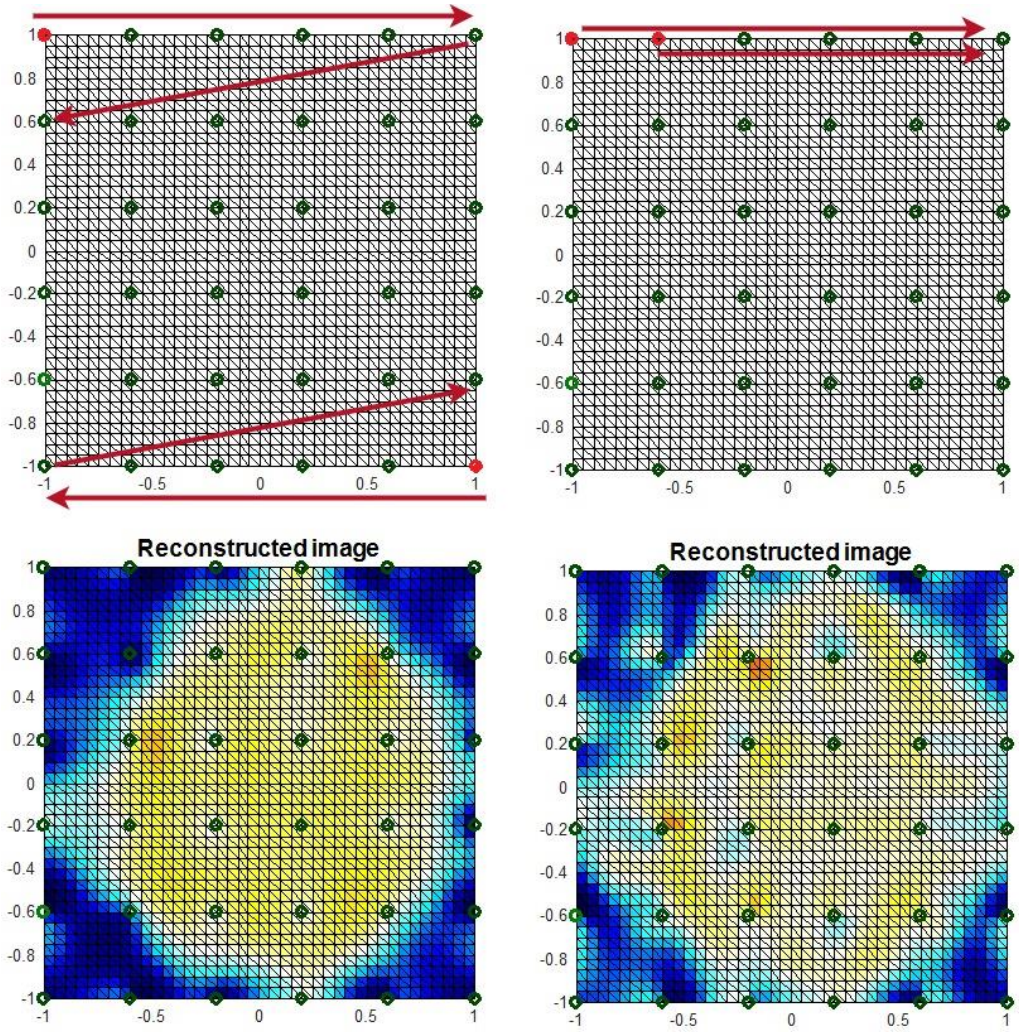

Figure 6: Comparison of injection patterns based on the inverse tomography results

2.2.2 Influence of the measurement pattern

The measurement pattern determines in which order the electrodes will be read. There is a measurement matrix for each injection, this matrix has a M x N size, and N contains the number of electrodes, while M is the amount of measurements for each injection.
In the practical measurement pattern, each electrode Is measured at a time, except the electrodes that received the injection current.

-0.0100	0	1	-1	0	0	0	0	0
0.0100	0	0	1	-1	0	0	0	0
0	0	0	0	1	-1	0	0	0
0	0	0	0	0	1	-1	0	0
0	0	0	0	0	0	1	-1	0
0	0	0	0	0	0	0	1	-1
0	-1	0	0	0	0	0	0	1
0								

Figure 7: injection and the corresponding stimulation pattern

The next measurements patterns are purely theoretic on simulated data. If your data isn't simulated, then you should respect the patterns made in the practical exercise.

On the next image, you can see three measurement patterns, the first one is the same as in Figure 7, it measures every electrode but not between the electrodes that inject the current flow. The second one only reads in the neighbouring electrodes, coloured in blue [11]. The last one is the default measurement pattern of EIDORS and is almost the same pattern as the first one. The difference is that it will not measure between an injection electrode and an inactive electrode. This means that the first and last row from Figure 6Figure 7 will be removed. It's very clear that the default measurement pattern in EIDORS achieves the best results.

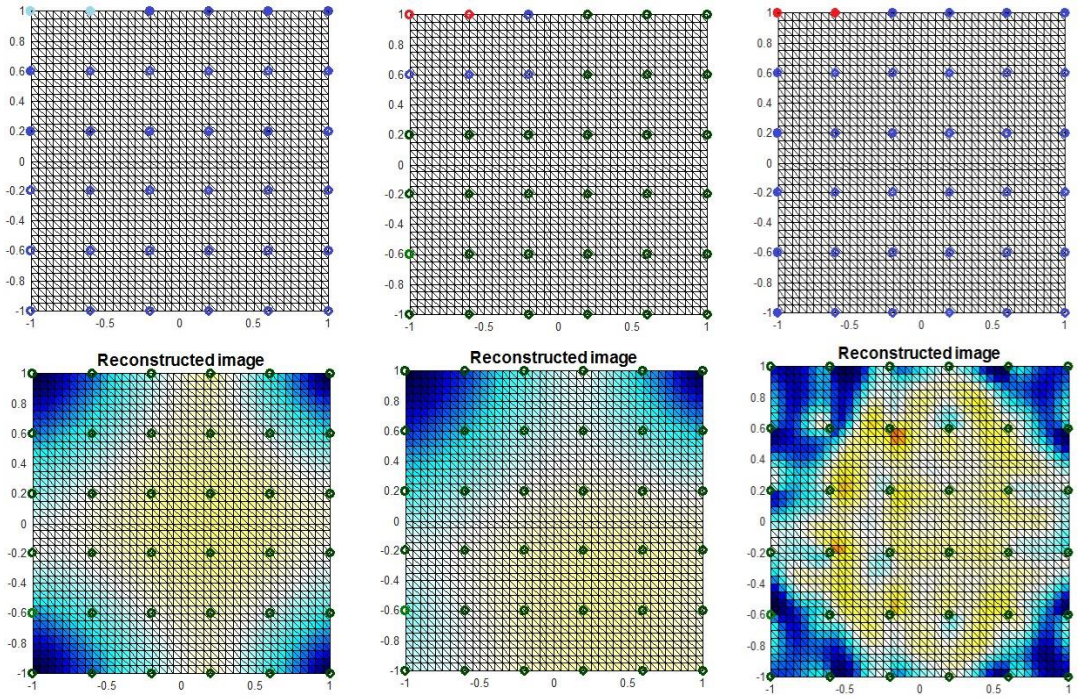

Figure 8: Comparison of measurement patterns based on the inverse tomography results

2.2.3 Influence of the electrode model

The electrode model determines in which position the electrodes are placed. Very common, and pre-coded in EIDORS, are circular and rectangular models, you can find them in various papers by just searching for "EIT". Figure 2 gives an example of such a circular model. When you want to create other 2D models, for example a model that could fit around the surface of a wound, you'll have to code and change the values of the node field from the electrode parameter mentioned in the explanation of Figure 3. In practical applications, a circular model has been used. In the theoretic approach, a rectangular one has been used, and further modified.

It has already been explained how you can modify the electrode position of a rectangular model, but what isn't explained yet, is how you do it to match your desires. The electrode position depends on a single value and not on a X- and Y value as one might think. The image below will visualize this.
On the right, you can see something that would insinuate a X- and Y- value, but on the left, you see a single value for each node. Electrode 6 and 5 are positioned at position 11 and 31, this is the horizontal direction. If you consider the vertical line standing at -1 to be position 1, then you can count and see that the electrode on location -0.5 is positioned at location 11, count further and you'll see that electrode 5 is located at position 31. Electrode 4 is positioned on the second line, and has location number 42. Instead of indicating the position in two dimensions, the locations continue to increase and when it has reached the end of the grid, it moves a line up. To following formula helps to determine an electrode model out of a 2D plot:

$$node(i,1) = y(i) * gridsize + (x(i) + 1)$$

For example: Electrode 4 would be at location (**0**,**1**) -> **1** x 41 + (**0** + 1) = 42.

Fields	nodes	z_contact
1	1651	1.0000e-03
2	1671	1.0000e-03
3	1271	1.0000e-03
4	42	1.0000e-03
5	31	1.0000e-03
6	11	1.0000e-03
7	411	1.0000e-03
8	1231	1.0000e-03

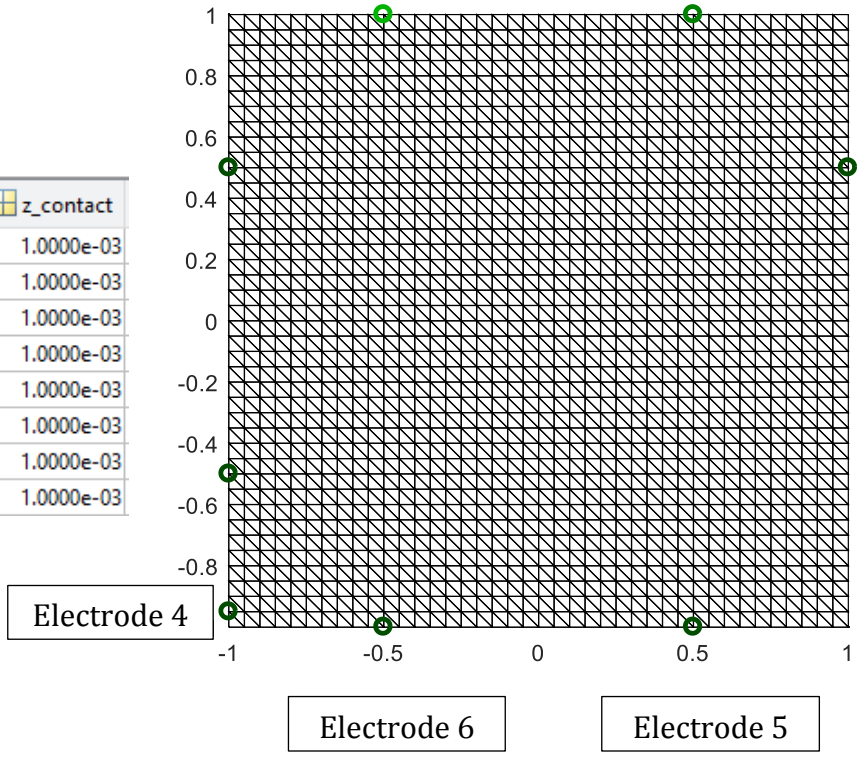

Figure 9: Electrode nodes and their corresponding position

Now that everything about electrode positioning has been explained, let us move on to a more realistic, yet purely theoretic, example: an electrode model around the surface of a wound. First the wound gets extracted out of an image (more about this in 5.2 Wound segmentation based on colour with the K-means algorithm, and the following chapters), afterwards the program will calculate the boundaries of this wound and place electrodes around it. Finally, the positions of the electrodes will be calculated, by using the formula as described before. The results are the following.

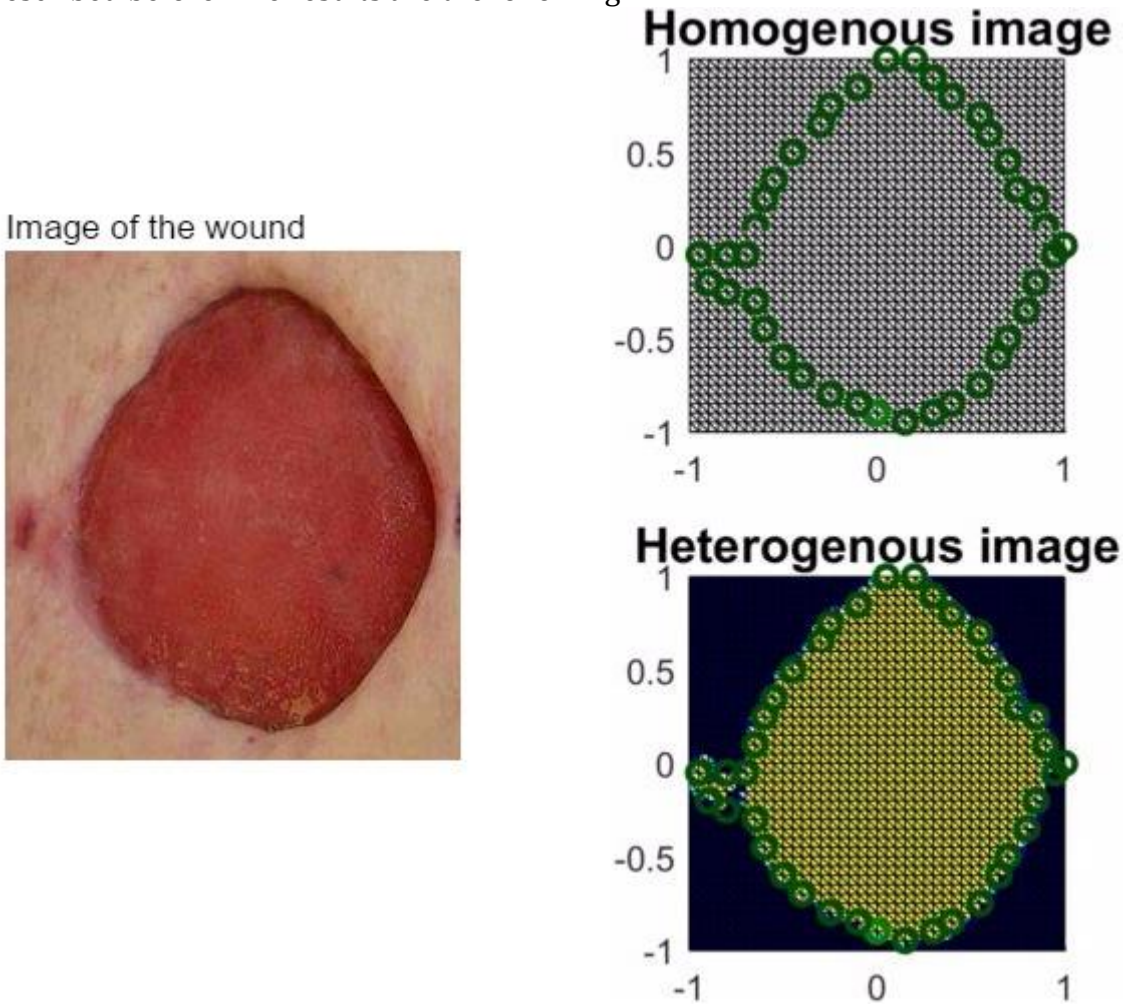

Figure 10: (GRAPHIC) Image of a wound and its forward model with electrodes around the surface of the wound

Simulating such a model isn't hard, but realising this in real would be inefficient. Every wound has a different size, which means that the electrode model will change for every wound, it would be better to create a more general model.

The next image will show the influence of an electrode model on the conductivity or resistivity plot. First a default rectangular model will be simulated. Next is a model that fits around the surface of a wound and finally you can see a wound located in an electrode grid. The simulated conductivity is based on the intensity, or grayscale values, of the wound. The reconstructed images, or conductivity/resistivity data, are clearly different from each other. At a first glance, one should notice that the first two reconstructed images present the best results. The second image beats the first one because of the separation between the two yellow wounds, this creates a perfect representation of the wound and could be used, theoretically, to monitor the wound.

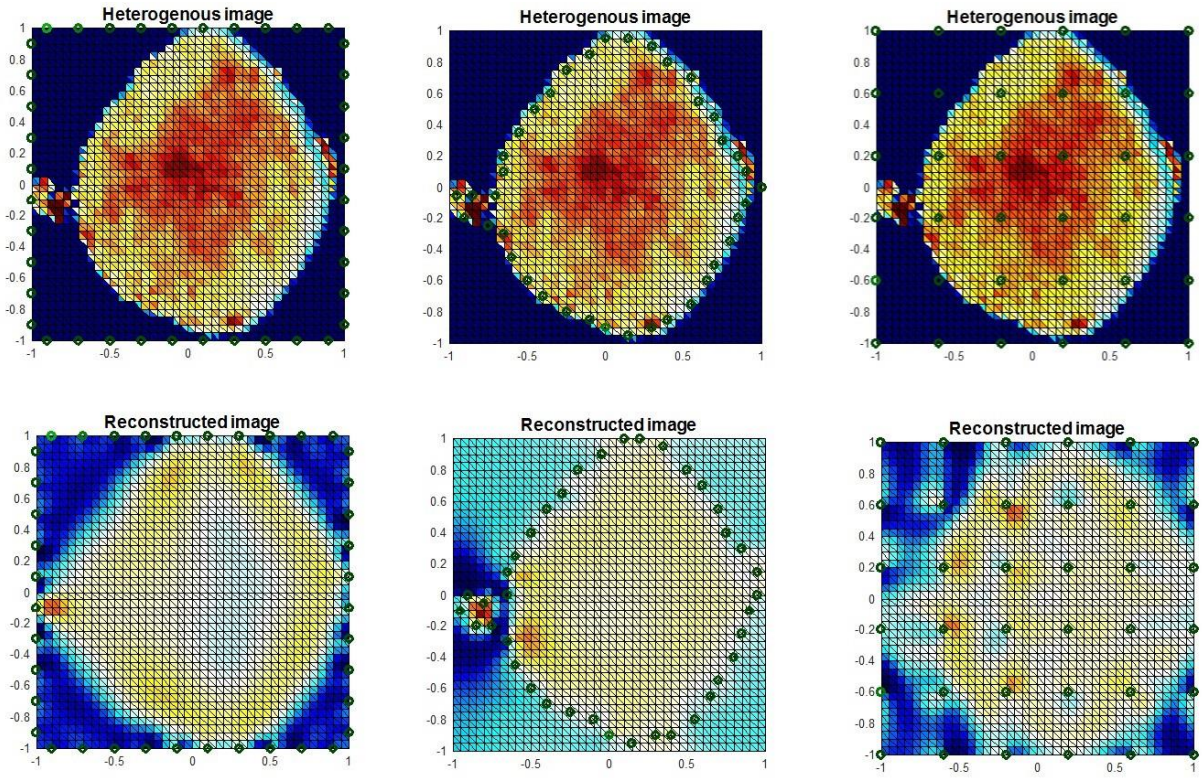

Figure 11: Comparison of electrode models based on the inverse tomography results

2.3 Simulated wound detection with EIDORS

The goal in this chapter is to use one of the previous obtained models on an image with and without a wound, to compare the inverted data with each other. The following injection pattern have been chosen in combination with the default EIDORS measurement pattern.

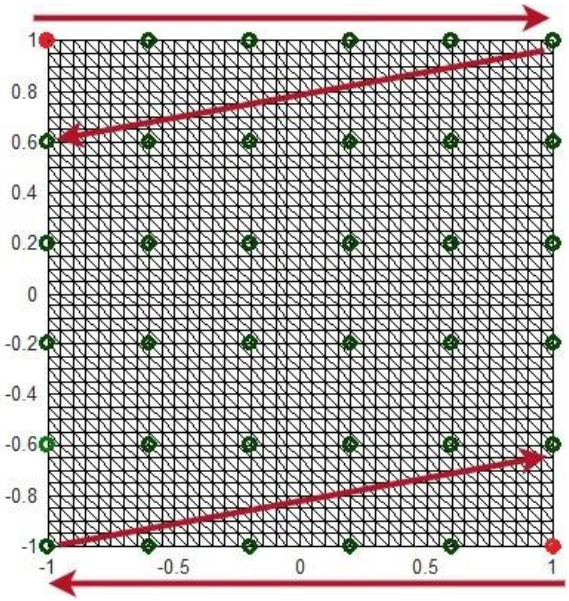

Figure 12: Injection pattern for the forward model

Next image will show the homogenous and heterogenous image of the forward model, as well as the inverted data. The image of the actual wound has been displayed in Figure 5.

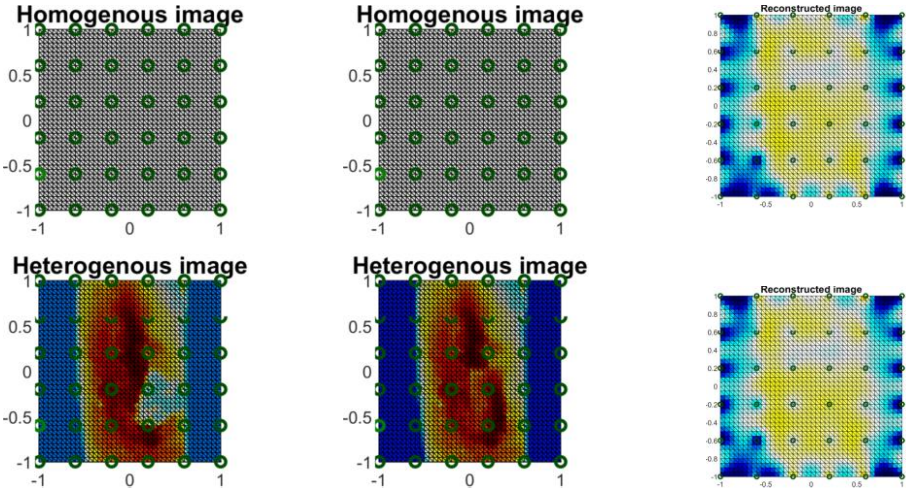

Figure 13: forward- and inverse data of an image with and without wound

The inverse data doesn't seem to make any sense at all, since the wound is not visible. However, when we subtract both images from each other, the results become more promising.

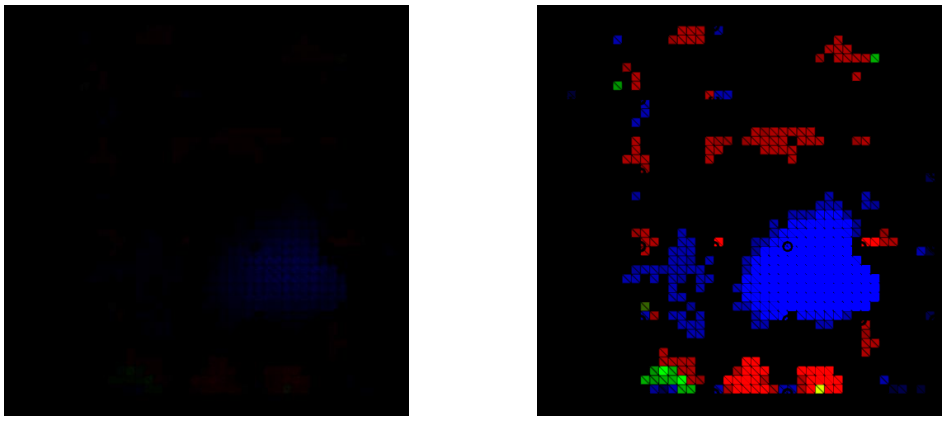

Figure 14: Subtraction of the inverse data with and without wound, normal and with an amplification of 25

The left image is the normal subtraction, here you can already see the wound in blue, when the values of the image get amplified with 25, the blue of the wound becomes more intense and some noise seems to appear. This noise is most likely caused by bad editing skills. The image without a wound, is an attempt to remove the wound from the original image, this removing was done in an amateurish way by myself and therefore causes noise. The following image is graphic, but will explain the problem visually.

Figure 15: (GRAPHIC) Image with and without wound

3 Pre-processing of the data

Pre-processing data will always be an import step in machine learning. The data you receive can become out of range, or missing due to measuring faults which makes it confusing and disorganised [12], this is why you need to pre-process the data, for everything to make sense. This pre-processing can be done in many ways (3.1, 3.2, 3.3), and heavily depend on the project you're working on and what your final goal is. This chapter will briefly explain what pre-processing is, to make you realise how important it is.

3.1 Data selection

In this step, you need to wonder what data is necessary to realise your goal. To put this into an example, if you observe human beings and want to create artificial intelligence to automatically detect signs of cancer, or other diseases, it is safe to assume that the colour of that person's eyes has nothing to do with the disease. Therefore, you can eliminate, or simply not include this data. Relevant data could for example be the size of rashes on the skin, and the colour of those rashes. Sometimes this step can be skipped, for example when you're already equipped with the right data from the start. Once you've gathered all the relevant databases, you're ready to move to step two [13].

3.2 Data pre-processing

In this step, you take a closer look at the data and manipulate it. A first step is to reformat the data, for example from a .txt file to a .np file, to later work with the data in python. Afterwards you need to correct the available data, does it contain missing values, or perhaps impossible data, then this must be corrected. This step is called data cleaning. Finally, you need to implement feature extraction and – selection. Feature extraction means that you are still missing relevant data, and that you'll need to calculate this data with the help of other features. While feature selection means that some features in the database are irrelevant and need to be removed. Once the data has been processed, you can move on to the last step.

3.3 Data Transformation

The relevant databases have already been selected, and the right features are extracted but sometimes this isn't enough. Some variables can be unnecessary complex and need to be decomposed. To give an example, when you want to predict on which day of the month it rains the most, only the days are relevant, not the whole date. Another problem with data is that it can be incompatible with the machine learning algorithm you want to use. Machine learning algorithms require numeric data [14], while your data might be text. A way to solve this is to convert the data to a number, for example: "cat, dog, lion" becomes "1,2,3". This might be confusing for some because it insinuates a ranking, in this case you can use "One-hot encode", this maps a unique code generated by 1's and 0's to your labels so it becomes "001,010,100" [15]. The very last step in this chapter will be vectorising every feature. This needs to be done so the algorithms know which data belongs to which label.

Once the data has been optimized, it's ready to be used. This data will go through algorithms which on their turn classify the data, group them or calculate other parameters. The most popular algorithms that are available and relevant to this project, and a more advanced topic, will be discussed in this chapter.

4 Regression algorithms

Regression algorithms are algorithms that predict a value of a parameter based on another measured parameter. There are many forms of regression, but this chapter will not discuss them all since they are all based on the same principle.

This regression model must be trained if you want to further use it in machine learning. A model should be calculated out of the given data points. This model will try to approach the given data points as close as possible. The more given data points, the better the accuracy of this model will be. Once the model has been trained, it'll be used to predict the required parameter by applying the model on the data. The most used regression algorithms are linear and logarithmic regression.

4.1 Linear regression

The linear regression method is based on the equation of a straight line: $y = m * x + b$. This formula calculates y while x is the given variable parameter and the others are constants. Although there are more forms of linear regression, this thesis will only visualise the ordinary least squares regression. This method measures the distance between the regression line and each data point, squares it and tries to minimize it, following image is an example of a solution obtained by this method [16].

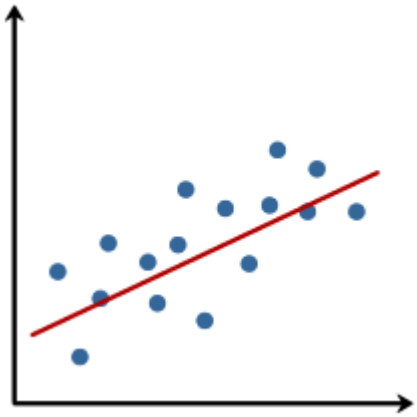

Figure 16: Linear regression, ordinary least squares method [17]

4.2 Logarithmic regression

Logistic regression is a form of binary classification, it calculates the probability that a certain data point belongs to either 1 or 0. When it reaches over a certain threshold, it'll be a 1 [18]. This kind of regression is done with a sigmoid curve, which has the following formula and curve. The X-axis is the input, while the Y-axis is the output.

$$y = \frac{1}{1 + e^{-x}}$$

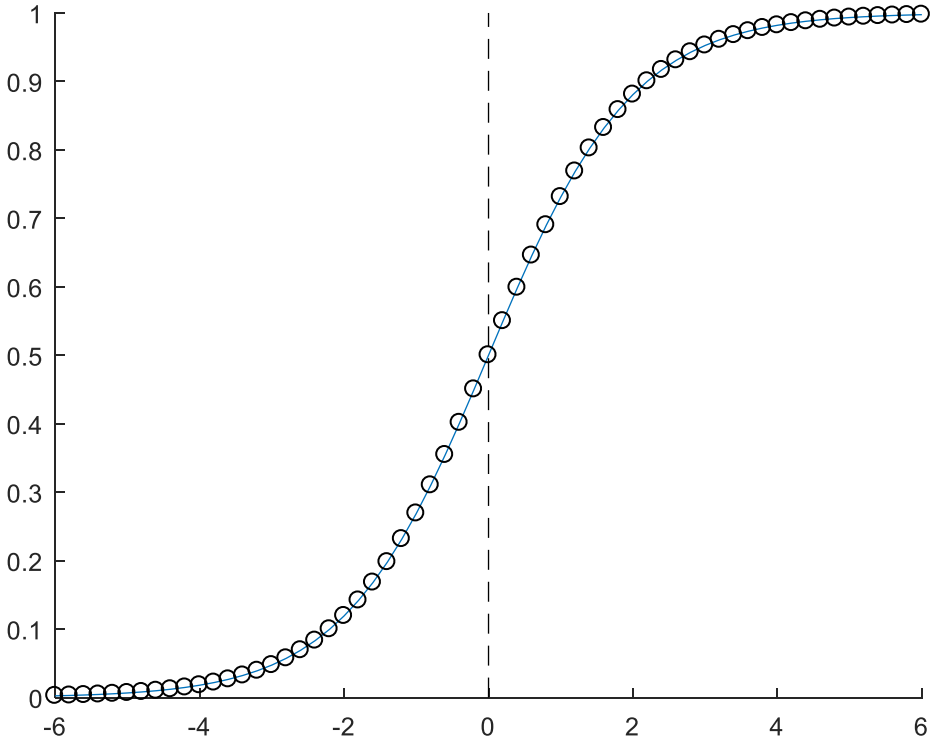

Figure 17: Logarithmic regression, sigmoid curve

4.3 Regression on a pump application

This subchapter will explain the setup of the application, how the measurement data was gathered and how the regression is implemented, with a small comparison between regression algorithms.

4.3.1 Overview of the application

The task was to do impedance measurements on the change of liquids in pipelines. First the pipelines were filled with orange juice, and afterwards water ran through. The impedance was monitored the whole time, and later visualised. The next image visualises the measurement setup. The impedance measurement is a two-point measurement, as visualised on the image, and is done with an impedance meter from the IMO. The pump is a Williamson pump from the 100 series [19].

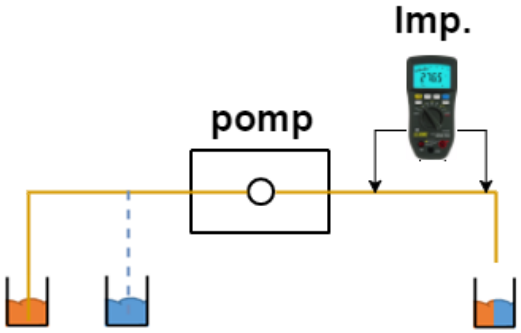

Figure 18: Pump application, measurement setup

The measurements were afterwards loaded into excel to visualise and control the data, but also to get a better idea how to implement regression. The following graph will show the impedance magnitude in function of the time that has past, or also considered as the number of measurements.

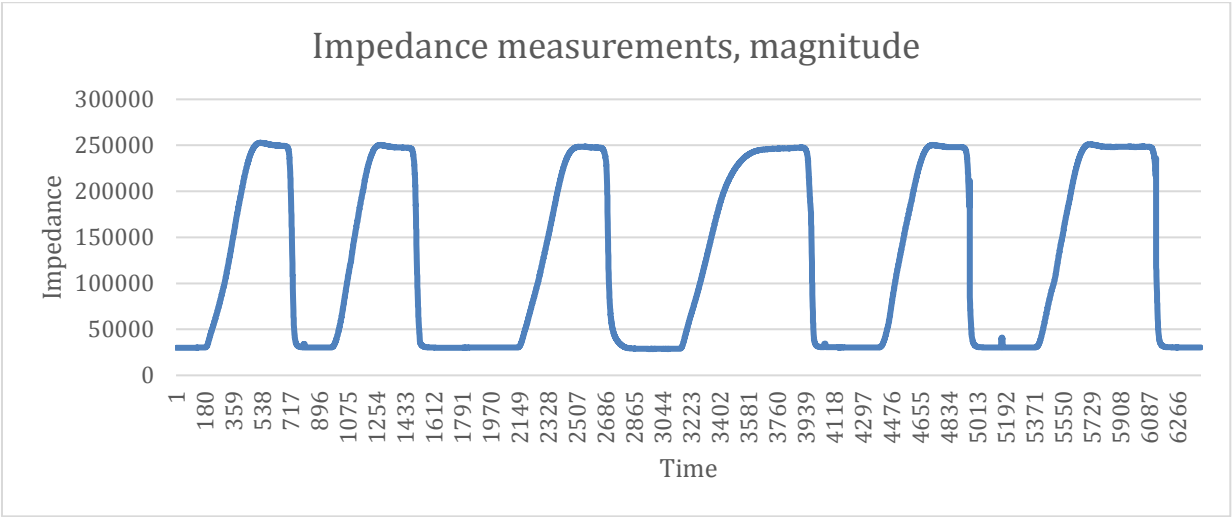

Figure 19: Pump application, impedance magnitude measurements in function of the time

4.3.2 Regression models of the concentration

The first regression that has been implemented, is the regression where the concentration of the liquid is modelled in function of its impedance. 100% concentration means that there's only orange juice flowing, 0% is pure water. The next image shows a power regression implemented by Matlab. This form of regression follows the following formula.

$$y = a * x^b + c$$

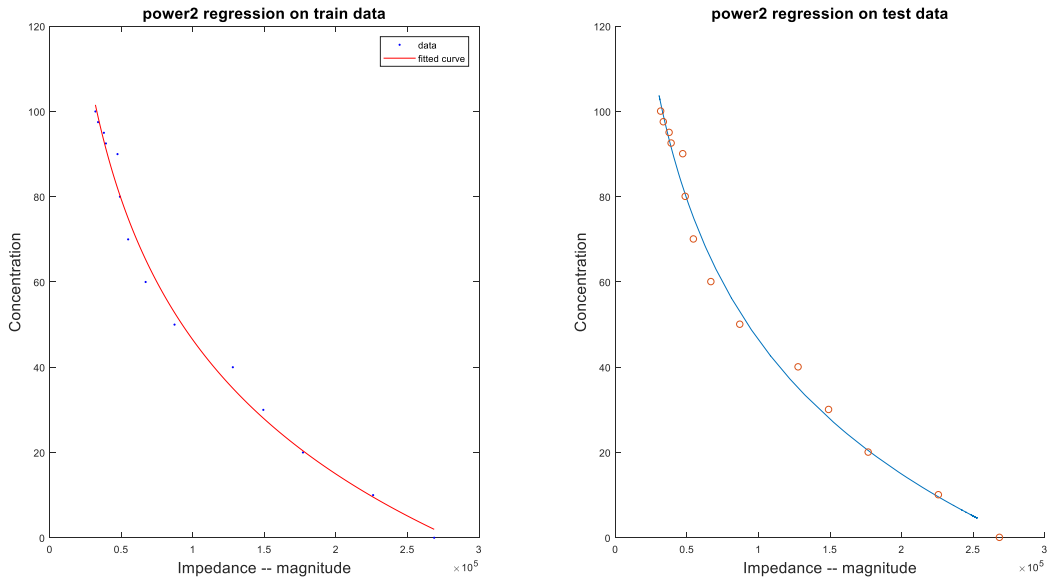

Figure 20: power regression to model the concentration in function of the impedance

The left graph is the trainings data, the model in its most optimal state. The right represents a scatter plot of some test data, together with the predictions of the model. It's clear that the concentration, both in training and test phase does not reach 0% when only water flows through the pipeline. The minimum concentration of the test data will be around 4%, while the maximum concentration is around 106%. The R-square value is 0.9895, the R-square value is a coefficient that is based on the variance between the data and the regression graph. The maximum value is 1, which indicated a perfect fit, and 0 is the lowest [20]. The results aren't bad, although it might be considered problematic that the regression doesn't go down till 0% concentration, like it should.

The next implemented regression is also realised in Matlab. This regression makes use of a custom logarithmic equation of the form:

$$y = a * \log_{10}(b * x) + c$$

The b coefficient is always positive, while the a must be negative to obtain the desired fit from a logarithmic equation. The next image will show this regression.

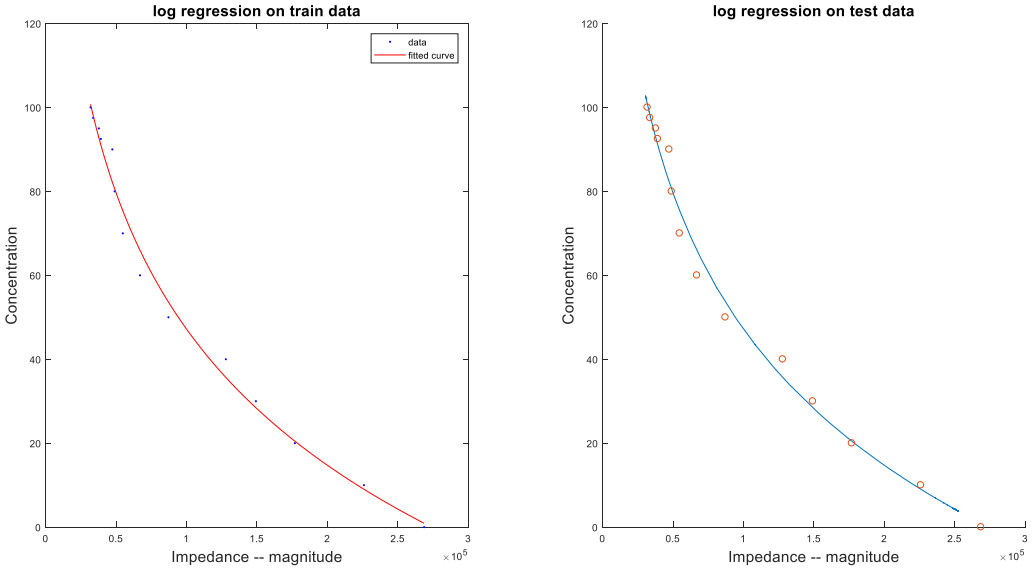

Figure 21: logarithmic regression to model the concentration in function of the impedance

The left image is the training data, which means that the model is in its optimal state. The right represents the same a scatter plot as before, together with the predictions of the model. The fit appears to be the same as made before, but this can't be entirely true. The minimum concentration on the test data is around 3%, while the maximum is around 105%. Although this might indicate a better fit, the values in between also need to match. The R-square value is 0.9892, which is slightly worse than the power regression. Next graph shows both regressions in comparison to each other.

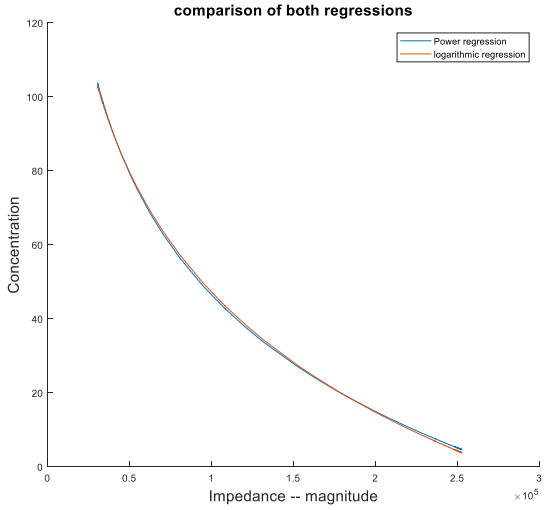

Figure 22: Comparison between the power- and the logarithmic regression

The previous regression results were based on one complete measurement between 100%- and 0% concentration. The next image will show the results of both regressions on continuous data. The liquids are constantly changed. You can see that the curve changes moderately when the liquid changes from orange juice to water, but the opposite results in a steep graph. This is because orange juice is heavier than water, which means that it doesn't mix with water, but instead pushes it through the pipelines. The change from orange juice to water is more moderately because the water has trouble pushing the orange juice through the pipeline, causing it to mix. The impedance of orange juice is around the 32kΩ, while the impedance of water is around 250kΩ. The graph also shows that the previous statements about the regressions are true, the concentration doesn't reach 0% in both cases, and goes higher than 100%.

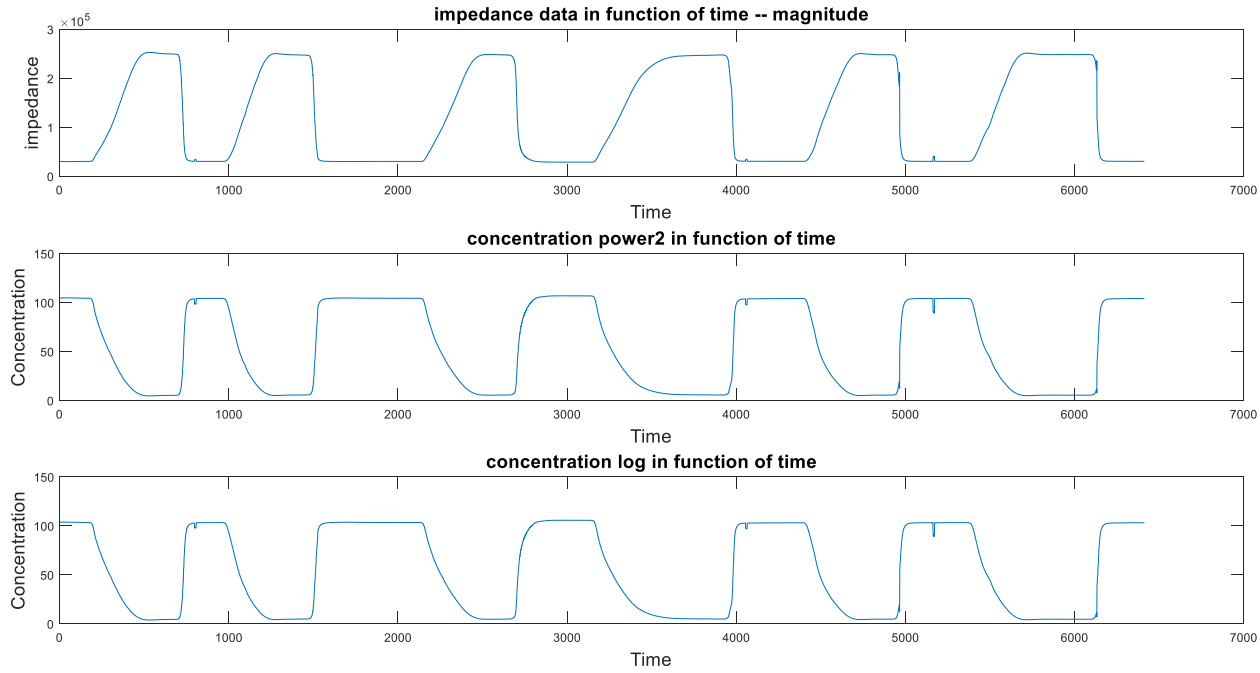

Figure 23: Both the power- and the logarithmic regression on continuous impedance data

The regression implementations in Matlab are good, based on the graphs and the R-square values. There is however still room left for improvement. The next regression model will try to model the same data, but in this case, no formula will be used. Instead, a support vector regression will be implemented, with a RBF-kernel and the parameters will be decided by doing a grid search to compare different gamma and C values with each other. To learn more about the kernels and their parameters, please consult yourself with chapter 7 Machine learning kernels.

The following image shows the SVR-regression. A first glance indicates that the fit does not fully reach 100%, however, it does seem to reach 0% concentration. The R-square value is 0.9877, which indicated that the fit is comparable to the power model.

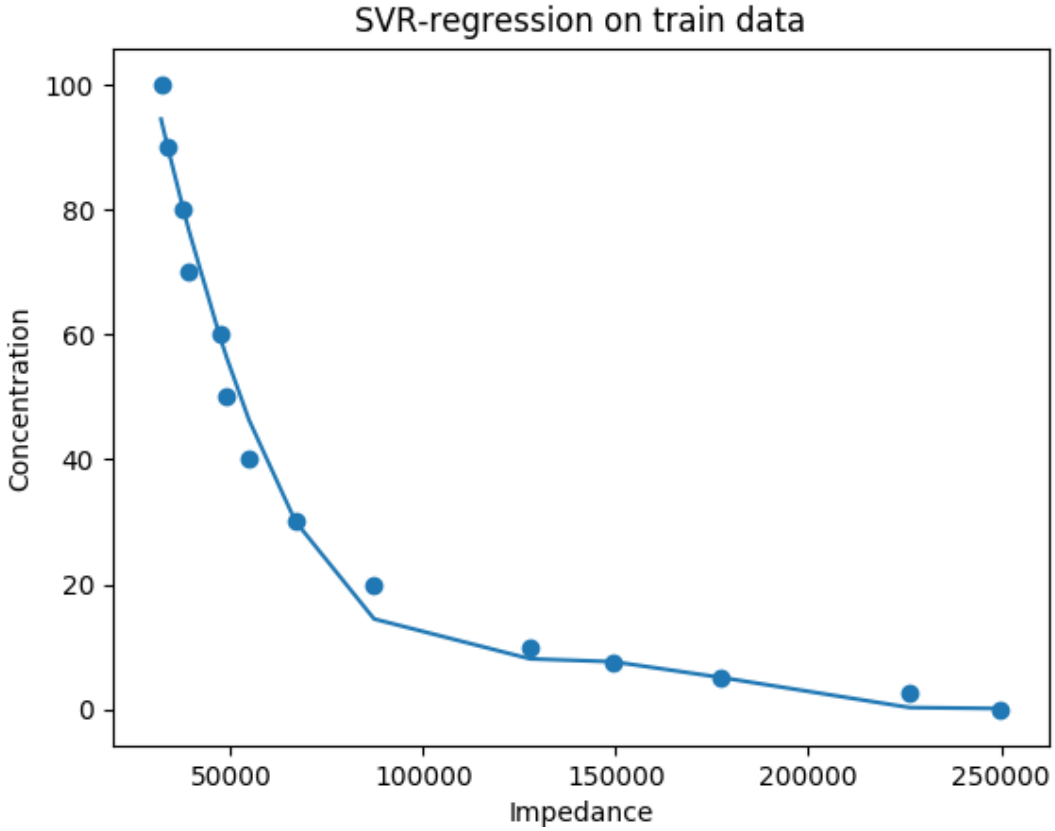

Figure 24: SVR-regression to model the concentration in function of the impedance

The next graph will show the achieved regression on the same continues data used before. The orange line indicates 100% concentration, while the green line stands for 0% concentration. It's clearly visible that the regression fit reaches 100%, and only surpasses this limit once. The 0% concentration is never surpassed and reached in every cycle. This indicates that this regression model is the most optimal for this application.

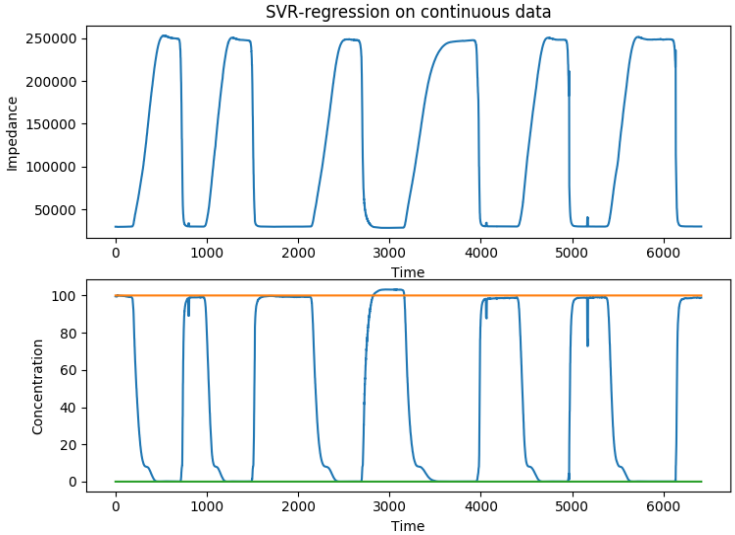

Figure 25: SVR-regression on continuous impedance data, determining concentration

4.3.3 Regression models of the required time

The next regression that has been implemented, is the regression where we model the time till only water remains in function of its impedance. The next image shows an exponential regression implemented by Matlab. This form of regression follows the following formula.

$$y = a * e^{b*x} + c * e^{d*x}$$

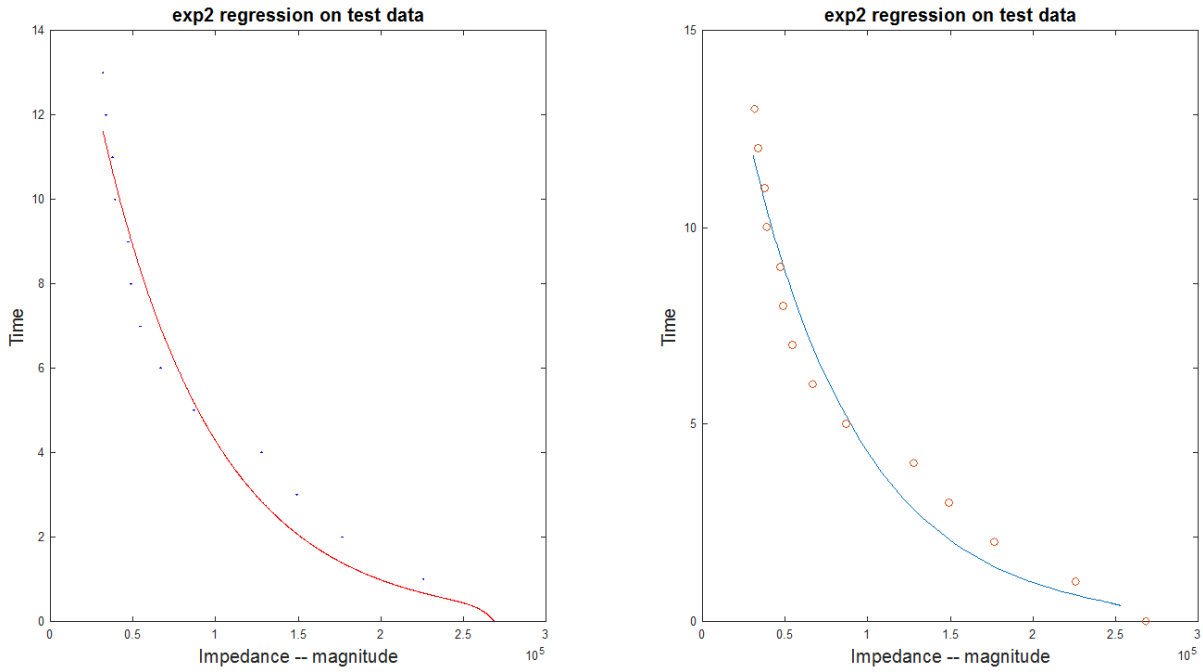

Figure 26: exponential regression to model the time left in function of the impedance

The left graph is the trainings data, the model in its most optimal state. The right represents a scatter plot of some test data, together with the predictions of the model. The fit towards the data is bad, although it reaches 0 seconds on the training graph, it doesn't on the test graph. The highest indications of time aren't reached, not in the training- nor test graph. The R-square value is 0.9597, these results aren't horrifically bad, although they need to be improved a lot to have an accurate representation.

The next implemented regression makes use of a custom logarithmic equation, as seen in the previous chapter, and has the following form:

$$y = a * \log_{10}(b * x) + c$$

The b coefficient is always positive, while the a must be negative to obtain the desired fit from a logarithmic equation. The next image will show this regression.

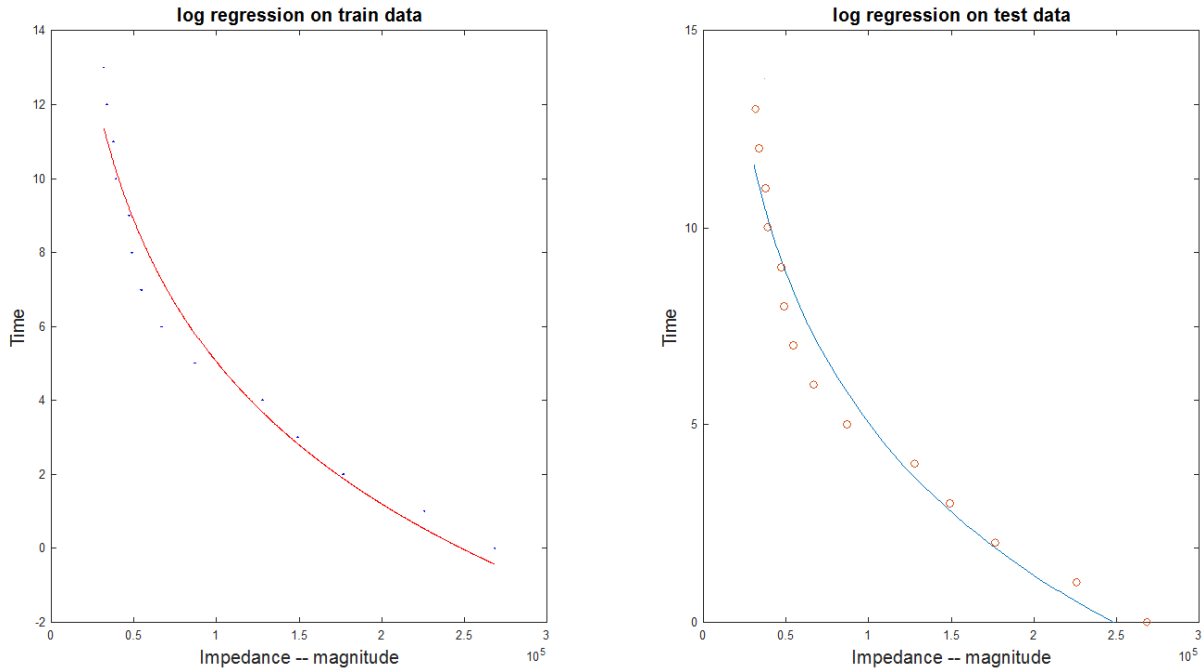

Figure 27: logarithmic regression to model the time left in function of the impedance

The left image is the training data, which means that the model is in its optimal state. The right represents the same a scatter plot as before, together with the predictions of the model. The fit looks similar, since it's also a bad representation of the data. The R-square value is 0.9577, which is slightly worse than the exponential regression. Next graph shows both regressions in comparison to each other.

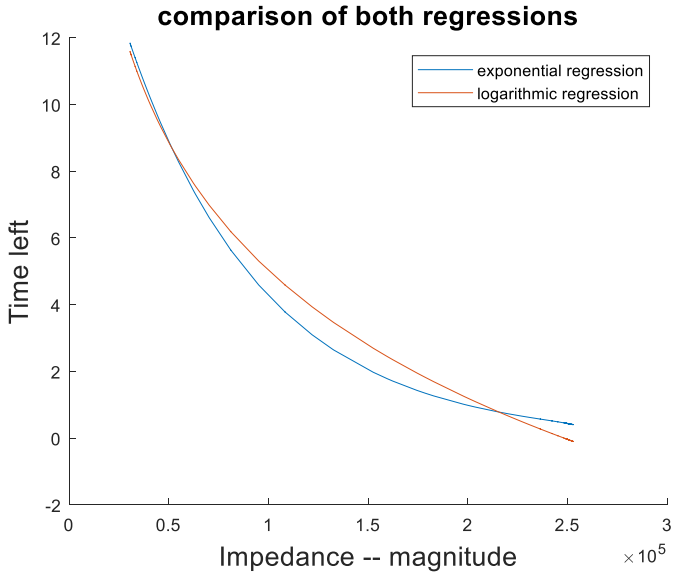

Figure 28: Comparison between the exponential- and the logarithmic regression

The previous regression results were based on one complete measurement. The next image will show the results of both regressions on continuous data. The liquids are, just like before, constantly changed. The impedance of orange juice is around the 32kΩ, while the impedance of water is around 250kΩ. The exponential regression graph also show us the gap in time between the minimum predicted value and 0 seconds. The logarithmic regression however does reach 0, which indicates it as a more suitable model for our application.

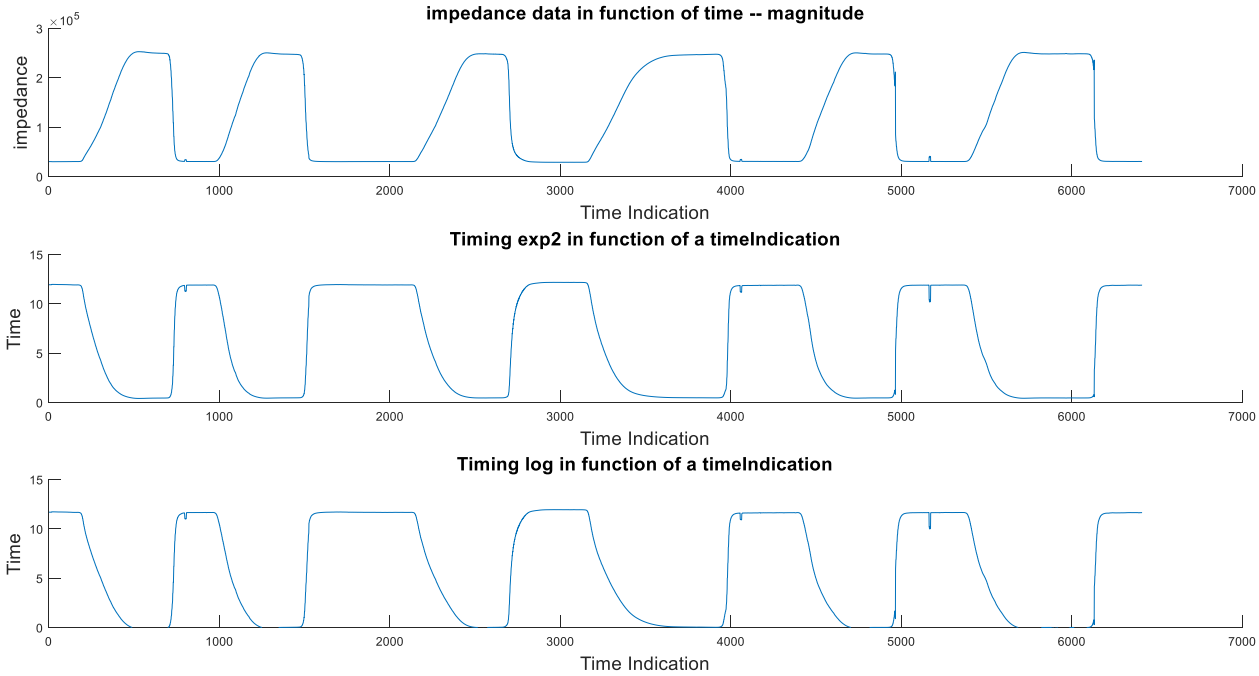

Figure 29: Both the exponential- and the logarithmic regression on continuous impedance data

The regression implementations in Matlab are not that good, based on the graphs and the R-square values. The next regression model will try to model the same data, but in this case, no formula will be used. Instead, a support vector regression will be implemented, with a RBF-kernel and the parameters will be decided by doing a grid search to compare different gamma and C values with each other. To learn more about the kernels and their parameters, please consult yourself with chapter 7 Machine learning kernels.

The following image shows the SVR-regression. The R-square value is 0.9970, which indicated that the fit is nearly perfect, and the best of all models.

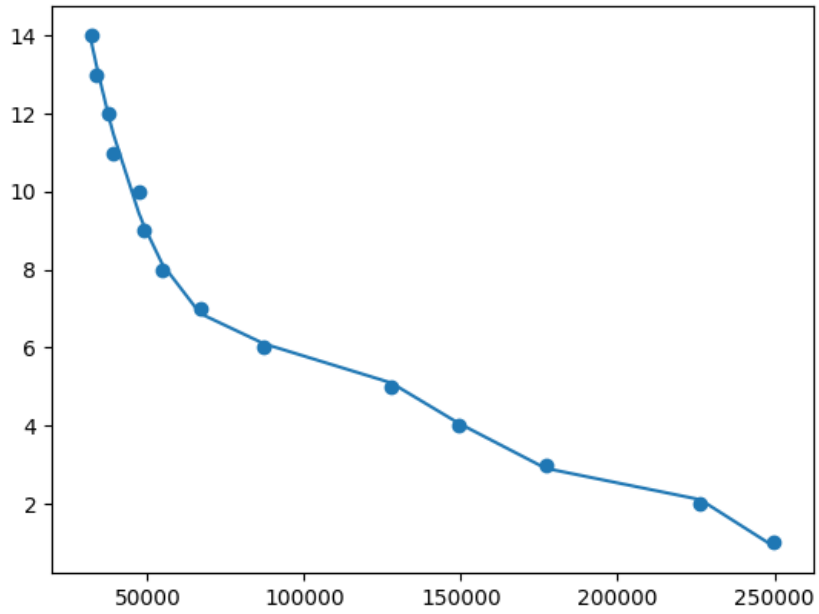

Figure 30: SVR-regression to model the time left in function of the impedance

The next graph will show the achieved regression on the same continues data used before. This indicates that this regression model isn't as optimal as first thought. The yellow line indicates the minimum of the data, which is 0.66, this should be 0.

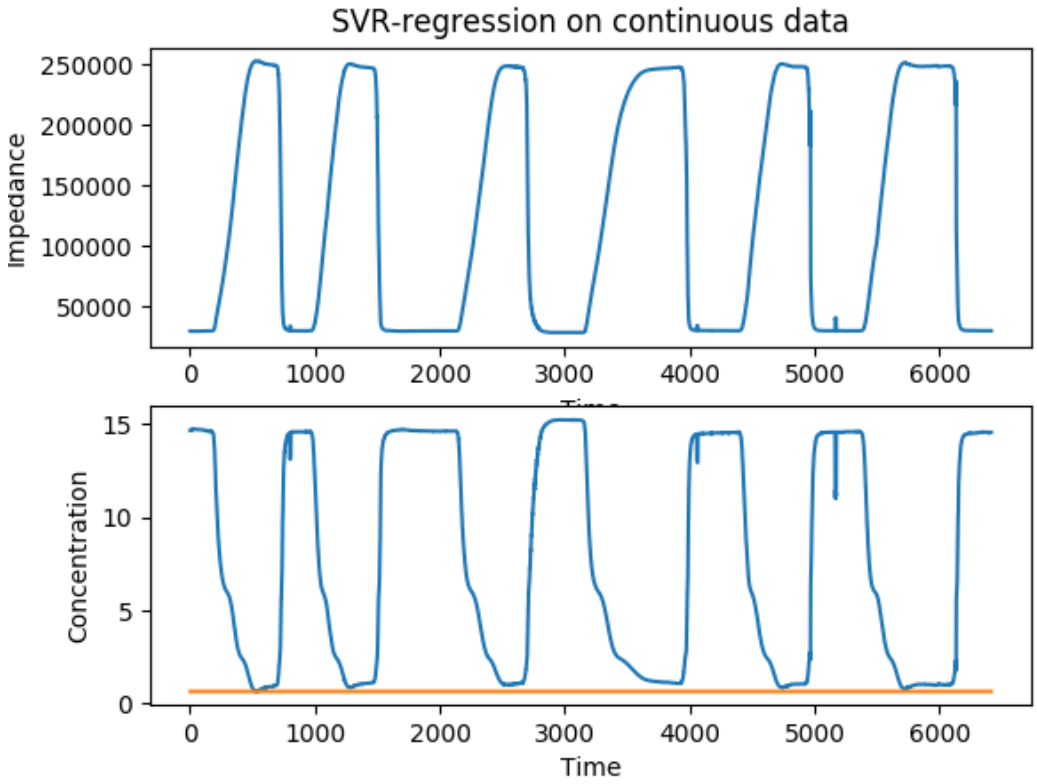

Figure 31: SVR-regression on continuous impedance data, determining the time that is left

4.3.4 Conclusion

There are many ways to implement regression, and not all of them are mentioned in the previous chapters. It's proven that regression can be achieved by models that are normally used for classification purposes, and these results can surpass those of normal equation models. The regression models could be deployed in a real-time application, since the results match reality. To determine the concentration, it's most optimal to use the SVR-regression, while the time left can best be represented with the logarithmic approach.

These experiments have shown that sometimes the regression fits perfectly around the training data, giving a near perfect R-square score, but still fails when tested on other test data.

4.4 Regression on yeast cells

The data used in this chapter, is the resistance data of the growth of yeast cells in a glucose concentration of 2%, for 10g, 15g and 75g of yeast. Three regression formulas were picked and fitted on the data, the 'goodness of fit'-data from these regressions, determined the quantity of yeast we're dealing with. This calculation, was a classification realised with a Support Vector Machine (see chapter 6.2) equipped with a RBF kernel (see chapter 7.2). Following image will visualise the setup.

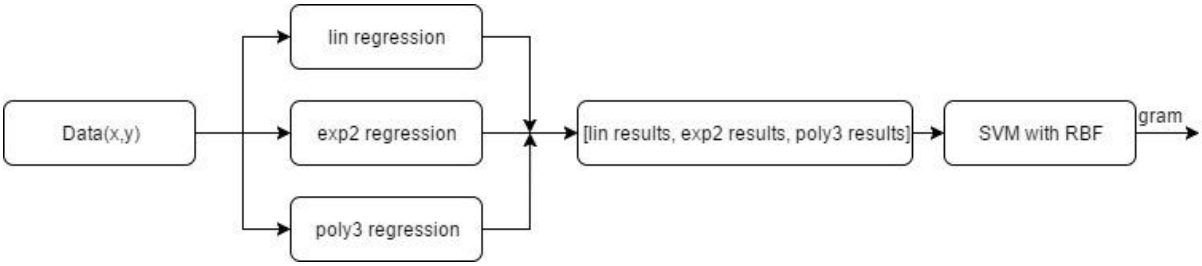

Figure 32: Regression on yeast cells, setup

4.4.1 Preprocessing the data

The original data were 517004x2 tables, this table described the timing and the resistance. Using all these data points as features to classify, is obscure and inefficient. The first step was to check the tables for unique values, so that repetitive values get removed. The results are the following:

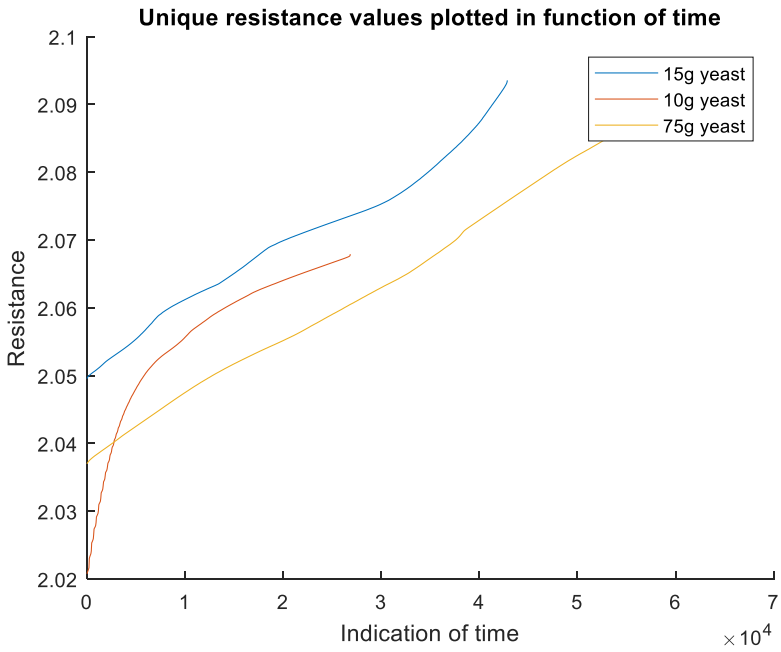

Figure 33: Unique resistance values of yeast cells, in function of time

4.4.2 Implementing regression

Classifying the previously visualised data could be as easy as determining how well a linear regression fits to the data. Instead we fitted an optimal regression for each separate curve. The best fit for the resistance plot of 75g yeast was a simple linear regression. While the curve for the 15g yeast was a polynomial regression of the 3rd grade. At last, the curve for 10g yeast was best fitted with a double exponential regression. The formulas are the following.
3rd degree polynomial regression: $y = a * x^3 + b * x^2 + c * x + d$
Double exponential regression: $y = a * e^{bx} + c * e^{dx}$

The results for each of these regressions are as follows.

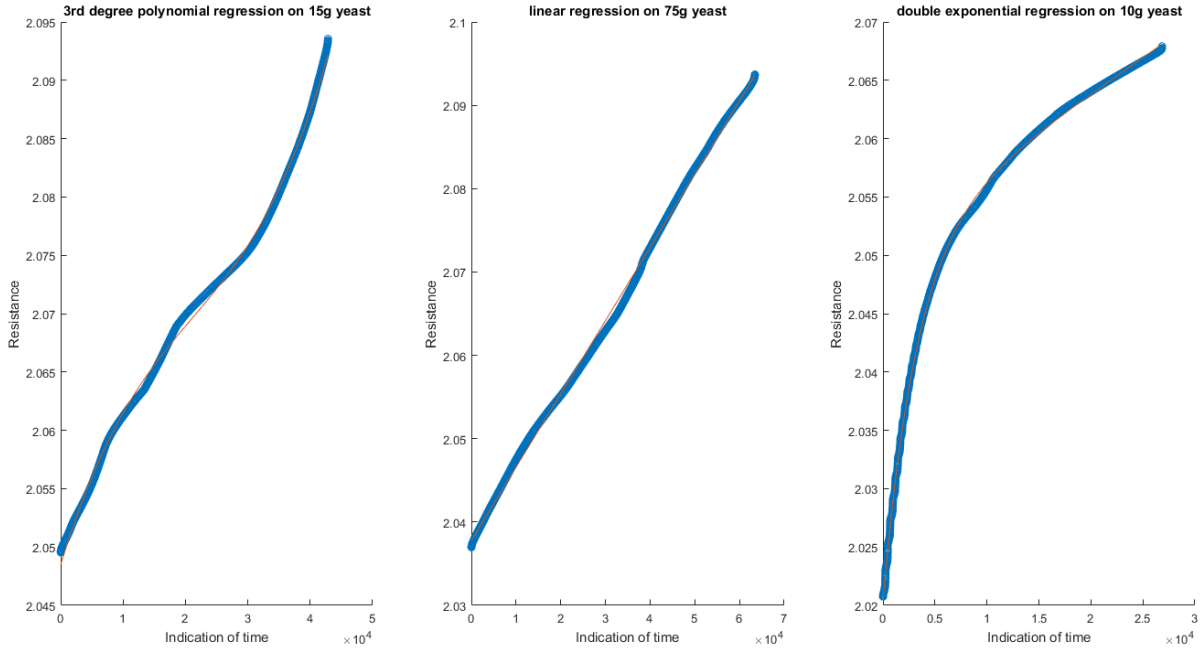

Figure 34: All regressions on different yeast quantities

4.4.3 Implementing the SVM-classifier

Each data has an optimal curve, as shown above. So, if the linear regression of the 75g yeast, would try to fit around the data of the 10g yeast, the results of these fit would be less accurate. The features are made of this principle. The data from the 10g yeast, gets fed to the linear-, polynomial- and double exponential regression, the goodness of these fits are saved in an array and used as features. The optimal classifier to separate the yeast quantities from each other is determined by the classification learner of Matlab, and was a support vector machine with RBF kernel, it reached 100% on the training data, but has not been tested due to a lack of data.

5 Grouping algorithms

5.1 K means algorithm

This algorithm is used to add structure to your data and to group them in K number of groups. The following image will show the algorithm in its full glory for K = 2.

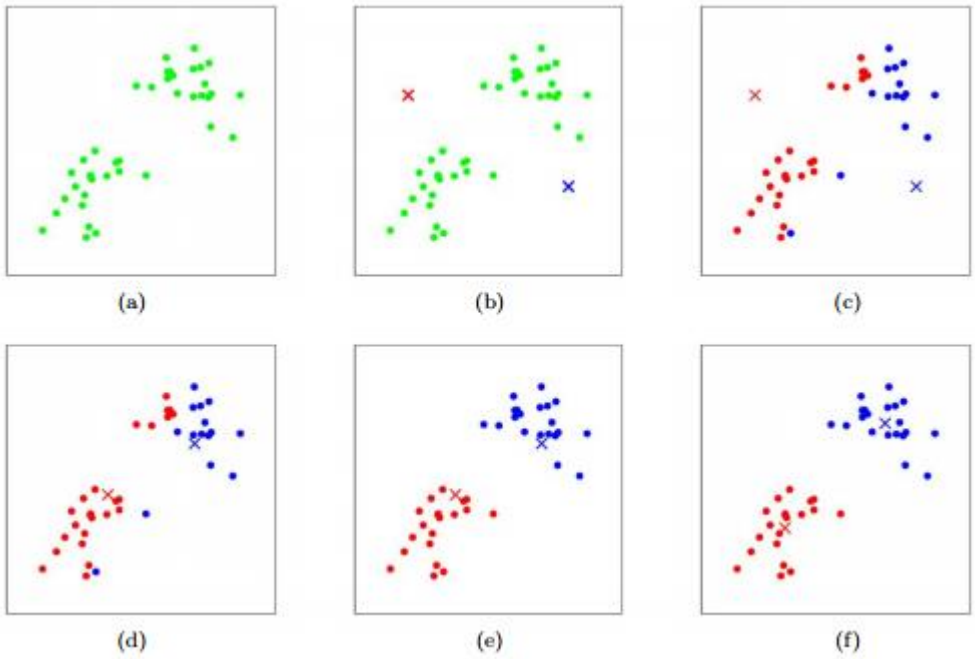

Figure 35: K means algorithm, example [21]

We start off with an ungrouped amount of data. The algorithm will place two random centroids (K = 2) and group every data point by its distance relative to the centroid. If a data point is closer to the red centroid, the data will be labelled as red.
In the next step the algorithm will determine the centre of gravity of the groups and place the centroid in that location.

The data gets regrouped again per the smallest distance relative to the centroids. This procedure will be repeated until the centroids are in the right position and won't move when recalculated. Now we have reached K optimal groups, and like said before K = 2 in this example [22].

5.2 Wound segmentation based on colour with the K-means algorithm

Image segmentation based on colour can be done is different ways. The first method that has been tested was thresholding. Matlab has a nice app for that called: Colour Thresholder. The problem is that the threshold changes with each image, and sometimes even the optimal colour space. It's bad coding to apply a change for every image, there must be a dynamic method that will work automatically on all images, and that method is K-means segmentation. This example explains how to realise wound segmentation on a normal image. The following chart will visualize the principle of this K-means based application.

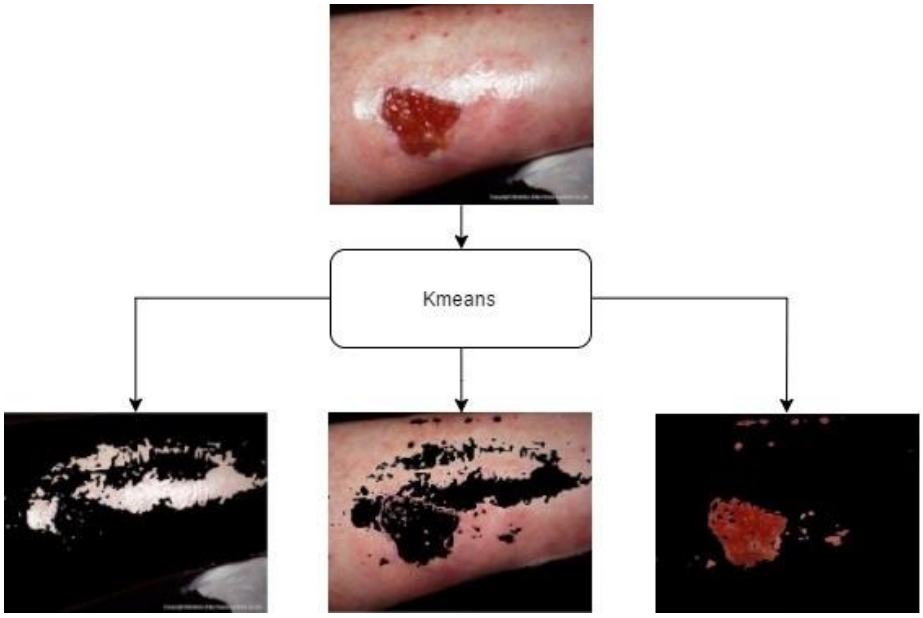

Figure 36: K-means Image segmentation based on colour

The first step is to determine the right colour space, to have an optimal segmentation. Instead of listing all the colour spaces and comparing them, the one that has been used and the mostly used colour space will be compared, before and after the K-means algorithm. Once the colour spaces have been picked, you can use the k-means algorithm to segment the image. The final step is to display the results.

5.2.1 Deciding the optimal colour space

The two colour spaces that will be compared are the RBG- and LAB colour space. The RBG colour space needs little explanation, it exists out of three layers: red, blue and green. These layers added together can represent 16777216 different colours, since every plane can go from 0 to 255, which are 256 different values and 256^3 is 16777216. The LAB colour space is different; it exists out of a luminosity layer 'L' and two chromaticity layers 'A' and 'B'. The A-layer indicates where colour falls along the red-green axis, while the B-layer indicates where the colour falls along the blue-yellow axis [23]. It's said that the LAB-space is the best for finding differences in colours [24].

5.2.2 Applying the K-means algorithm

To be able to apply the algorithm for colour segmentation, one must shape the right matrix. The rows are the values for each variable and the columns are the different variables. When we work with a RGB-space, the input matrix should be of the size (number of Columns x number of rows, 3). The columns are the RGB-planes and the rows are the values corresponding to those planes. The matrix for the LAB-space is different since only the A- and B- plane determine the colour. The input matrix should therefore be of size (number of Columns x number of rows, 2). The columns are the AB-planes and the rows are the values corresponding to those planes. To calculate the location of the centroids, the squared Euclidean distance is used, this is the default mode where the position of the centroids gets determined by the means of the points in that cluster [25]. The last thing to decide is the amount of clusters the K-means algorithm should return, 3 clusters gave the best results in this example.

5.2.3 Comparing the results

Below image shows the results of the three clusters. It's very clear that the LAB-segmentation returns the best results since there's almost no skin visible. The noise that is still visible can easily be filtered away.

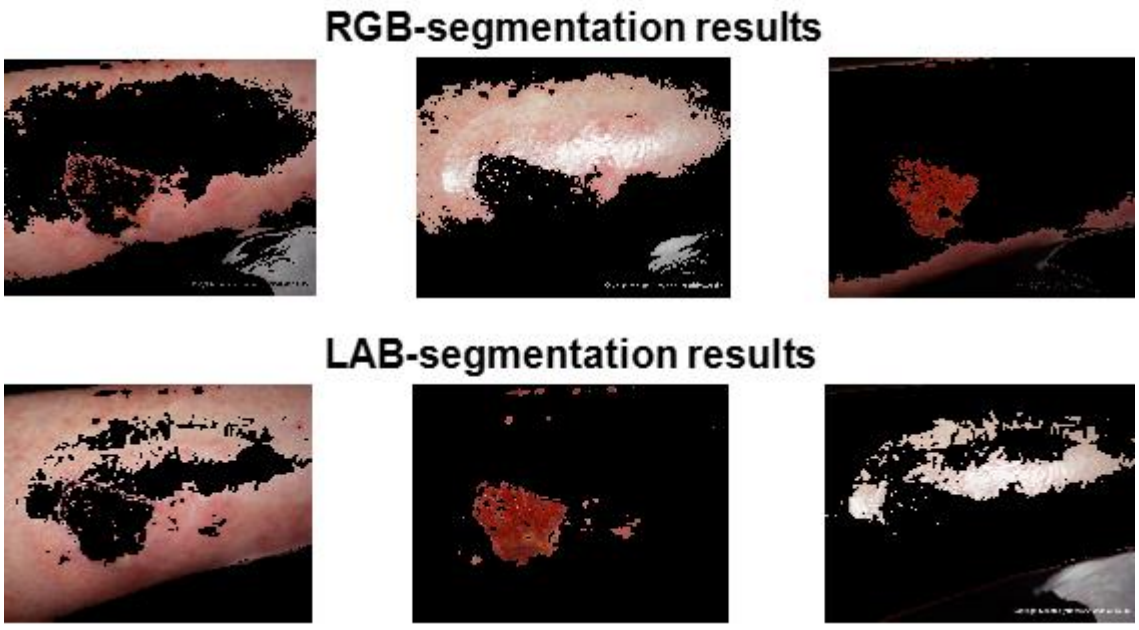

Figure 37: Comparing the results of the RGB- and the LAB-segmentation after the K-means algorithm

The code used in this application is realised with the help of a tutorial [26].

6 Classification algorithms

Classification algorithms are used to make conclusions about data. With this algorithm, we could, for example, determine if someone is running, walking or sleeping based on the heart rate of that person. The whole idea is to classify certain data points based on their features.

In the next subchapters, the decision tree, support vector machine and k nearest neighbours algorithm will be explained.

6.1 Decision Tree algorithm

Following image will be used to explain this algorithm.

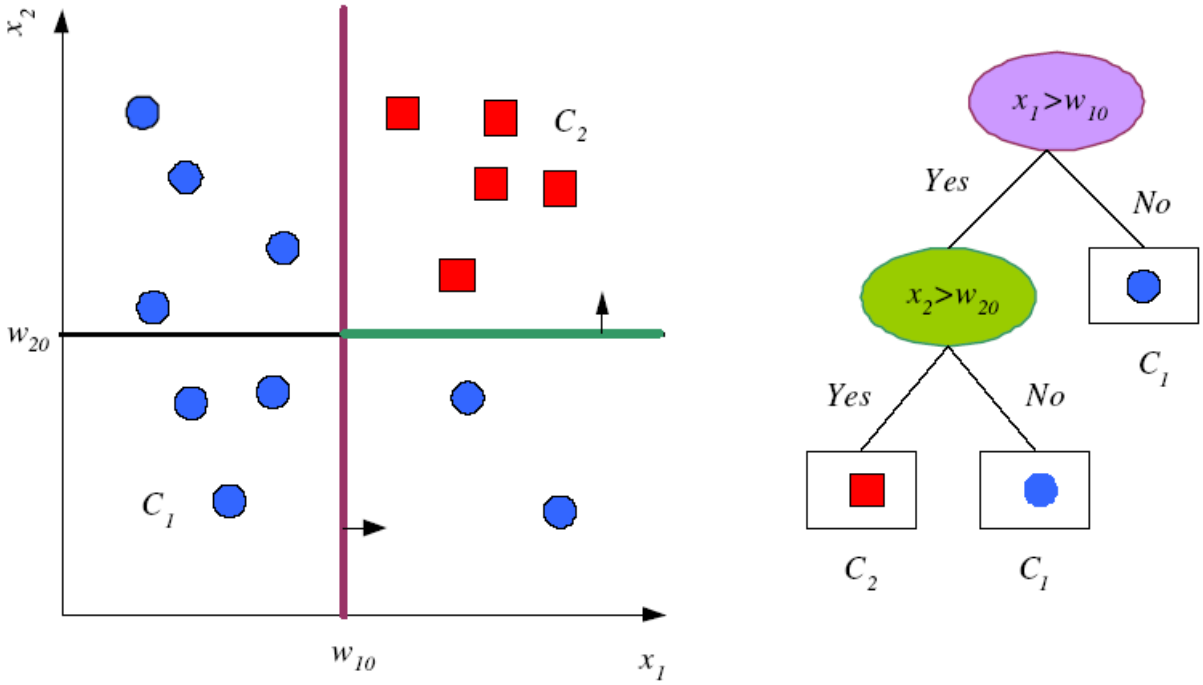

Figure 38: Decision Tree algorithm, example [27]

This image is a representation of data points that are classified by a decision tree. The tree will classify any data point where x_1 (or the horizontal value) is smaller than w_{10} as a blue dot. When this value is bigger than w_{10} we will compare x_2 with w_{20} to further classify the remaining data points.

To train a decision tree we will need a training set of data. This data will be fed into an algorithm that will return the decision tree. This classification is based on features. In the image above we can notify two features, x_1 and x_2. The algorithm will decide the best possible split and executes this as first. $x_2 > w_{20}$ and $x_1 > w_{10}$ are nodes. Because the amount of data and features is limited, only two nodes will be sufficient. However, this is not the case for most machine learning problems. Normally there will be more features and data points and therefore also more nodes [28].

6.2 Support Vector Model algorithm (SVM)

To explain this algorithm, following image will be used:

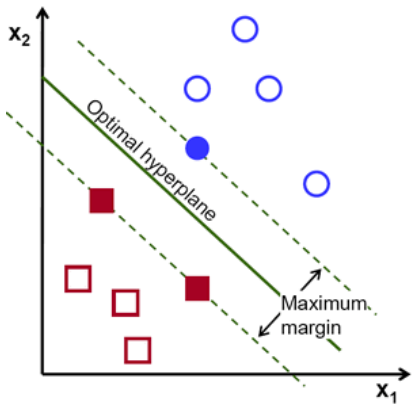

Figure 39: Support Vector Model algorithm, example [29]

This algorithm will classify a data point based on its position relative to the optimal hyperplane, given in the image. This optimal hyperplane is not necessary a line, it very rare is in fact.

The optimal hyperplane is determined by studying data points. The best boundary will be decided for classification and will be considered as a reference to further classify new data points [29].
Following image will show a real example of a SVM model and will show that linearity is not usual.

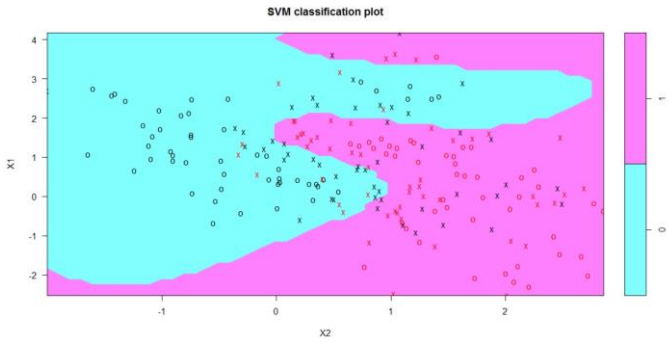

Figure 40: Support Vector Model algorithm, practical example [30].

When programming a SVM, you'll notice that there is a very important parameter C, this parameter determines the accuracy of the algorithms but also the speed. When C is low, it'll smooth-out the decision boundary, or classification, and will therefore allow for a certain inaccuracy. Increasing C will do the opposite; the decision boundary will not be smooth and every training data point will be correctly classified. Although the latter seems like a logic choice, once should be cautious of overfitting. The image on the next page will show this [31].

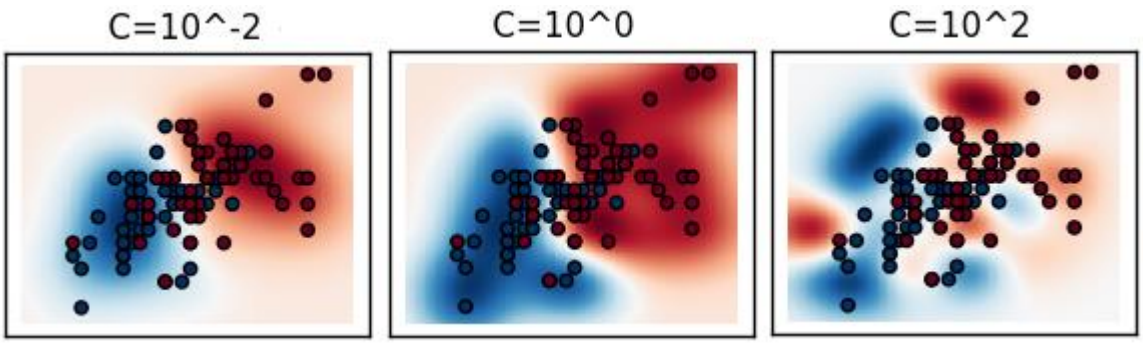

Figure 41: SVM, influence of parameter C

6.3 K Nearest Neighbours algorithm (k-NN)

This algorithm is the easiest one to implement and fastest to execute. However, there's always a price to be paid and in this case it's the accuracy.

To explain this algorithm, following image will be used.

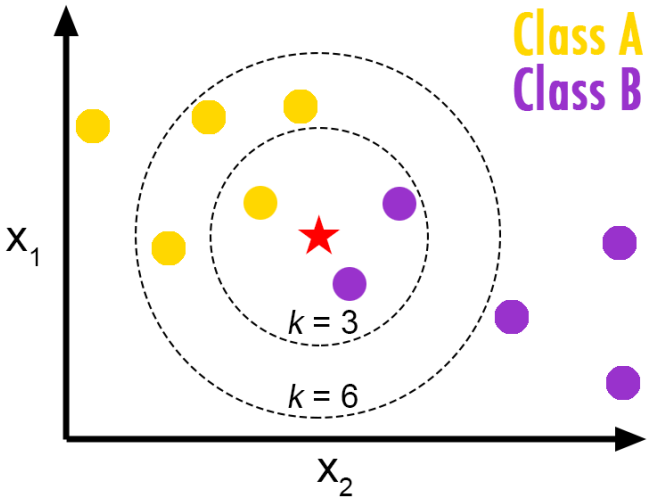

Figure 42: K Nearest Neighbours algorithm, example [32]

In this example, the star is a new data point. The algorithm will decide that the star belongs to the purple dots, or class B, if we are dealing with a 3 (=k) nearest neighbour algorithm. This is because the three nearest neighbours are 2x class B and 1x class A. When k would be 6, the algorithm would decide that the data point belongs to class A or the yellow dots. This is because the six nearest neighbours are 4x class A and 2x class B. To train this algorithm, an already classified data structure has to be given [32].

6.4 Applying classification on images to recognize wounds

This example is a continuation of the example explained in 5.2 Wound segmentation based on colour with the K-means algorithm. The K-means algorithm returned 3 images, to further determine which of these images is a wound, the images should be classified with the labels 'wound' and 'no_wound' and return the image of the wound. The next chart is a completion of the chart in Figure 36 and further explains what this example will try to achieve.

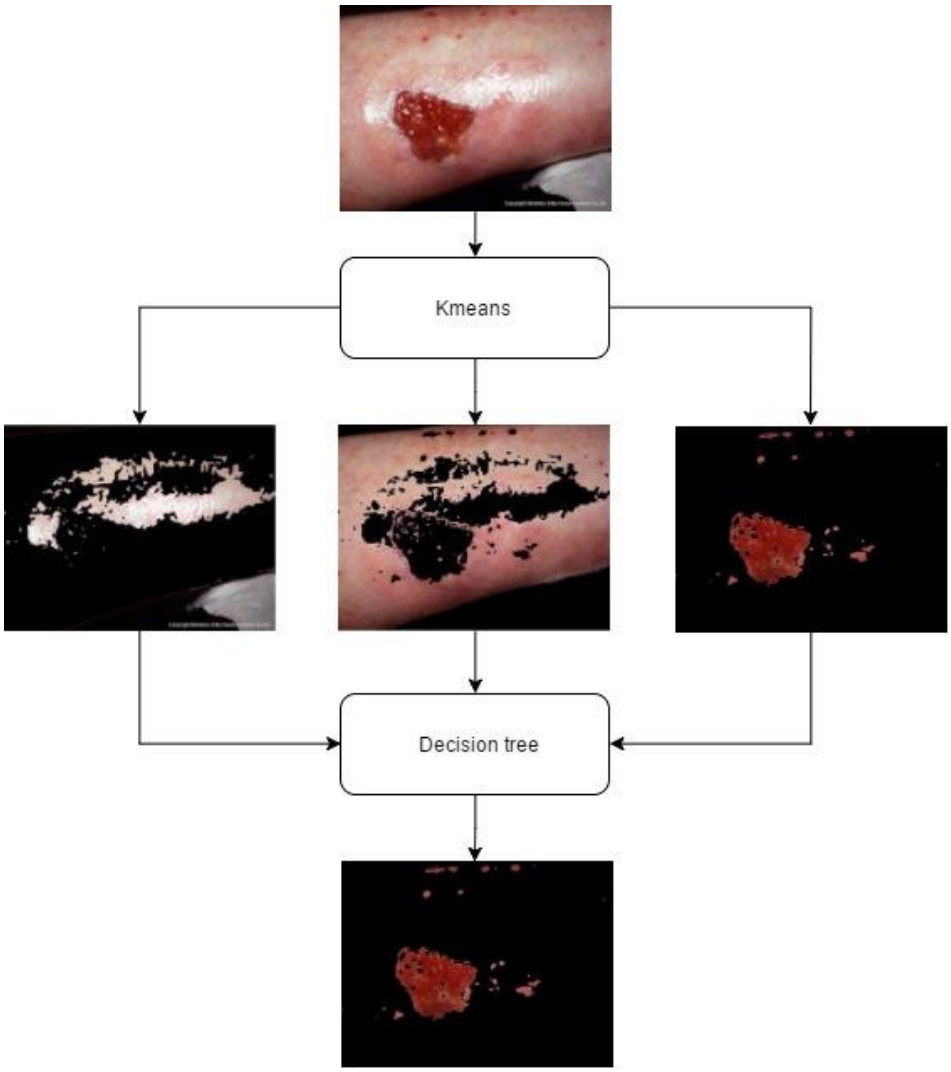

Figure 43: Image classification for wound detection

The algorithm of choice to do the classification, is a decision tree. No confusion matrix has been made due to the lack of wound images, the tree has been trained and immediately implemented in the colour segmentation application. The results were outstanding. No other algorithms have been tested, however, there has been algorithms tested before on other data, and the decision tree happened to be the most accurate. The only pre-processing that has been implemented is calculating the grayscale of the images and rescaling them to a 128x128-matrix, afterwards the image has been flattened to a 1x16384-matrix, to be used as features.

7 Machine learning kernels

Kernels are used in several machine learning algorithms, but perhaps the most famous usage, is within support vector machines. The most basic explanation of a kernel is that it makes calculations on your data so it can be processed, or rightly classified by the chosen algorithm. The kernel you should choose depends on how the data is spread.

7.1 Linear kernel

A linear kernel separates the data in a linear way, thus with a straight line.

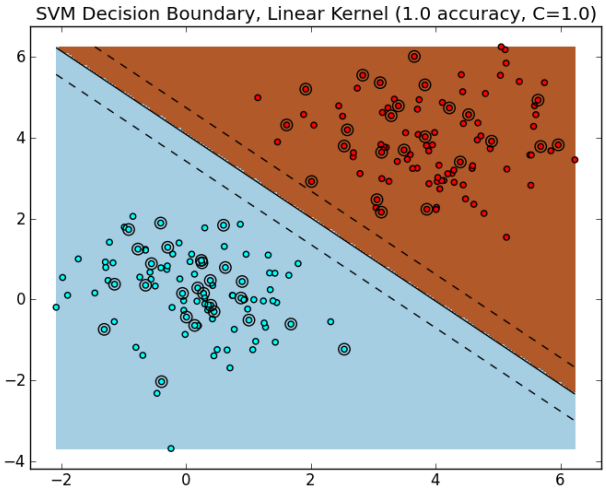

Figure 44: The linear kernel [33]

The kernel function is as follows:
$k(x, y) = x^T * y$ [34]
The T in the formula indicates a transposition.
In many practical cases, the data won't be linearly separable, and that's where this kernel will fail but the next two prevail.

7.2 Radial basis function kernel (RBF)

Many algorithms select this kernel as their default kernel, because it's a very common one and will in many cases achieve the best accuracy. Following images will help you realise what this kernel does when compared to the linear kernel.

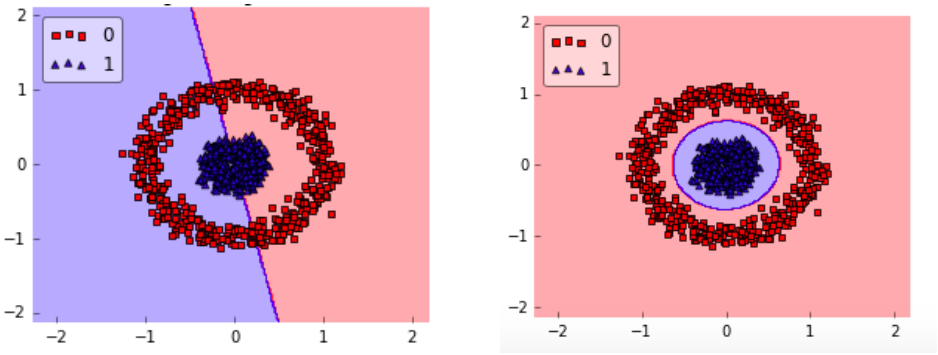

Figure 45: linear vs. RBF-kernel

The data in both images are spread in circles, therefore not linear. When you approach this data with a linear kernel, you'll get the result on the left. This result is not what we want because it has a very low accuracy. The RBF-kernel has as goal to classify the data, in this case as circles. This is achieved with help of the following formula.

$k(x, y) = \exp(-\gamma \|x - y\|^2)$ [34]

This function maps all the non-linear data to a n-dimension where it becomes linearly separable and then it maps this classifier back to the original space. Following image will illustrate this behaviour.

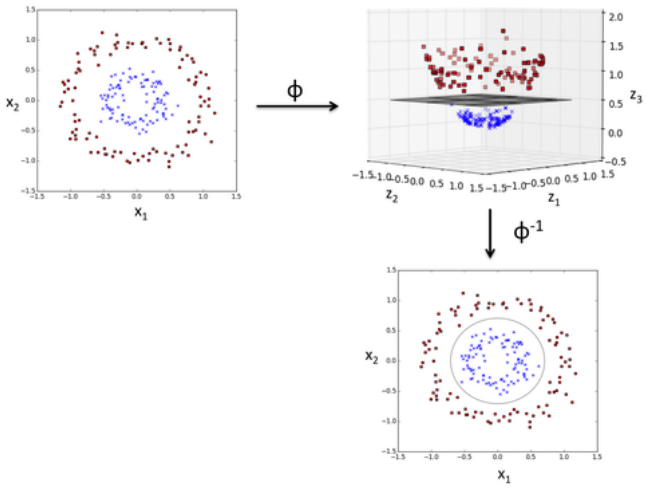

Figure 46: Calculating the RBF-classifier [33]

Here you can see that the original data is mapped into a 3D space, where it can place a linear hyperplane. This calculation is then reversed back to the original space, and a classifier has been made.

7.3 Polynomial kernel

The last kernel that will be discussed is the polynomial kernel, this is another variant of a not linearly separable classification. It's not made to classify circular positioned values but values that can be separated by a polynomial function. This kernel is not as common as the ones mentioned above. The following image will show all three kernels at once.

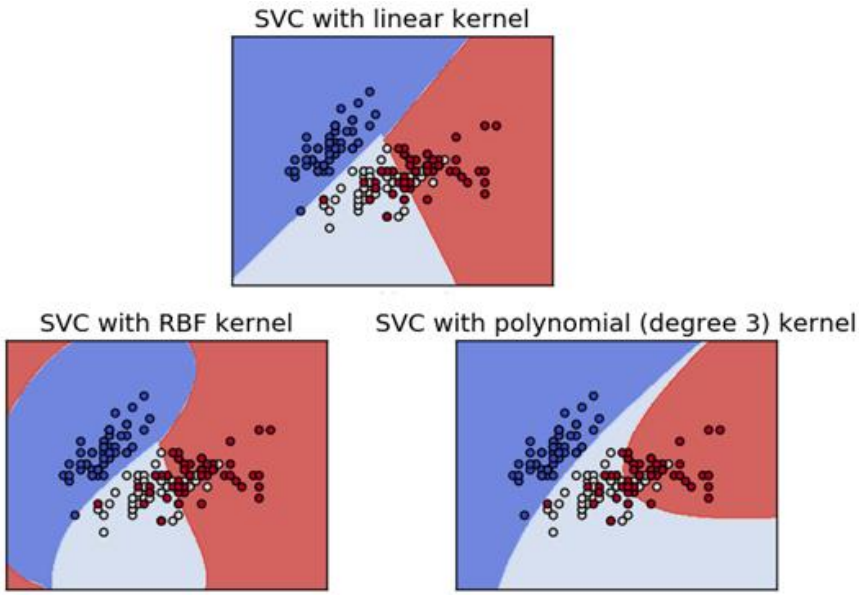

Figure 47: Comparison of all the kernels [35]

Here you can clearly see that the kernel is trying to separate the data with a polynomial function rather than a linear or RBF. This is achieved with the following function. It is also important to notice that the same approach as in RBF-kernels is used to get the right classifier. The mapping will not be shown again.

$k(x, y) = (\gamma x^T y + c_0)^d$ [34]

On the image, you can also see that in this case all the kernels could deliver a decent accuracy. The only way to determine the best one, is through testing them out on your data.

7.4 Important parameters

Every kernel has its own formulas, and therefore also its own parameters that the user must choose. This chapter will explain all the parameters used in this chapter.

γ or gamma, is both used in the RBF and polynomial kernel. This parameter decides how much influence a single data point can have, or how far data points may variate from each other for the algorithm to still consider them close related. Following image will explain the influence of this parameter [36].

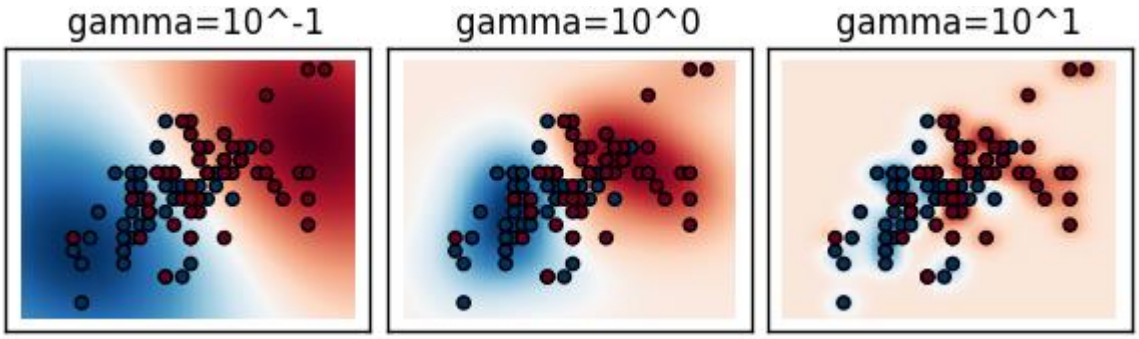

Figure 48: Kernels, influence gamma parameter

d or degree, is in this paper only mentioned in the polynomial kernel. It's a degree of flexibility, the higher d, the more flexible the decision boundary can become. Following image will illustrate this [37].

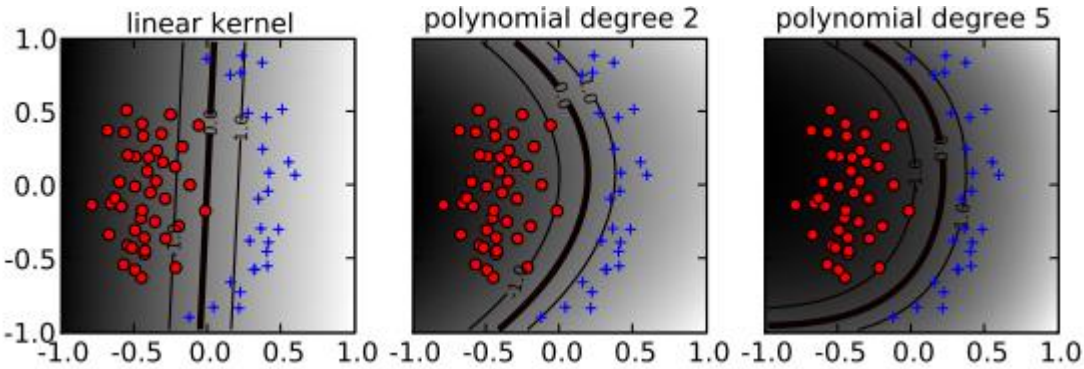

Figure 49: kernels, influence of the degree parameter

7.5 Conclusion

Only three kernels have been discussed, although there are many more. The choice of kernels is very often a fight between the linear kernel and the RBF-kernel where you need to compromise between a simple, fast, but very often a less accurate kernel or between a complex kernel that can take lots of time to train and predict, since it has to map data to other dimensions, and these dimensions can be, in theory, infinite. Another option is to calculate your own kernel, but this will not be further explained in this chapter, since there is no theoretical approach to this, it solely depends on your data.

8 Artificial neural networks

Our brain is truly remarkable, the fact that we can remember many events and memories for extended periods of time, is mindboggling. It would make life easier if we could implement this wonder of nature in a machine, so it can do complex logic things, that our mind can already do. This is exactly what artificial neural networks try to accomplish. What they are, how they work and how to implement them, will be described in this chapter.

8.1 Neural networks and the human body

Neurons are the unit a brain uses to communicate and process information. A neuron exists out of a cell body and many branches. The axon is one of these branches, made for transmission to other neurons while the synapse receives the information of an axon, and directs it to the neuron. The other branches are called dendrites. Following diagram represents such a neuron.

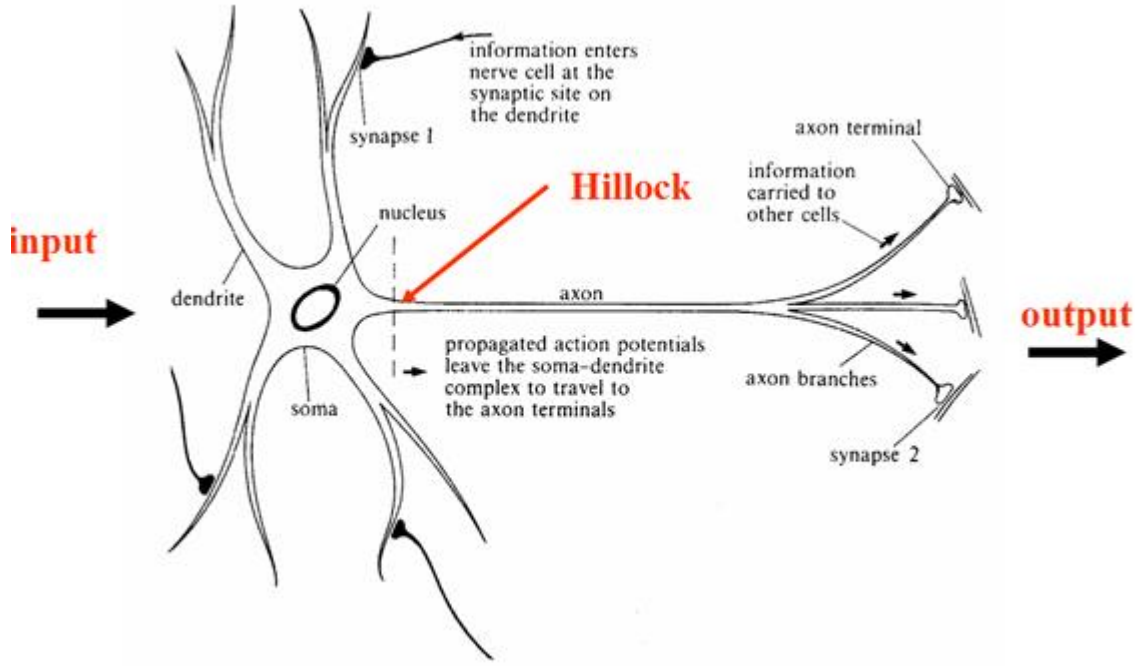

Figure 50: Diagram of a human neuron

A neuron has a cell membrane, this membrane is responsible for a voltage difference in- and outside the cell, this voltage difference is also called the membrane potential. When the neuron receives a certain input that is high enough, it'll provide an action potential, or neural spike to the axon, which will carry on the signal and provide it to a synapse that is connected to a neuron. These spikes are very important, since the brain uses them to communicate between neurons [38].

8.2 Single layer neural networks

The next diagram will show the configuration of a perceptron. A perceptron is a neuron connected to n-different neurons. This connection is not direct, a weighted summation will be done of each neuron value, when this summation is high enough, it'll fire an output, this output depends on the activation function. The threshold upon which the neuron fires is adjustable by changing the bias, also called threshold processor [39].

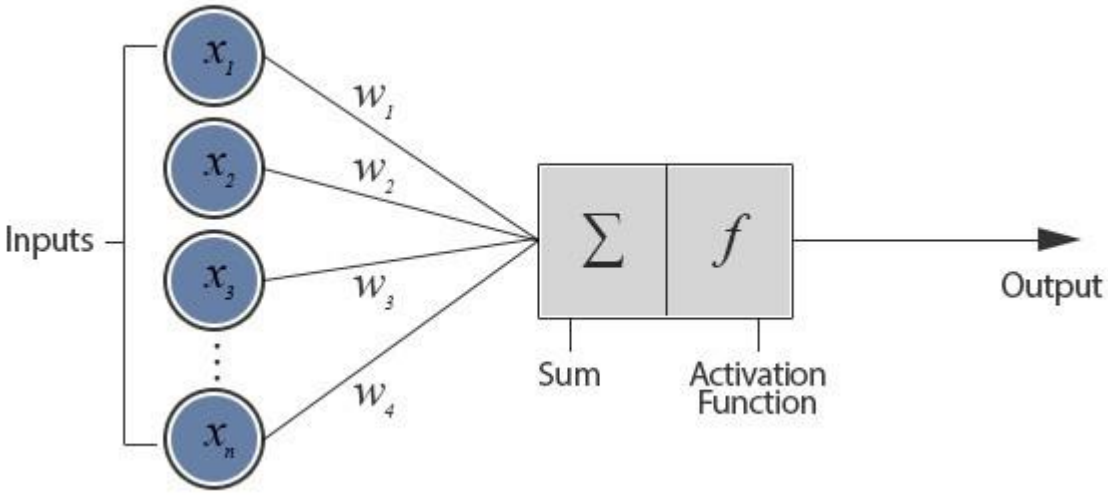

Figure 51: Diagram of a perceptron configuration [39]

8.2.1 Training a single layer network

Training neural networks can be done by optimising the values of the weights and biases, so that the output will fire correctly. When the output is correct, no changes to the bias and weight will be applied. Otherwise the weight and bias will be adjusted per the following formulas:

$$b = b + (Ycor - Ypred)$$
$$W(i) = W(i) + \alpha * g'(sumOfAllInputs)(Ycor - Ypred) * X(i)$$

- b is the bias;
- Ycor is the output that should have appeared on the neuron;
- Ypred is the actual output from the neuron;
- W is the weight vector;
- α is the learning rate;
- g' is the derivative of the activation function, ignored in case of a step function;
- X is the input fed to the neuron [39].

The training is complete when all the epoch has been completed without error. One epoch has been finished when all the input vectors have been fed once to the neural network [38].

8.3 Multi-layer neural networks

The training of a single layer network is simple to understand, since the input is directly connected to the output. Therefore, the output knows what the input value is and can train with this information. A multi-layer network however is different; it has hidden layers. Every input node is connected to a hidden layer node, and every hidden layer node is connected to the output. Following diagram shows this.

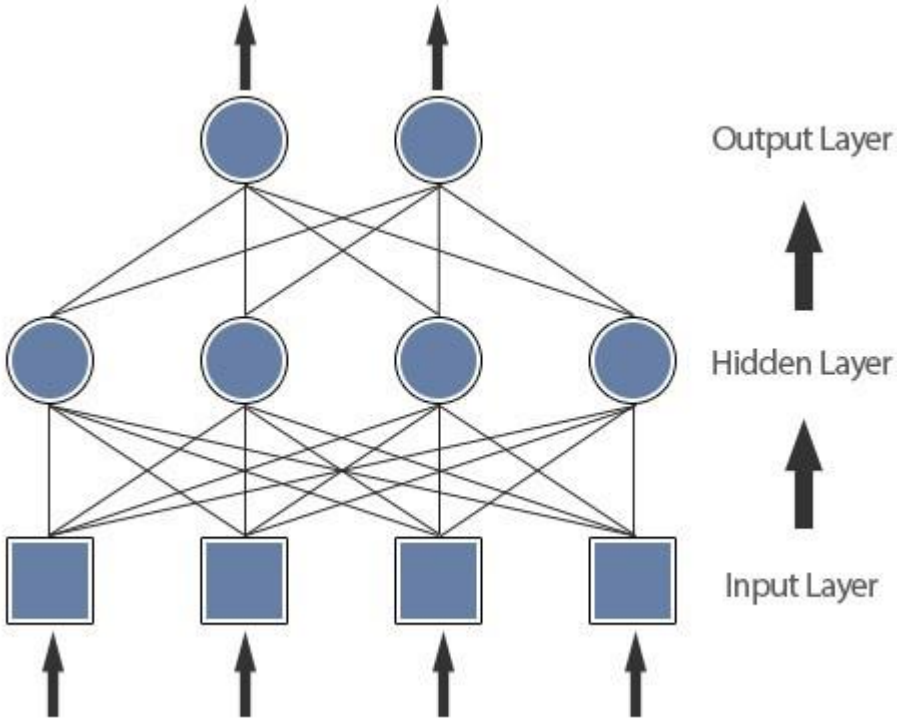

Figure 52: Diagram of a multi-layer network [40]

The problem here is that the output knows nothing about the input, it can only see what the hidden layer sends. This means that we can only reach one error, the error of the output, which can be used to train the weight vectors between the hidden layer and the output. To train the weights between the input and the hidden layer, we need to know the error at the hidden layer, this is done with backpropagation [40].

8.3.1 Training a multi-layer network with backpropagation

To explain this technique, let's consider following neural network, chosen to keep the calculations clean and simple.

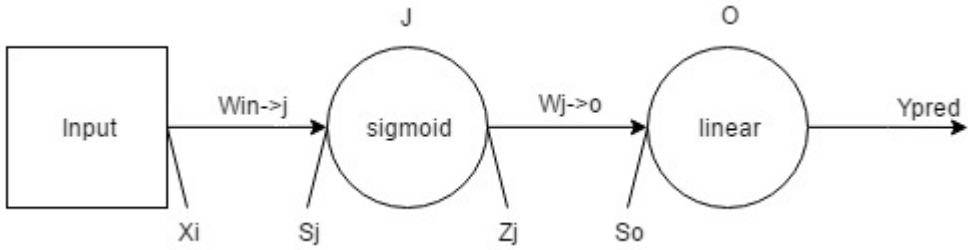

Figure 53: Simplified multi-layer neural network with 1 hidden layer

In this example, we'll want to define the weights to minimalize the error. This can be done with stochastic gradient descent, which updates the weights with the following formula. The W stands for the weights; the E represents the error and η is the learning rate.

$$W(i \to j) = W(i \to j) - \eta \frac{\partial E}{\partial W(i \to j)} \quad (eq.1)$$

In our case the error will be defined with the following formula. Ypred are the predicted labels, and Yact is the actual label that the predicted label is supposed to be.

$$E = \frac{1}{2}(Ypred - Yact)^2 \quad (eq.2)$$

Now that we know the basics, let's move on to the figure, you can say:

$$Sj = W(in \to j) * Xi$$
$$Zj = \sigma(INj) = \sigma(W(in \to j) * Xi)$$
$$So = W(j \to o) * Zj$$
$$Ypred = INo = W(j \to o) * \sigma(W(in \to j) * Xi)$$
$$E = \frac{1}{2}(Ypred - Yact)^2 = \frac{1}{2}(W(j \to o) * \sigma(W(in \to j) * Xi) - Yact)^2$$

Let's calculate all the derivatives, note that the first calculation is based on a linear activation formula and the second on a sigmoid. Going on with eq. 2:

$$\frac{\partial E}{\partial W(j \to o)} = \frac{\partial}{\partial W(j \to o)} \frac{1}{2} (Ypred - Yact)^2$$
$$= \frac{\partial}{\partial W(j \to o)} \frac{1}{2} (W(j \to o) * Zj - Yact)^2$$
$$= (W(j \to o) * Zj - Yact) \frac{\partial}{\partial W(j \to o)} (W(j \to o) * Zj - Yact)$$
$$= (Ypred - Yact)(Zj) \quad (eq.3)$$

$$\frac{\partial E}{\partial W(in \to j)} = \frac{\partial}{\partial W(in \to j)} \frac{1}{2} (Ypred - Yact)^2$$
$$= (Ypred - Yact) \left(\frac{\partial}{\partial W(in \to j)} (W(j \to o) * \sigma(W(in \to j) * Xi) - Yact \right)$$
$$= (Ypred - Yact) * W(j \to o) \left(\frac{\partial}{\partial W(in \to j)} \sigma(W(in \to j) * Xi) \right) \quad (eq.4)$$

First some additional information, so the next step won't be overwhelming [41]

$$\sigma(x) = \frac{1}{1+e^{-x}} \rightarrow \frac{d\sigma}{dx} = \frac{\partial(1+e^{-x})^{-1}}{\partial x} = \frac{-1}{(1+e^{-x})^2} * (-e^{-x}) = \frac{e^{-x}}{(1+e^{-x})^2}$$
$$= \frac{1}{1+e^{-x}}\frac{e^{-x}}{1+e^{-x}} = \frac{1}{1+e^{-x}}\frac{1+e^{-x}-1}{1+e^{-x}} = \frac{1}{1+e^{-x}}\left(1-\frac{1}{1+e^{-x}}\right)$$
$$= \sigma(x) * \big(1-\sigma(x)\big) \quad (eq.5)$$

Let's move on where we left off, and insert eq. 5 into eq. 4:

$$= (Ypred - Yact) * W(j \rightarrow o)\left(\frac{\partial}{\partial W(in \rightarrow j)}\,\sigma(W(in \rightarrow j) * Xi)\right)$$
$$= (Ypred - Yact) * W(j \rightarrow o)\left(\sigma(Sj) * \big(1-\sigma(Sj)\big) * \frac{\partial Sj}{\partial W(in \rightarrow j)}\right)$$
$$= (Ypred - Yact) * W(j \rightarrow o)\left(\sigma(Sj) * \big(1-\sigma(Sj)\big) * \frac{\partial(W(in \rightarrow j) * Xi)}{\partial W(in \rightarrow j)}\right)$$
$$= (Ypred - Yact) * W(j \rightarrow o) * \sigma(Sj) * \big(1-\sigma(Sj)\big) * Xi \quad (eq.6)$$

Previously we stated the formula for the recalculation of the weight vector done with stochastic gradient descent:

$$W(i \rightarrow j) = W(i \rightarrow j) - \eta\frac{\partial E}{\partial W(i \rightarrow j)}$$

This means that $-\eta\frac{\partial E}{\partial W(i \rightarrow j)}$ equals the variation between the recalculated and the previous weight therefore we can conclude that eq. 3 appears in eq. 6:

$$\Delta W(in \rightarrow j) = -\eta * (Ypred - Yact) * W(j \rightarrow o) * \sigma(Sj) * \big(1-\sigma(Sj)\big)(Xi)$$
$$\Delta W(j \rightarrow o) = -\eta * (Ypred - Yact)(Zj)$$

These equations make it possible to update all the weights and gain an optimal neural network. This was just an easy example, imagine calculating the weights of a network with 3 hidden layers, were each layer has 100 nodes, that would take forever, unless you notice the structure. The error propagates backwards through the network. If there would be another layer between the input and the hidden layer, you would notice that when you calculate it, the beginning of the weight variation formula would start with $-\eta * (Ypred - Yact) * W(j \rightarrow o) * \sigma(Sj) * \big(1-\sigma(Sj)\big)$ and be multiplied by another term. There's a way to simplify this, with layman's equations $\delta(j) = \frac{\partial E}{\partial Sj}$ or, the error of the network varies with the input of layer j.

$$\delta(j) = \frac{\partial E}{\partial Sj} = \frac{\partial}{\partial Sj}\frac{1}{2}(Ypred - Yact)^2 = (Ypred - Yact)\frac{\partial Ypred}{\partial Sj}$$

If j would be the output node you could say (notice that the output function is linear):

$$(Ypred - Yact)\frac{\partial Ypred}{\partial Sj} = (Ypred - Yact) * f'j(Sj)$$

Because $Ypred = Fj(Sj)$

Now let us say that layer j is not an output, but a hidden layer that leads to another layer then we can implement the chain rule.

$$\frac{\partial Ypred}{\partial Sj} = \frac{\partial Ypred}{\partial Zj}\frac{\partial Zj}{\partial Sj} = \frac{\partial Ypred}{\partial Zj}\ f'j(Sj)$$

This just states that the Ypred is dependent on the output of that layer, but not equal to since it's still connected to another node. While Zj is obviously also related to Sj. Layer j goes to another layer, in our case it's the output later. It's however possible to have multiple output nodes, with $So = Zj * W(j \rightarrow o)$ while the So of one output node doesn't affect the So of another output node. Therefore, we can use the chain rule again.

$$\frac{\partial Ypred}{\partial Zj} = \frac{\partial Ypred}{\partial So}\frac{\partial So}{\partial Zj}$$

The explanation is the same as previously. When we fill this into $\frac{\partial Ypred}{\partial Sj}$, we get the following.

$$\frac{\partial Ypred}{\partial Sj} = f'j(Sj)\sum_{nNodes}\frac{\partial Ypred}{\partial So}\frac{\partial So}{\partial Zj}$$

The summation is there because of the multiple nodes.

$$So = Zj * W(j \rightarrow o) \rightarrow \frac{\partial So}{\partial Zj} = W(j \rightarrow o)$$
$$\frac{\partial Ypred}{\partial Sj} = f'j(Sj)\sum_{nNodes}\frac{\partial Ypred}{\partial So}W(j \rightarrow o)$$

We can fill this in, into a previously obtained equation.

$$\delta(j) = (Ypred - Yact)\frac{\partial Ypred}{\partial Sj}$$
$$\delta(j) = (Ypred - Yact)\ f'j(Sj)\sum_{nNodes}\frac{\partial Ypred}{\partial So}W(j \rightarrow o)$$

The next step is to push $(Ypred - Yact)$ into the summation and replace it with

$$\delta(o) * W(j \rightarrow o)$$
$$\delta(j) = f'j(Sj)\sum_{nNodes}\delta(o) * W(j \rightarrow o)$$

In this case, we also backpropagated the error. Now let's test this equation with our previous diagram. Do note that all the summations become obsolete since there is only one node in each layer.

$\delta(o) = (Ypred - Yact)\ \ (eq.7)$

$\delta(j) = \delta(o) * W(j \rightarrow o) * \sigma(Sj)\big(1 - \sigma(Sj)\big)\ \ (eq.8)$

If you look at the weight calculations from the previous backpropagation calculations, you can see some similarities.

The following equation displays eq. 6, eq. 3, eq. 7 and eq. 8

$$\begin{cases} \Delta W(in \to j) = -\eta * (Ypred - Yact) * W(j \to o) * \sigma(Sj) * \big(1 - \sigma(Sj)\big)(Xi) \\ \Delta W(j \to o) = -\eta * (Ypred - Yact)(Zj) \\ \delta(o) = (Ypred - Yact) \\ \delta(j) = \delta(o) * W(j \to o) * \sigma(Sj)(1 - \sigma(Sj)) \end{cases}$$

$$\Delta W(in \to j) = -\eta * \delta(j) * (Xi)$$
$$\Delta W(j \to o) = -\eta * \delta(o) * (Zj)$$

This is the simplest and most general form of backpropagation. All the previous calculations are inspired by a tutorial [42].

8.4 Activation functions

There are a lot of activation function, some of which will be explained in the subchapters below.

8.4.1 Step activation function

Following image will show a step activation function. When the input is smaller than 0, nothing will happen but when it becomes higher or equal to zero, the neuron fires. This is a simple true or false activation function. This function is also called the Heaviside function. The Y-axis is the output and the X-axis is the input [38]. The problem here is that there is no derivative of this function, which makes it unfit for backpropagation.

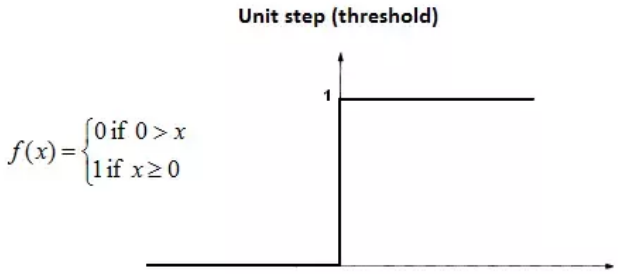

Figure 54: Visualisation of the step activation function [43]

8.4.2 Linear activation function

This function is just a basic linear function; the input is mapped to the output. This activation function is not often used because of its linearity, most data in practical applications aren't linear, and therefore require a non-linear activation function. This activation function is often used on the output layer.

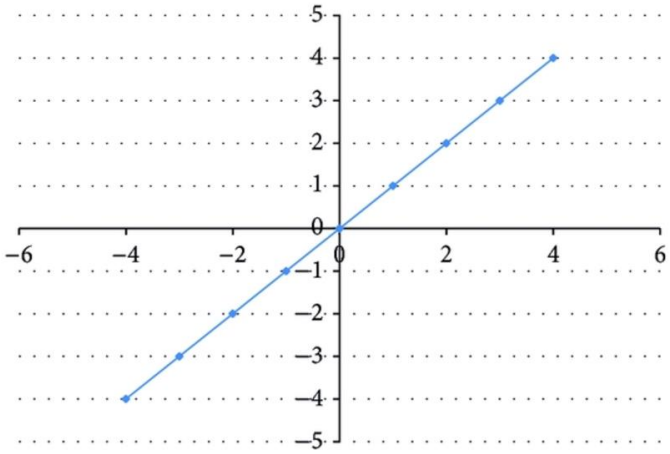

Figure 55: Visualisation of the linear activation function [44]

8.4.3 Sigmoid activation function

The sigmoid activation is a non-linear function and is a bit more complex, it maps the data between 0 and 1 and it increases the firing rate of neurons when the input increases. The next image will show this function. The Y-axis is the output and the X-axis is the input [38].

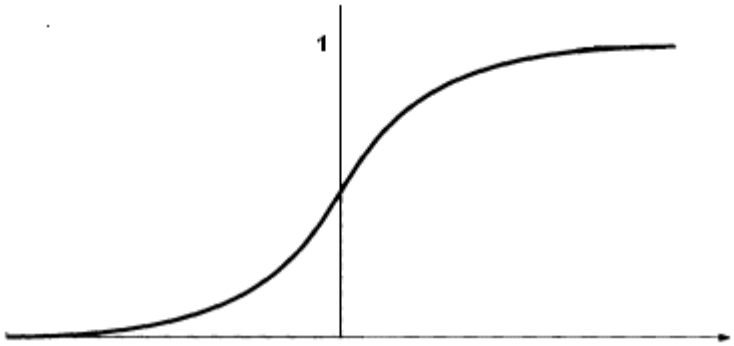

Figure 56: Visualisation of the sigmoid activation function [45]

This activation function was very popular, until they discovered the two main disadvantages [46].

- The sigmoid function saturates when y is either 0 or 1, this kills the gradient since it'll be almost equal to 0, this is a problem when backpropagation returns the error backwards.
- The output isn't zero-centred and always positive. This means that the weights will either all be positive or negative, this phenomenon could create zigzag effects between the weights updates, which isn't optimal.

8.4.4 tanh activation function

The tanh activation function is very similar to the sigmoid activation function, the main difference is, that this function maps the data between 1 and -1, which makes it zero-centred. The saturation problem is however still there. The next image will show this function. The Y-axis is the output and the X-axis is the input [46].

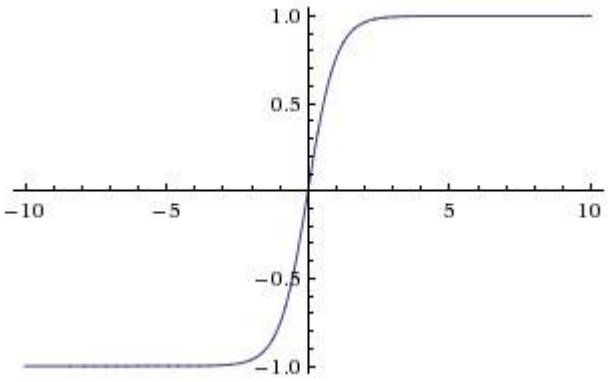

Figure 57: Visualisation of the tanh activation function [46]

8.4.5 ReLU activation function

The ReLU, short for Rectified Learning Unit, tries to solve the shortcomings of the other functions. It's zero-centred, you can derivate it and it doesn't saturate. Another advantage is the simplicity of the function; this causes the training of a network to be more efficient.

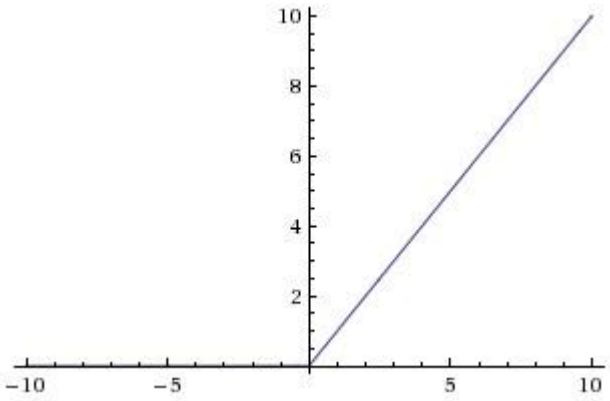

Figure 58: Visualisation of the ReLU activation function [46]

However, there's also one disadvantage, ReLU neurons can die. Sometimes the gradient is too large, which influences the weights drastically. The drastic change will prevent the neuron from ever activating again, and causes it to die. This can however be prevented by decreasing the learning rate, although that will not solve everything [46].

8.5 Neural networks in TensorFlow

This chapter will test classification done by a neural network, based on tomography data. There are three types of wounds labelled: daw, vlue and tib. The meaning of these labels is irrelevant.

8.5.1 Pre-processing the wounds

The wound gets extracted from the image, placed into a forward model, simulated and the inverse image gets calculated. The reversed image will afterwards be transformed into a grayscale image of 128x128. This image is flattened in the final step, and fed to the neural network. The full process is visualised on the next figure.

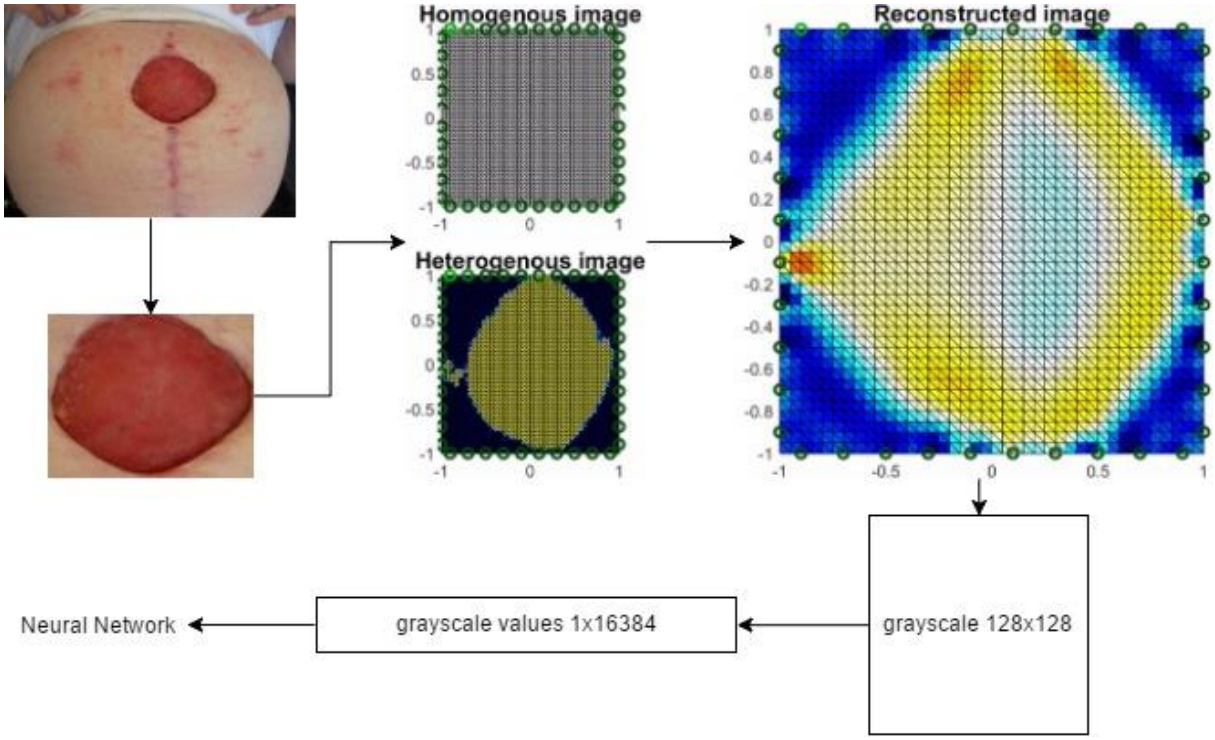

Figure 59: Pre-processing task chart

8.5.2 The multi-layer neural network for wound classifications

The most popular activation function is a Rectified Linear Unit, this is used on every layer, except the output layer. The output layer exists out of a linear activation function that maps the input to the output. There is a total of 3 classes, which always equals the number of outputs. All the other layers exist out of 1500 neurons, which are all connected to each other. The input, or the labels, are one-hot-encoded. This means that the first label is 0 0 1, the second 0 1 0 and the third 1 0 0. This neural network is inspired by a MNIST-tutorial [47] and visualised below.

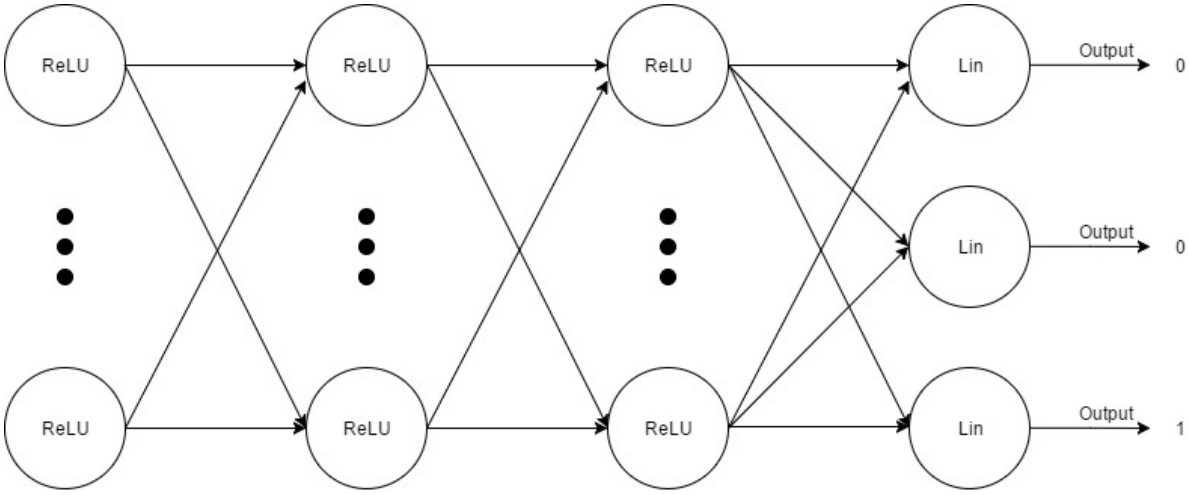

Figure 60: Implemented artificial multi-layer neural network

8.5.3 Results of the multi-layer neural network

The network hasn't been accurately tested due to a low number of images, roughly 10 to 12. The accuracy of the neural network reaches from 85.7% to 52%, with 72% being the most common accuracy. The following image will show the loss value in function of the epoches.

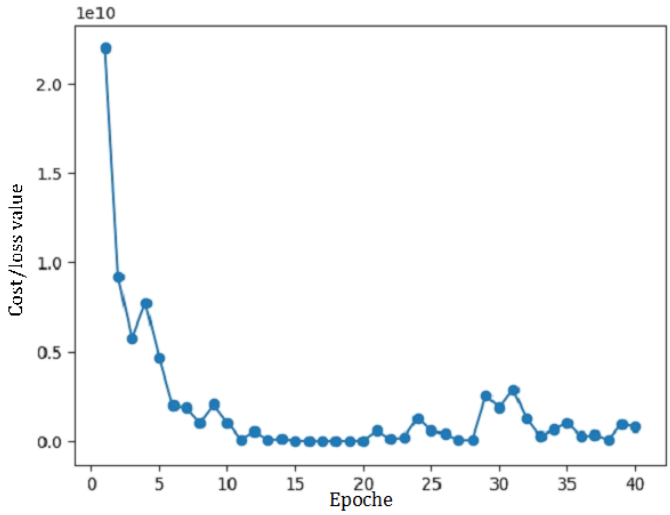

Figure 61: The decreasing of the loss value in function of the Epoche

9 Conclusion

As explained in the abstract, the main objective was to do a feasibility study on implementing machine learning algorithms on medical applications, which succeeded. The study on tomography, wound segmentation and neural networks seems to be useful in the real-time monitoring of wound recovery. While a similar form of regression on liquids, could be applied to liquids in infusion pumps. Finally, we have the regression on yeast cells which could be useful for many health-related applications, and on other cell studies.

Some of the projects did not go without any problems. The regression on liquids is now based on the impedance, but the temperature should also be an important factor, since it can alter the impedance level. The reason why the temperature hasn't been implemented in the regression is due to measurement difficulties, the measurements were done twice and were not reliable.

The measurements on the growth of yeast cells failed, due to a short-circuit in the measurement system. This was discovered after visualising and analysing all the data, which caused a lot of lost time.

The tomography data in this thesis, is simulated. This has been done because the tomography measurements were not accurate at all, due to technical issues, which were beyond my reach.

Bibliography

[1] "Glossary of Terms Journal of Machine Learning." [Online]. Available: http://ai.stanford.edu/~ronnyk/glossary.html. [Accessed: 05-Jun-2017].

[2] "4-Steps to Get Started in Machine Learning: The Top-Down Strategy for Beginners to Start and Practice - Machine Learning Mastery." [Online]. Available: http://machinelearningmastery.com/4-steps-to-get-started-in-machine-learning/. [Accessed: 05-Jun-2017].

[3] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," in *2014 Science and Information Conference*, 2014, pp. 372–378.

[4] "How to Improve Machine Learning Results - Machine Learning Mastery." [Online]. Available: http://machinelearningmastery.com/how-to-improve-machine-learning-results/. [Accessed: 05-Jun-2017].

[5] "Train models to classify data using supervised machine learning - MATLAB - MathWorks Benelux." [Online]. Available: https://nl.mathworks.com/help/stats/classificationlearner-app.html. [Accessed: 05-Jun-2017].

[6] "Machine Learning | Microsoft Azure." [Online]. Available: https://azure.microsoft.com/nl-nl/services/machine-learning/. [Accessed: 05-Jun-2017].

[7] N. R. C. (US) and I. of M. (US) C. on the M. and P. of E. D. B. Imaging, "Electrical Impedance Tomography," *Chapter 9, Electr. Impedance Tomogr.*, 1996.

[8] "Electrical Impedance Tomography (EIT) System for Radiation-Free Medical Imaging Based on LabVIEW - Solutions - National Instruments." [Online]. Available: http://sine.ni.com/cs/app/doc/p/id/cs-14779#prettyPhoto. [Accessed: 22-May-2017].

[9] "EIDORS." [Online]. Available: http://eidors3d.sourceforge.net/index.shtml. [Accessed: 22-May-2017].

[10] "Pictures of wounds and surgical wound dressings." [Online]. Available: http://www.medetec.co.uk/files/medetec-image-databases.html. [Accessed: 05-Jun-2017].

[11] C. L. Yang, "Electrical impedance tomography: algorithms and applications," 2014.

[12] "Data pre-processing - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Data_pre-processing. [Accessed: 27-Feb-2017].

[13] J. Brownlee, "How to Prepare Data For Machine Learning - Machine Learning Mastery," 2013. [Online]. Available: http://machinelearningmastery.com/how-to-prepare-data-for-machine-learning/. [Accessed: 27-Feb-2017].

[14] "How to Prepare Data For Machine Learning - Machine Learning Mastery." [Online]. Available: http://machinelearningmastery.com/how-to-prepare-data-for-machine-learning/. [Accessed: 05-Jun-2017].

[15] "What is one hot encoding and when is it used in data science? - Quora." [Online]. Available: https://www.quora.com/What-is-one-hot-encoding-and-when-is-it-used-in-data-science. [Accessed: 05-Jun-2017].

[16] "Linear Regression for Machine Learning - Machine Learning Mastery." [Online]. Available: http://machinelearningmastery.com/linear-regression-for-machine-learning/. [Accessed: 30-May-2017].

[17] “Linear Regression Analysis in SPSS Statistics - Procedure, assumptions and reporting the output.” [Online]. Available: https://statistics.laerd.com/spss-tutorials/linear-regression-using-spss-statistics.php. [Accessed: 30-May-2017].
[18] J. Howbert, “Machine Learning Logistic Regression.”
[19] “100 series DC powered Pumps,” *Williamson - Specif. sheet*, p. 2, 2013.
[20] “Regression Analysis: How Do I Interpret R-squared and Assess the Goodness-of-Fit?” [Online]. Available: http://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit. [Accessed: 05-Jun-2017].
[21] “CS221.” [Online]. Available: http://stanford.edu/~cpiech/cs221/handouts/kmeans.html. [Accessed: 06-Jun-2017].
[22] “K-means clustering: how it works - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=_aWzGGNrcic. [Accessed: 06-Jun-2017].
[23] “Color-Based Segmentation Using K-Means Clustering - MATLAB & Simulink Example - MathWorks Benelux.” [Online]. Available: https://nl.mathworks.com/help/images/examples/color-based-segmentation-using-k-means-clustering.html. [Accessed: 25-May-2017].
[24] Gggustafson, “The Known Colors Palette Tool - Final Revision - Hopefully,” 2015. [Online]. Available: https://www.codeproject.com/Articles/243610/The-Known-Colors-Palette-Tool-Revised-Again. [Accessed: 25-May-2017].
[25] “k-means clustering - MATLAB kmeans - MathWorks Benelux.” [Online]. Available: https://nl.mathworks.com/help/stats/kmeans.html. [Accessed: 25-May-2017].
[26] “Color-Based Segmentation Using K-Means Clustering - MATLAB & Simulink Example - MathWorks Benelux.” [Online]. Available: https://nl.mathworks.com/help/images/examples/color-based-segmentation-using-k-means-clustering.html. [Accessed: 05-Jun-2017].
[27] Santini Marina, “Lecture 02: Machine Learning for Language Technology - Decision Trees…,” 2013. [Online]. Available: https://www.slideshare.net/marinasantini1/lecture02-machine-learning?from_action=save. [Accessed: 06-Jun-2017].
[28] P. Flach, “Desicion Trees,” in *Machine Learning The Art and Science of Algorithms that Make Sense of Data*, Cambridge University Press, 2012, pp. 133–138.
[29] “Introduction to Support Vector Machines — OpenCV 2.4.13.2 documentation.” [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html. [Accessed: 06-Jun-2017].
[30] “Support Vector Machine Tutorial (SVM).” [Online]. Available: https://www.dezyre.com/data-science-in-r-programming-tutorial/support-vector-machine-tutorial. [Accessed: 06-Jun-2017].
[31] É. D. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, “Support Vector Machines — scikit-learn 0.18.1 documentation.” [Online]. Available: http://scikit-learn.org/stable/modules/svm.html. [Accessed: 28-Feb-2017].
[32] “Classification using k-Nearest Neighbors in R | en.proft.me.” [Online]. Available: http://en.proft.me/2017/01/22/classification-using-k-nearest-neighbors-r/.

[Accessed: 06-Jun-2017].
[33] Raschka Sebastian, “How to Select Support Vector Machine Kernels,” 2016. [Online]. Available: http://www.kdnuggets.com/2016/06/select-support-vector-machine-kernels.html. [Accessed: 27-Feb-2017].
[34] É. D. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, “Pairwise metrics, Affinities and Kernels,” 2010. [Online]. Available: http://scikit-learn.org/stable/modules/metrics.html. [Accessed: 27-Feb-2017].
[35] É. D. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, “Plot different SVM classifiers in the iris dataset — scikit-learn 0.18.1 documentation,” 2010. [Online]. Available: http://scikit-learn.org/stable/auto_examples/svm/plot_iris.html. [Accessed: 27-Feb-2017].
[36] “RBF SVM parameters — scikit-learn 0.18.1 documentation.” [Online]. Available: http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html. [Accessed: 05-Jun-2017].
[37] A. Ben-Hur and J. Weston, “A User’s Guide to Support Vector Machines.”
[38] S. Barber, “AI : Neural Network for beginners (Part 1 of 3),” *CodeProject*, 2007. [Online]. Available: https://www.codeproject.com/Articles/16419/AI-Neural-Network-for-beginners-Part-of.
[39] “For Dummies — The Introduction to Neural Networks we all need ! (Part 1).” [Online]. Available: https://medium.com/technologymadeeasy/for-dummies-the-introduction-to-neural-networks-we-all-need-c50f6012d5eb. [Accessed: 27-May-2017].
[40] “For Dummies — The Introduction to Neural Networks we all need ! (Part 2) – TechnologyMadeEasy – Medium.” [Online]. Available: https://medium.com/technologymadeeasy/for-dummies-the-introduction-to-neural-networks-we-all-need-part-2-1218d5dc043. [Accessed: 27-May-2017].
[41] “calculus - Derivative of sigmoid function.” [Online]. Available: https://math.stackexchange.com/questions/78575/derivative-of-sigmoid-function-sigma-x-frac11e-x. [Accessed: 27-May-2017].
[42] “Artificial Neural Networks: Mathematics of Backpropagation (Part 4) — BRIAN DOLHANSKY.” [Online]. Available: http://briandolhansky.com/blog/2013/9/27/artificial-neural-networks-backpropagation-part-4. [Accessed: 27-May-2017].
[43] “What is the unit step Function in Artificial Neural Network? - Quora.” [Online]. Available: https://www.quora.com/What-is-the-unit-step-Function-in-Artificial-Neural-Network. [Accessed: 27-May-2017].
[44] “The most common activation functions: (a) step function; (b) linear... - Figure 5 of 13.” [Online]. Available: https://www.researchgate.net/figure/259843708_fig9_The-most-common-activation-functions-a-step-function-b-linear-function-c-sigmoid. [Accessed: 29-May-2017].
[45] “Artificial Neural Network.” [Online]. Available: http://www.saedsayad.com/artificial_neural_network.htm. [Accessed: 27-May-

2017].
[46] “CS231n Convolutional Neural Networks for Visual Recognition.” [Online]. Available: http://cs231n.github.io/neural-networks-1/. [Accessed: 29-May-2017].
[47] “Python Programming Tutorials.” [Online]. Available: https://pythonprogramming.net/cnn-tensorflow-convolutional-nerual-network-machine-learning-tutorial/?completed=/convolutional-neural-network-cnn-machine-learning-tutorial/. [Accessed: 29-May-2017].

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Applying machine learning algorithms on multi-sensor applications

Richting: **master in de industriële wetenschappen: elektronica-ICT**
Jaar: **2017**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Kelher, Tom

Datum: **6/06/2017**