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

Masterproef
learning algorithms for sensor interpretation on an exo-skeleton

Promotor :
Prof. dr. ir. Bart VANRUMSTE

Promotor :
dr. LUDO CUYPERS

Copromotor :
dr. TODOR GERGANOV

Ruben Bonné

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

Gezamenlijke opleiding Universiteit Hasselt en KU Leuven

KU LEUVEN | universiteit hasselt

KU LEUVEN | universiteit hasselt

2016•2017
Faculteit Industriële ingenieurswetenschappen
master in de industriële wetenschappen: elektronica-ICT

Masterproef

learning algorithms for sensor interpretation on an exo-skeleton

Promotor :
Prof. dr. ir. Bart VANRUMSTE

Promotor :
dr. LUDO CUYPERS

Copromotor :
dr. TODOR GERGANOV

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

universiteit hasselt | KU LEUVEN

Foreword

First, I want to thank my promotors for giving me this chance to develop myself and support me along the way. They gave valuable information and hints at the right times while giving me the freedom to walk my own path.

Second, I want to thank my volunteers that provided me with the data for their patience and comprehension.

Last, I want to thank my family and friends for supporting me along the way. Their ideas and vision on certain points in the thesis were invaluable to me.

Summary

List of Tables

List of figures

Abstract (Dutch)

COMmeto werkt samen met 7 andere partners in een Europees project genaamd Axo-Suit aan de ontwikkeling van een assistief exo-skelet voor oudere personen. COMmeto verzorgt de software architectuur. De manuele afregeling (calibratie) van de HID (*Human Intention Detection*) sensoren op de arm van het exo-skelet is tijdrovend en bovendien gebruikersafhankelijk. Deze masterproef heeft als doel een *machine learning* algoritme te ontwikkelen a.d.v. een zelfbedachte leerperiode die bewegingen in het sagittale vlak kan detecteren en classificeren om zo automatisch de HID-meting op het exo-skelet af te leiden.

Om een *machine learning* algoritme te ontwikkelen zal eerst de nodige data die van de HID-sensoren komen overgedragen worden naar het werkstation. Daarnaast wordt er ook nagedacht over oefeningen die nuttige info kunnen bevatten voor verscheidene kenmerken eruit te halen. Naderhand zijn verschillende *machine learning* algoritmen getest op de verzamelde data van verschillende personen om de bewegingen te classificeren.

Uit de resultaten kan men concluderen dat na een korte leerperiode de verschillende bewegingen geclassificeerd kunnen worden. Om zo goed mogelijke resultaten te leveren en om de arm van het exo-skelet zo comfortabel mogelijk te maken, zal er voor elke persoon een ander model gecreëerd moeten worden. Een andere belangrijke factor is hoe goed de sensorbanden bevestigd zijn rondom de gebruikers arm.

Abstract (English)

COMmeto, active in software architecture services and software development, is involved together with 7 other partners in a European project called Axo-Suit to develop an assistive exo-skeleton for elderly people. COMmeto is responsible for the software architecture. In the case of the arm of the exo-skeleton the adjustment of the exo-skeleton to a person is carried out manually which takes a long time. This thesis focuses on the development of a machine learning algorithm to detect and classify the different movements in the sagittal plane and on the machine learning algorithms to adjust the exo-skeleton to the user.

To develop the algorithm, the data used to adjust the exo-skeleton was defined to determine which movements are required. Additionally, the transfer of this data from the human intention detection (HID) to the workstation was solved. Furthermore, different algorithms were tested on the gathered data from different persons to carry out the tuning and classify the movements.

Based on the results it can be stated that after a short learning period the various movements can be classified. To ensure the best results and to make the exo-skeleton arm as comfortable as possible, a different machine learning model is applied for each user. Another important factor to note is how well the sensor belts are attached to the user's arm.

1 Introduction

1.1 Situation

COMmeto situated in Ham focuses on the development of software architectures for electronic sensor actor systems for application in a wide range of areas such as eHealth, eCare and engine control. COMmeto is part of a European Union AAL (Active & Assisted Living) project called AXO-SUIT together with three universities and 4 other companies in 4 different countries. This joint project develops an assistive exoskeleton for the elderly to assist them in their daily tasks and social activities despite their limitations. In this project, COMmeto provides the exoskeleton with the software for the moving parts to assist human movement. The full-body exoskeleton is comprised of three major parts.

- The lower body
- The upper body
- The spine

This thesis will essentially focus on the right arm of the exo-skeleton. The structure of the right arm will be discussed in one of the following chapters.

What every major part consists of is seen in figure 1.

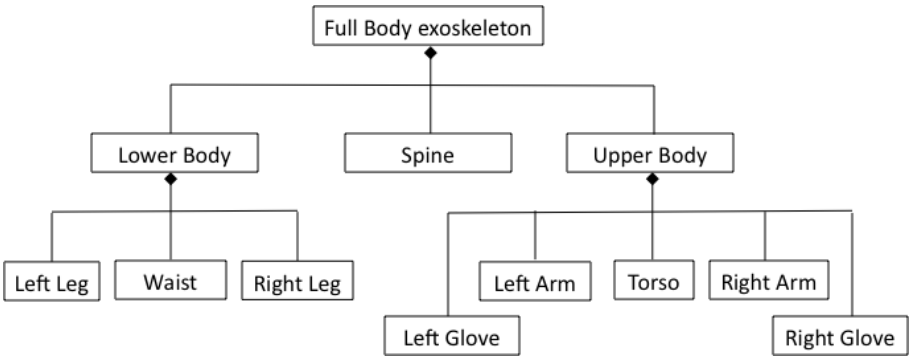


Figure 1 UML diagram of the full body exoskeleton [1]

1.2 Issue

Up until now, there exists a test-rig prototype arm of the exoskeleton that can follow the human's movement. The system of the test rig's arm consists of two force sensitive resistors located at the wrist of the exo-arm. The issue with this test-rig prototype is that it is too simple and cannot detect the movement intention of the person wearing the exoskeleton but can only follow the movement. At the end of the project, the exo-arm needs to assist the movement of humans, not only follow it.

Another issue is entering the parameters for calibrating the exoskeleton-arm actuators to fit the exo-arm to a specific user. Every human is different and thus the parameters will be different for every user to make the exoskeleton feel comfortable for everyone. Until now the calibrating and fitting of the exo-arm happens manually. This is a tedious task and needs to be commuted into an automatic

task. The machine learning algorithm being implemented will realize this. The exoskeleton will be of limited use and will feel uncomfortable to the user as long as these problems are not solved.

1.3 Goal

The goal at the end of this thesis is to develop a learning period for the exoskeleton arm so that after a few learning exercises, the arm of the exoskeleton will recognize certain movements in the one-dimensional rotation plane of the elbow and react accordingly.

This will be realized by:

- First, developing the data acquisition boards so the necessary data can be gathered.
- Secondly, gathering useful data from different persons to help the machine learning algorithm
- Third, preparing the gathered data and experiment with different algorithms
- Lastly, optimizing the algorithm

1.4 Material and methods

In a first step, I make a study of the specifications and software that are already available for the exoskeleton arm. This is particularly a literal study about the software architectures and the hardware of the exo-arm. This information helps to understand and to improve the current setup.

The relevant research questions are:

- What programming environment to use to make a machine learning algorithm?
- How to simulate and validate the results?
- Which sensors are present or absent in the exo-arm?
- Which parameters are relevant to the algorithm?
- How to communicate with the sensors for gathering data?

In the next step data is gathered by asking different persons to perform various exercises with the exo-arm. Next, the machine learning algorithm uses the gathered data. The algorithm will first be developed in MATLAB and later converted to java.

In parallel the different machine learning algorithms are being researched. The theoretical research of these algorithms provides inspiration to adjust an existing algorithm to the needs of this project

The relevant research questions concerning the algorithms are:

- Which machine learning algorithms exist?
- What are useful parameters to the algorithm?
- Which algorithm is the best suitable one for this application?
- Does the data need to be transformed?

In the last step, the human arm movement is being analyzed to determine which exercises are needed in the learning stage of the algorithm. This study can give inspiration to use new sensors that are currently not available on the prototype.

The relevant research questions concerning the human movement analyses are:

- Which exercises contain useful data?
- How many times does an exercise need to be carried out to give accurate data?
- How many people are needed to carry out these exercises?

The combination of literal studies on previous mentioned subjects forms the basis for this thesis.

Following this research, the practical part of the thesis starts. First, the data needs to be available for capture. The data acquisition board itself exists but is not yet completely assembled. Therefore, my first task is to assemble the board. As a backup, more than one board is assembled.

Second, the present data needs to be transferred from the acquisition board to the workstation (laptop). Currently, the data is transferred to the user control module (UCM) and can be made visible on the workstation. The next step is to transfer this data from the UCM to the workstation and store it there. The training data is stored in several CSV-files. Every file has the data from one training exercise. These files are then used to develop a first machine learning model.

Third, a test environment is made to experiment with different machine learning algorithms to find patterns and interesting features in the training data. Following this, a first model is developed and evaluated. An excess of training data at the early stage is very likely. This creates the need to go back and forth between gathering data and developing models in the early stage of the implementation. The preferable programming languages to create machine learning models are MATLAB, Java and C++. For practical reasons the final implementation is in Java.

Last, the machine learning models that give good results are selected to do some further testing. At the end, the best model is selected and used in the exo-arm.

2 Literature study – AXO-SUIT arm

2.1 The right arm

2.1.1 Introduction

The exo-skeleton arm can be split up in three different parts:

- Shoulder
- Elbow
- Forearm

These three are the main components that represent a fully operational exo-arm. For the exo-arm we only take into consideration two degrees of freedom (DOF) for the human shoulder. Only 2 of the 3 movements (DOFs) of the glenohumeral joint are considered. The shoulder-girdle or scapula-thoracic movements (DOFs) are not actively supported and not taken into consideration. The first DOF is the flexion/extension movement of the shoulder (imagine yourself lifting a box from the ground upon a table with both hands). The second DOF is the abduction/adduction movement of the shoulder (imagine doing jumping jacks). Every DOF needs to be carried over into the AXO-SUIT design and is actively supported by the exoskeleton. The first DOF has been carried over as an electric motor in the design. When a joint is powered by an electric motor it will be referred as an active joint. The second one is a currently a passive joint, there is no electric motor, but will be active in the final prototype. For the current study only the first DOF (flexion/extension) is being considered.

The elbow has one degree of freedom, the flexion/extension movement of the elbow (imagine lifting dumbbells with your biceps). This DOF has been carried over into the design in the form of an electric motor. The forearm will be carried over as a sensor. No other DOF is added.

2.1.2 Sensors

Different sensors are needed to control the exoskeletons arm. These sensors are also needed to acquire data to supply the upcoming algorithm.

Table 1 Right arm sensors [1]

Location	Joint	A/P	Type	CN	Nr
Shoulder	Sagittal	A	Absolute encoder	ARS	1
			Torque sensor	ARS	1
	Coronal	P	Absolute encoder	ARS	1
Elbow	Sagittal	A	Absolute encoder	ARE	1
			Torque sensor	ARE	1
Forearm			1 DOF force sensor	ARE	3

The absolute encoders are used to measure the angle of the shoulder and the elbow joint. Measuring the orientation of the magnetic field of a magnetic position marker does this. This orientation will then be transformed into an analogue voltage.

The torque sensors can be used to supply data for the learning algorithm. It is still debatable if they are useful or not. A different sensor outside of the arm provides us with current measurements of the electric motors that can be used to give a rough estimate of the torque.

The force sensitive resistor (FSR) is used to detect whether the person wearing the exoskeleton wants to move his arm upwards or downwards (human intention detection). This is information is also useful to the learning algorithm.

In the new version of the exoskeleton arm, eleven sensors are added to detect muscle activity and movement intention. These sensors are split over two belts. Six sensors are present in one belt for the upper arm to detect muscle activity in the biceps and triceps, three sensors for each muscle group. The other belt consists of five sensors responsible for detecting muscle activity in the forearm's muscle groups. All these sensors together are used to register the human intention detection (HID).

The following figure represents a quickdraw of the two separate bands. The numbers of the sensors correspond on which channel they are attached.

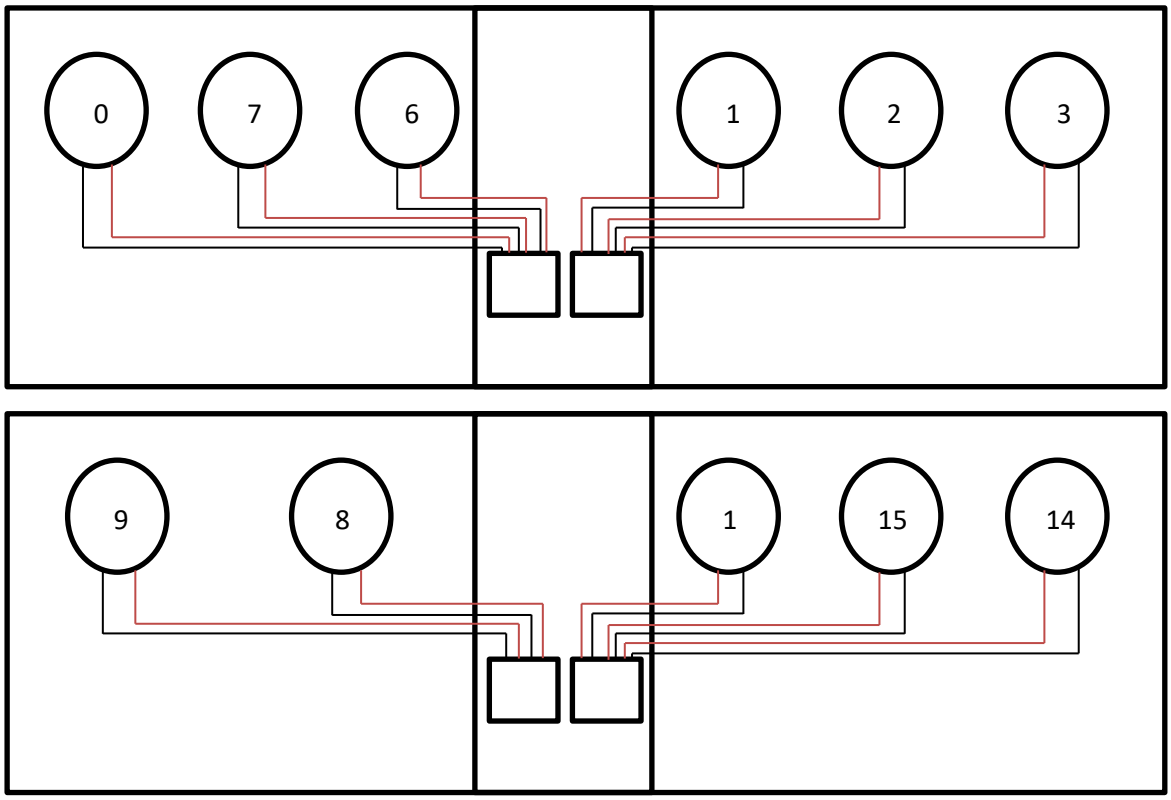
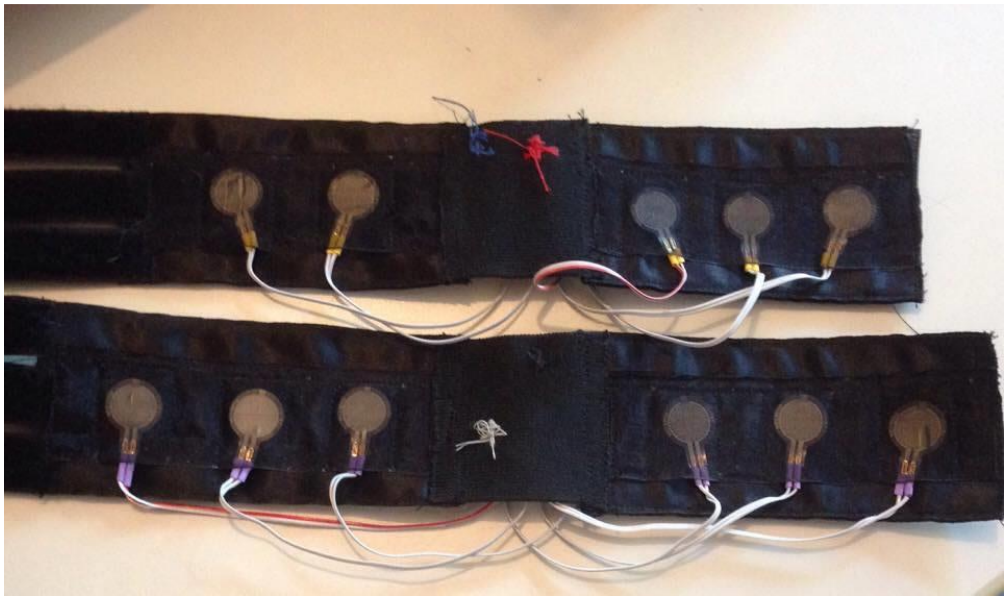


Figure 2 HID sensor belts

A picture of how the sensor belts look like is presented on the next page.



2.1.3 Actuators

The right arm has two actuators: two electrical motors, one inside the elbow joint and one inside the shoulder joint of the exo-arm and are actuated by ± 0-10V. CAN nodes are also installed at these places. They gather the data from the sensors and send it to the user control module (UCM).

Table 2 Right arm actuators [1]

Location	Type	Name
Elbow	EC motor	Maxon EC45
	CAN-node	
Shoulder	EC motor	Maxon EC60
	CAN-node	

2.1.4 Communication

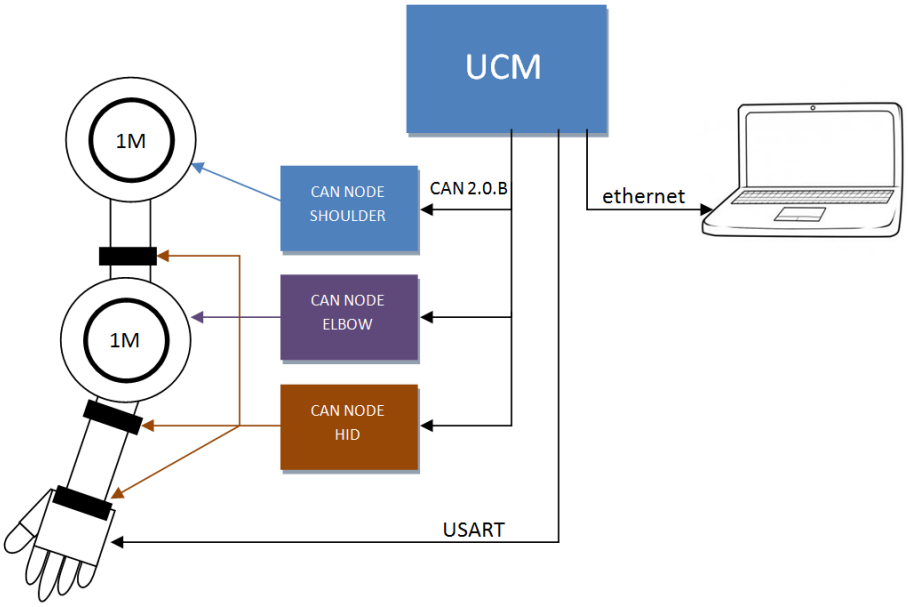


Figure 3 Upper body electric diagram [1] [2]

The shoulder and elbow CAN node have one motor which has two sensors attached to it. The first sensor is a Hall sensor to measure the current and speed of the motor. The second one, the encoder, is used to determine the angle of the joint.

The Human intention detection (HID) CAN node consists of three FSR bands. The first one located around the upper arm with six FSR's, the second one located around the forearm with five FSR's and the last one around the wrist with four FSR's.

Communication between the upper body control module and the sensors inside the right arm are provided by CAN-bus. CAN-bus is a good choice for this project. The advantages of CAN-bus complement the project very well. Some of the advantages that are of use are:

- Two wire connection
- Max speed 1Mbps
- Differential signal

The CAN-bus being a bus is advantageous because there is less space needed for wiring. Everything is connected to the bus (two wires) and thus saves space because less wiring is needed. If there was no bus system, every sensor and actuator had their own wires running everywhere.

The exo-skeleton arm will be used for following/assisting human movement. It is all-important that there is as little or rather even no bits/packages that contain false information.

CAN-bus makes use of differential signals which is excellent for this application, since the CAN-nodes are close to the actuator which generate electromagnetic noise. A dominant bit (logic 0) is achieved by a differential voltage in the two data wires and a recessive bit (logic 1) is achieved by the two data wires having the same voltage. Electromagnetic interference is exposed to both data wires and thus the difference between them stays the same, cancelling out the interference.

I won't go in to detail about how CAN-bus works as a protocol. If more information is required, I refer you to the literature list[3].

There is a USART bus present for the bionic glove, but this will not be discussed any further in this thesis.

The data collected from the exo-arm can be gathered on a workstation, in this case a laptop. The communication between the laptop and the user control unit (UCM) is established by an Ethernet connection.

3 Literature study – Machine learning

3.1 Introduction

Artificial intelligence (AI) is becoming more and more popular these days. AI can be used in many different fields like:

- Military (e.g. Big Dog)
- Driving (e.g. Google autonomous cars)
- Websites (e.g. adapted advertisement)
- Speech recognition (e.g. Siri)
- ...

Big Dog is a military project that can walk/run up to 10kmh in rough terrain. “BigDog has four legs that are articulated like an animal’s, with compliant elements to absorb shock and recycle energy from one step to the next”[4]. Machine learning is used in this project to help the robot learn how to not fall over in rough terrain with the help of various sensors.

Google autonomous cars is the same. Sensors are used to provide the necessary data to make the right decisions where necessary.

Adapted advertisement makes use of the data that is collected behind the scenes when you surf to different websites. This can range from the website itself to the time of day to even your general location[5]. The machine learning algorithm will factor in many different things to personalize advertisement to you.

Speech recognition is based on capturing the sound and sampling it at a high rate. These sample values hold the amplitude of the signal and a mix of different frequencies in a certain timeframe. Various calculations will be done on the data to convert it into useful data but I’m not going to go in detail here. The point is that all this will be converted and associated with a letter. Then a matrix of letters will be formed and that’ll be your output. The output is checked if it makes any sense by using a database that contains heaps of words that are commonly used in that language.[6]

Machine learning will develop even further and soon it will be all around us in different forms. In the next chapters, I explain the basics of machine learning and my thought process on the chosen learning algorithm.

3.2 What is machine learning

Machine learning is the science of "giving computers the ability to learn without being explicitly programmed"[7]. When a human is presented with a large amount of data, we can't scan through it instantaneously, let alone find some hidden secrets in the data. Fortunately, a computer can with the right algorithm. In machine learning we give the computer a heap of data, the computer will analyze this data and will find some hidden patterns in it. When found, these patterns can be used to generate code that will allow us to find these patterns in new data. Before we begin choosing an algorithm, we first need information. In the next subtitle, I'll discuss how to get started.

3.3 Different useful platforms

3.3.1 Weka

Weka is a useful tool to get started in machine learning. It can be used to load in certain data in .arff format. Weka has a collection of machine learning algorithms that can be used to analyze data. It also possesses visualization tools to visualize your results. Weka is written in Java and can be used in this project.

Weka is licensed under the Gnu General Public license and thus one must present ones source code to the public.

3.3.2 Spyder3

Spyder3 is a program within the Anaconda platform presented by Continuum Analytics. This is a handy program to get oneself started with machine learning in python. Python is a popular language to get started with programming/machine learning. This platform can be used in this project to do some rapid prototyping before coding in Java. Continuum and their programs are open-source so using this platform depends if you want to go open-source or not. Anaconda has no license but is then limited in functionality.

3.3.3 MATLAB

MATLAB is well known for its computational power with matrixes. Recently they added a useful toolbox that helps tremendously with machine learning. After putting the data in the right format, one can simply use the toolbox to quickly experiment with lots of algorithms.

MATLAB is used at first to experiment with machine learning and to develop the first algorithms that can be beneficial to the project, due to being familiar with it.

3.3.4 Eclipse

Eclipse will be used in this project to develop Java code. In the end, the machine learning algorithm must be present on the beaglebone black (BBB), which runs on java. Due to Java not having that many libraries and MATLAB having such a convenient toolbox the experimental phase will be done in MATLAB. Later when everything is in order, the developed MATLAB code will be converted to java code.

3.4 Different natures of learning

3.4.1 Supervised learning

In this method, a "teacher" that presents the computer with labelled training data assists the computer. The computer uses this training data to produce an inferred function[8]. This function can later be used to map new data. In ideal circumstances the algorithm correctly maps new data, even in unforeseen circumstances. This will depend on the inductive bias of the algorithm. As stated by Mitchell, T. : *"The inductive bias (also known as learning bias) of a learning algorithm is the set of assumptions that the learner uses to predict outputs given inputs that it has not encountered."*[9]

3.4.2 Unsupervised learning

In this method, the algorithm is left on its own. The only information that is presented to it is some unlabeled data. It is the algorithms task to find hidden structure in its inputs.

3.4.3 Reinforcement learning

Reinforcement learning differs from the previous two. This method makes use of a dynamic environment where as the other two make use of a controlled/semi-controlled environment, also known as a Markov decision process (MDP) [10]. The algorithm autonomous in its decision making. The algorithm must perform a certain goal in this dynamic environment but without a teacher explicitly telling it if the algorithm succeeded [6]. An example of this is autonomous cars.

3.5 Different machine learning algorithms

3.5.1 SVM

A Support Vector Machine (SVM) is a supervised learning model that is associated with learning algorithms that analyses data used for classification and regression analysis. All data points are viewed as n-dimensional vectors and are separated by (n-1) hyperplanes. e.g. when SVM is exposed to training data that is marked in two different classes, it builds a model that classifies new data in one of the two categories. It'll use one hyperplane to separate the two classes.

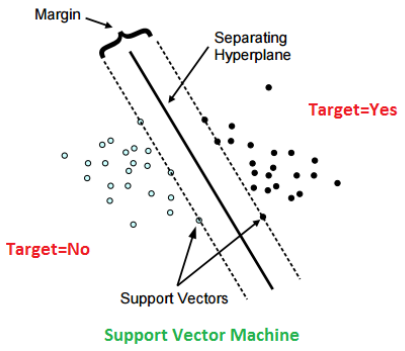


Figure 4 Support vector machine (picture take from DnI institute[11]**)**

Even though it uses only one hyperplane to separate two classes, it can still draw an infinite amount of lines for that one hyperplane. The most common separator will be the maximum-margin hyperplane. The hyperplane is chosen in such a way that the distance between the nearest data point on each side is maximized.

Previous example was a linear example but SVMs can efficiently perform a non-linear classification using what is called the kernel trick. Their inputs will be implicitly mapped into high-dimensional feature spaces. This will not be further discussed in the thesis.

3.5.2 Decision trees

Decision trees are one among the most popular and powerful methods. There are different sorts of decision trees in existence and some of them are explained below. But first we need to know what a decision tree is.

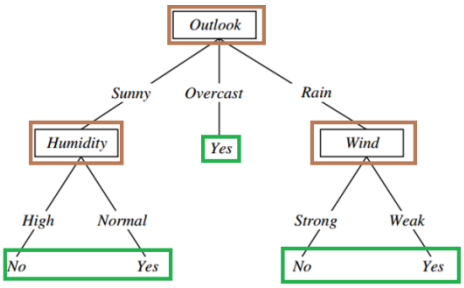


Figure 5 decision tree (picture taken from Tom M. [12]**)**

A decision tree consists of two types of nodes[13]:

- Leaf nodes: these are leaves at the end of a "branch" of a tree. They describe what is done if this route is followed. E.g. In the previous figure, we see that if the outlook is overcast we play tennis. The branch is at its end and made a decision.

- Internal nodes: Internal nodes ask a question based on variables that are present in the supplied data. By asking more questions the tree branches out more and splits the data even further.

Now that the structure of a decision tree is known, it is possible to talk about the different types of trees.

- Classification trees: this type of tree is used when variables have a finite set of values. E.g. figure 4. The internal node windy has a finite set of possibilities.

- Regression trees: used when internal nodes have continuous values like real numbers for example.

It is also possible to combine these two into a classification and regression tree (CART).

A typical characteristic of a decision tree is that they are a bit fragile. If for example a small change in the tree happens, it can have a big difference in the outcome. Another way of making changes is to change the training data. When training data is changed, the model of the tree is also changed. Therefore, they have a high variance that means they can be less accurate.

There are ways to improve the accuracy of decision trees:

- **Bagging**: also called bootstrap aggregation[14] is an ensemble method that combines the predictions of multiple machine learning algorithms to make an even better prediction than an individual one. As mentioned earlier, decision trees have a high variance in their outcomes. Bagging is one way to solve this problem. Instead of taking the result of one decision tree, the results of many are used. Once all the results are known, an average is taken resulting in the final outcome.

- **Random forest**: this method is an improvement on the earlier mentioned bagging method. A decision tree always takes the variable to split the data with minimum error. Now, random forest doesn't care about that. It will change the variables so that different sub-models are used to split the data. Now multiple different models are formed, meaning that the models are less correlated. The ensemble method is even more powerful than it already is with models that are uncorrelated or weakly correlated.[14]

- **Boosting**: the purpose of boosting is to take multiple learners (weak learners) each with a poor error rate and combine them to make one strong learner. The first weak learner is trained on a set of patterns that are randomly picked from the given dataset. The first learner classifies some of the patterns incorrectly. The second learner will be given a different training set. The new training set consists of the patterns that were classified incorrectly and contains new data that has a higher probability to those patterns. Now the incorrectly classified patterns occur more. This means that patterns that were difficult to classify the first time are now easier to classify correctly. This process goes on and on until there are enough weak learners to make one strong learner.[15]

3.5.3 Naïve Bayes

This algorithm is based in the Bayes Theorem that assumes that the chosen variables are independent from each other. For example, "a fruit may be considered an apple if it is red, round, and about 10 cm in diameter"[16]. The main advantage of this algorithm is its simplicity. When we have certain data, we want the best hypothesis for it. This algorithm does just that. By calculating the posterior probability of a hypothesis, the algorithm can decide which hypothesis (h) is the best for the data (d).

$$P(h|d) = \frac{\mathrm{P(d|h)} * \mathrm{P(h)}}{\mathrm{P(d)}}$$

where[17]:

- P(h|d) is the probability of the hypothesis h given data d. Also called posterior probability.
- P(d|h) is the probability of the data d when hypothesis h is true.
- P(h) is the probability of hypothesis h being true regardless of the data.
- P(d) is the probability of the data regardless of the hypothesis.

This algorithm can be used for two or multiple classes' classification. It's a swift algorithm because of the simple calculations. It does a wonderful job when presented with independent data but this is not possible for most cases in the real world. Even though data isn't fully independent from each other the algorithm does surprisingly well.

3.5.4 k-NN

The k nearest neighbors or k-NN for short is a simple machine learning algorithm that looks up the k (k is a numerical constant) nearest data points to the new data point. k-NN does not have a model; it stores the complete training data. This algorithm doesn't really "learn". The algorithm can be used for the classification of data as well as regression.

- Classification
  To classify new data points the algorithm calculates the distance between the new data point and the k nearest data points. This distance can be calculated in several ways but one of the most standard is the Euclidean distance. The Euclidean distance can be calculated with the following formula:

$$d(x, xi) = \sqrt{\sum_{1}^{j} (x_j - xi_j)^2}$$

  where:

  x is the new data point
  xi is an existing data point
  j is the amount of input attributes

Other distance measures include: Hamming distance, Manhattan distance, Minkowski distance, ...

When the distances are known for these k nearest neighbors, it is time to determine to which class the new data point belongs. Every training data point (xi) is labeled in a class. To determine where the new data point belongs is determined by which class has the most data points nearest to the new point.

E.g.

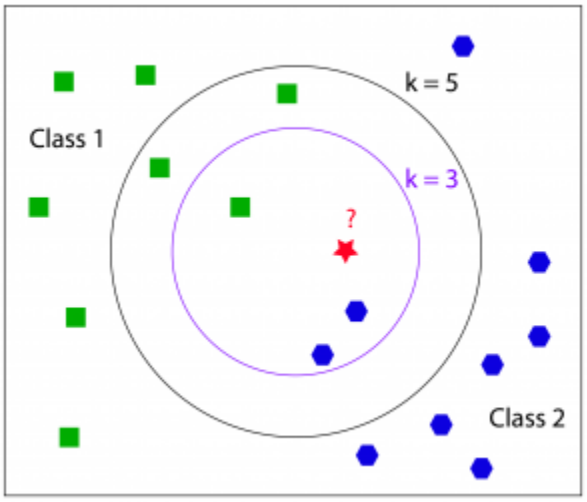


Figure 6 K-NN K=3 (picture taken from Fisher R.[18]**)**

In this example k = 3. The three nearest neighbors of the new data point are:
1xClass1 and 2x Class. B has the most points closest to the new data point. The algorithm will now decide that the new point belongs to class 2.

The value for k is also an important factor. K needs to be chosen wisely so that there can never be a tie between two classes. Some remarks on choosing the value for k are[19]:

- choose an odd value for k in an even class problem
- k cannot be a multiple of the number of classes

Another remark on this algorithm is that the search time goes up when there are more training data points. The algorithm calculates the distance from every point and chooses the k nearest ones[20].

- regression

When used as a regression algorithm, the value of the new data point is the average of the values of its k nearest neighbors. To make a more accurate prediction the values are weighted by their distance. A simple solution to determine the weight is 1/distance. This means that the closest neighbor has the most impact on the value of the new data point. The furthest neighbor has the least influence.

3.5.5 Linear regression

Regression is known in many fields, such as statistics, but can also be used as a machine learning algorithm. A machine learning algorithm wants to make the most accurate predictions as possible with minimal error. Linear Regression does just that. It was developed in another field but the purpose of it is to understand the relationship between input and output variables. This is essentially what we want to do in predictive machine learning. As the name implies, linear regression will assume that the relationship between input and output variables are linear.

There are two main categories in linear regression:

- Simple linear regression: In this method, the output (y) is dependent of only one input variable (x)

  $$y = a * x + b$$

  This type of regression is useful to understand how regression works. Most of the time it is impractical in the real world. It is rare to see an output only being dependent of one input in the real world.

- Multiple linear regression: The output (y) is now dependent of multiple outputs, hence the name of this method. This method makes a more realistic model of the world. As said before, most things are dependent on more than one input.

  $$y = a * x1 + b * x2 + \cdots n * xn$$

There are many forms of linear regression and have also been called different names. Below I'll explain some basic regression models.

- Ordinary least squares

  Ordinary least squares (OLS) estimates the unknown parameters (a and b in previous example) by first calculating the distance between the data point and the regression line. Then the distance will be squared (also called the squared error) and lastly all the squared errors will be summed together. OLS strives to make the sum of these errors as minimal as possible.

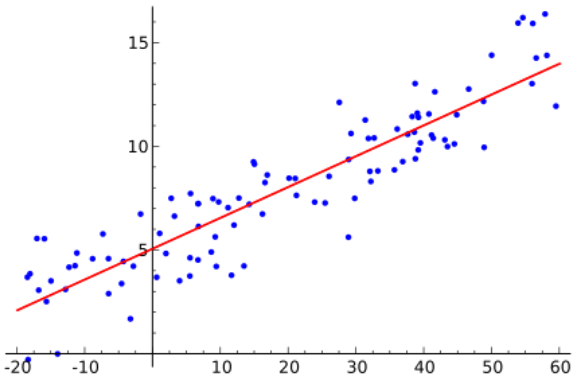


Figure 7 Linear regression (picture taken from Wikipedia[21]**)**

- Gradient Descent

  Another method is gradient descent. This method optimizes its coefficients by iteratively minimizing the error of the sum of the squared errors.

  A gradient points in a direction where if you take one step, you ascent proportional to the gradient. This will lead you to a local maximum, this is also called gradient ascent.
  Gradient descent does the opposite. If you were to take a step in the opposite direction of the gradient, you'll find a local minimum.

  First the sum of the squared errors is calculated for a pair of input and output values. After that a "learning rate" is implemented, this learning rate is a factor that'll point to the opposite direction of the gradient. By following the opposite direction, the way to a local minimum is found. The method will iterate as many times as possible with a step size proportional to the learning rate that was defined by the user.

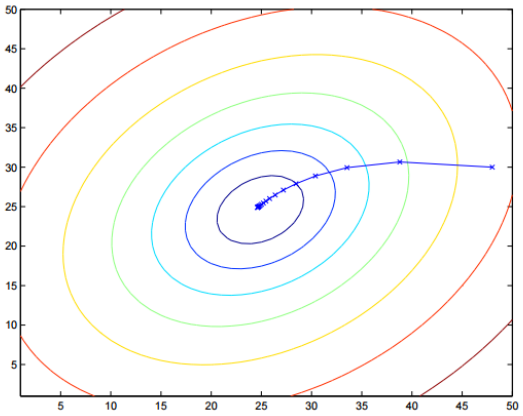


Figure 8 Gradient descent (picture taken from nasacj[22]**)**

- Regularization

  Regularization does not stop with only minimizing the sum of the squared error of the model on the presented data. It'll also reduce the complexity of the model by for example reducing the number of variables that are used in the model.

  Two examples of regularization are:
  Lasso regression[23]: reduces complexity by also minimizing the absolute sum of coefficients.

  Ridge regression[24]: reduces complexity by also minimizing the squared absolute sum of the coefficients.

  These two examples won't be discussed any further in this thesis.[25]

3.5.6 Logistic regression

Logistic regression is used for binary classification. In short, it calculates the probability of something and then classifies it as a "one" when the probability is high enough or vice versa. The name of this method is due to the use of the logistic function, which is used to calculate this probability.

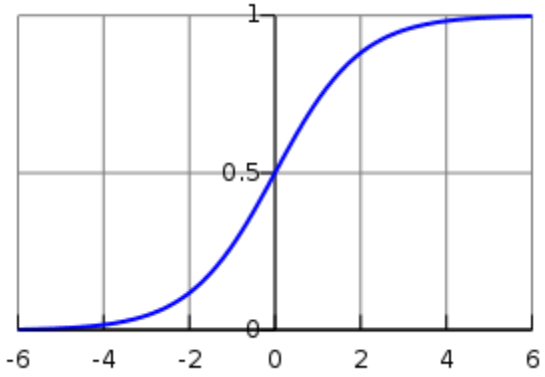


Figure 9 logistic function (picture taken from LISA lab[26]**)**

The logistics function is a S-curve. When x approaches +∞, the function approaches 1.

When x approaches -∞, the function approaches 0. This can be used to describe a probability in machine learning.

Another way of describing this function is:

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

This function tells us that the predicted output y is dependent of one variable x. In reality, it is possible that y is dependent on more than one variable. But for now, we'll hold it on one parameter.

Now how to go to a binary classification. The predicted output y is a probability that varies between zero and one. To go to a binary classification, we threshold this value at a specific point.

e.g.: logical 0 if y < 0.5

logical 1 if y > 0.5

Another neat feature of this algorithm is the ability to calculate odds.

An odd is a ratio between the probability of the event happening and the probability of the event not happening.

e.g. the probability of snow on day X is 80%.

$$\text{odds} = \frac{0.8}{1-0.8} = 4$$

This means that the odds of snow on day X is four times more likely than the odds of not having snow on day X.

Now to convert the logistic function[27]:

$$P(x) = \frac{e^x}{e^x + 1}$$

$$P(x) * (e^x + 1) = e^x$$

$$P(x) * e^x + P(x) = e^x$$

$$P(x) = e^x - P(x) * e^x$$

$$P(x) = e^x * \big(1 - P(x)\big)$$

$$\frac{P(x)}{1 - P(x)} = e^x$$

As you can see the left-hand side of the equation is now a ratio of the probability of the event and the probability of not the event. (odds)

$$odds = e^x$$

3.6 The thought process

The following chart will give you the basic thought process that I will follow throughout the thesis.

Figure 10 Machine learning process overview [28]

In every project, you'll first need to define your problem. Describe the problem you're having right now, give a motivation why it needs to be solved and then make an assumption on how to possibly solve it.[29]

In my case for example: The exo-arm of the exo-skeleton is uncomfortable for some persons. Every person moves their arm slightly different and the arm needs to adapt to this. This problem needs to be solved or the exo-skeleton won't feel right to the person. A possible solution for this is to apply machine learning. At the start, you'll have many different data attributes, but only a few may contain useful information. A good way to decide if data is relevant or not, is to give a motivation on why this data is useful and which benefits it can have. Data like current and speed of the motors, etc. will be important for the algorithm, these attributes will be used to refine the exo-arm's movement. Age won't be relevant because younger people can have muscle illnesses.

After you have described the problem and gathered data, it's time to analyze it. Analyzing data will help you better understand the problem. In this step, the data will be summarized, the number of data points and the data types are important here. It's also beneficial to convert some data attributes to different data types. Visualizing the data you have can inspire you to see new patterns with the naked eye.[30]

Next up is preparing the data. First, does the data cover everything? In this step, we need to be sure that all useful data is present. Maybe some data attributes couldn't be measured but need to be calculated (e.g. the speed of a motor can be calculated when you measured the output power and the torque of a motor). Some attributes may not even matter to one problem so it can be excluded

this time (but can be addressed in another problem). Excluding data can be as useful as including it. Taking too many attributes can give rise to over-fitting your algorithm later.

After that the data needs to be formatted. Formatting data means putting your data in the right place. At times you have data stored in databases but you want them in a text file or vice versa. Cleaning data is also important. Redundant information will be taken out and missing information will be added. Sampling data will help speed up your algorithm. Having too much data can be a burden. A smaller representative group of data is far more effective in testing environments.[31]

When the data is prepared, we need to define a test harness. A test harness contains test data where the algorithm will be tested against. The goal of the test harness is to quickly test different algorithms to see which performs better than the other. When multiple algorithms perform poorly on the test harness, it is a good thing to reconsider your test harness. This may depend on the lack of structure in the test harness. Some data attributes may need to be transformed to bring more structure so the algorithms perform better.

The next step is improving the chosen algorithms. After you tested some algorithms, only a hand few remain. These algorithms performed well but don't give the best results. Now is the time to fine tune them. You can do this by tuning the parameters of each algorithm. When you are tuning these parameters, you are making the algorithm more biased towards the test harness. Tuning can give good results but can also backfire. The more you tune, the more fragile it can become. This is referred as over-fitting your algorithm. Another way of increasing the performance is to ensemble multiple results into one.[32]

At last you report your results. This will include many "why did you do this?" questions. Can some algorithms still be improved? What are the limits of the chosen algorithm? and so on... When these questions are answered, then the work is done.

3.7 Preparing data

Preparing your data is one of the most important things in machine learning. If the data you work with is wrong or incomplete, then the algorithm won't give the result you were hoping for. There are several things you must check to guarantee that your data is prepared.

3.7.1 Select useful data

If you are selling product A to companies in different countries and you want to predict if the product will be successful with other companies in the same countries, then it is not useful to use data of product B for example. Carefully selecting your data so that nothing redundant is left is important for the algorithm. If you choose two parameters that are heavily correlated with each other, then we can assume that one of them is redundant to the algorithm.

3.7.2 Split dependent and independent parameters

This is not necessary but it is good to clearly see which parameters are independent and will contribute to predicting the dependent variables.

Table 3 Test data

Company	Name	Age	Salary	Manager
Philips	Peter		34780	No
Microsoft	Hans	52	54200	Yes
Sony	Vincent	42	41950	Yes
Apple	Jonathan	29		Yes
Nike	Ellen	24	31500	No
Adidas	Karel	49	47430	Yes

Note that this data is purely fictional. This data specifies if a person is a manager or not. The dependent variable here is Manager. Manager is dependent on the other four parameters to determine if the person is a manager or not.

3.7.3 Fill in missing data

Theoretically there is always a value to every parameter, but this isn't always the case practically. Sometimes the value is missing for a certain parameter. There are two ways to solve this problem: throw away the data or fill in the missing data. Throwing away data isn't the preferred way to go because you lose your precious data. The better way is to fill in the data but how?

A common and popular solution is to take the mean of that parameter. E.g.

The data in table 3 is not optimal. There are two missing values that need to be filled in. To solve this, we will calculate the mean of each parameter (column) that has a missing value and replace it with the calculated value. In this case, the missing values will be $\frac{52+42+29+24+49}{5} = 39.2$ for age and

41972 for the missing salary. Note that the divider is the number of the amount of known values and not of our total data instances.

Table 4 Test data filled in

Company	Name	Age	Salary	Manager
Philips	Peter	39	34780	No
Microsoft	Hans	52	54200	Yes
Sony	Vincent	42	41950	Yes
Apple	Jonathan	29	41972	Yes
Nike	Ellen	24	31500	No
Adidas	Karel	49	47430	Yes

3.7.4 Encoding categorical parameters

Algorithms can't work with text parameters. They can be useful to the algorithm but they need to be transformed to numerical values to be of use. This'll be explained by the data below

Table 5 Test data with countries

Country	Company	Name	Age	Salary	Manager
Belgium	Philips	Peter	39.2	34780	No
USA	Microsoft	Hans	52	54200	Yes
Italy	Sony	Vincent	42	41950	Yes
USA	Apple	Jonathan	29	41972	Yes
Germany	Nike	Ellen	24	31500	No
Germany	Adidas	Karel	49	47430	Yes

Now the countries are known of each company (this data is still fictional). The countries and manager are categorical data and need to be encoded, only numbers are needed in our equations.

This can be done by "labelling" the data. E.g.

Belgium = 1, Germany = 2, Italy = 3, USA = 4

No = 0, Yes = 1

Table 6 Test data encoded

Country	Company	Name	Age	Salary	Manager
1	Philips	Peter	39.2	34780	0
4	Microsoft	Hans	52	54200	1
3	Sony	Vincent	42	41950	1
4	Apple	Jonathan	29	41972	1
2	Nike	Ellen	24	31500	0
2	Adidas	Karel	49	47430	1

Now this is something that an algorithm can work with.

3.7.5 Split data into training set and test set

The algorithm that will be created needs to be trained on your data, but we don't want to feed it all our data or else there is nothing left to test. It is important to split the data that is available but what is a good ratio?

A good ratio for training/test set is 80/20. 80% of the data will be used to train the algorithm and the rest for testing. Of course, this is not set in stone but it is the most preferred ratio. The ratio can go up to 60/40 but never a higher percentage for the test set. If the algorithm is under trained then it'll give the wrong results.

3.7.6 Feature scaling

The last thing to do is to scale all the parameters so that none are overshadowed by others. Many machine learning algorithms make use of the Euclidean distance between data points. When one parameters range is between 10 to 50 and another one is ranging between 10 000 to 80 000, it's pretty much obvious that the first parameter won't have any effect to the Euclidean distance. The Euclidean distance is calculated by the formula stated below.

$$Euclidean\ distance = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

To give an example I'll use the data presented earlier. Data point Hans and Ellen will be used in this example.

$$Euclidean\ distance = \sqrt{(52 - 24)^2 + (54200 - 31500)^2}$$

$$Euclidean\ distance = \sqrt{784 + 515290000} = 22700$$

As you can see the parameter age is almost not affecting the outcome. It is overshadowed by the parameter salary.

There many ways to do this but the two that are commonly used are:

Table 7 Two ways of feature scaling

Standardization	Normalization
$X_s = \frac{X - mean(X)}{standard\ deviation(X)}$	$X_n = \frac{X - \min(X)}{\max(X) - \min(X)}$

Standardization makes use of the mean and standard deviation of your dataset to make the new range and is also the most popular.

3.8 Evaluate algorithms

Evaluating algorithms can be done in different ways but it's important to always define a test harness before every test/experiment. The test harness will consist of what data you'll use for training and testing your algorithm, as well as how you'll measure their performance [33]. The algorithms that'll be created later in this thesis will focus on classification of movements. Performance of classification is based on accuracy but can also be evaluated via confusion matrixes and others. The two afore mentioned performance measures are the ones mainly used in this project.

Accuracy will give us a general idea of how well an algorithm performs on the given data and if there is structure in the data. If some algorithms perform poorly while other don't then it's a good idea to drop certain algorithms. On the other hand, if all algorithms perform poorly then there is something wrong with the data.

The confusion matrix will give us insight were the algorithm makes mistakes. These mistakes can then be used to adapt. Data that was predict well is good to see but to improve algorithms, we'll have to know where it went wrong. By looking at the instances that were wrongly classified (not on the diagonal in the confusion matrix) we can see that for example class 1 was mostly predicted as class 2. Knowing these things can then help us understand our data and revise certain features that were used in the data or even add them.

3.9 Improving algorithm results

When you're not 100% satisfied with your results from the first batch, then it's time to improve them. Improving your results/algorithms can be done in several ways. For instance, it can be done by changing/adding features. Say that for example you have two classes which need to be separated but they hang closely together. You as the 'supervisor' know that these two can normally be split by a certain feature, take the mean for example. But the mean for the two classes is a bit too closely together, then it's a good idea to adjust that parameter so that the space between the two classes becomes clearer.

Another way of improving results is to take or make an even more advanced algorithm. Ensemble methods are a good example of that. Take for instance the decision tree. Alone, one decision tree can do so much. It'll have a decent result but the result can be a bit biased or the accuracy is not high enough to your taste. By using an ensemble method like bagging (bagging is making multiple decision trees with a slightly different view on the problem), the results are less biased and in some cases, an even higher accuracy can be achieved. Naturally there are more ensemble methods but bagging is used later in the thesis.

Algorithm tuning is also a way to improve results. After your initial results, you'll see that certain algorithms perform well on the problem but you want that tickle more. You can get that extra by tuning certain parameters of the algorithm that were used to make certain decisions. Take note that tuning parameters will make the algorithm more biased and fragile to other data which counters your intentions [34].

4 Literal study – Learning exercises

4.1 Introduction

In order for the algorithm to learn several movements, it needs some reference. First, there are several movements to calibrate the exo-skeleton arm. These movements will be used to determine the range of the parameters. There are mechanical stops present on the exo-skeletons arm but we don't want to rely only on them. After that there are several exercises that consist of different movements so that the learning algorithm can classify and predict several movements.

4.2 Calibration movements

The exo-skeleton arm has different sensors attached to it. In order to maximize performance and comfort, it'll be necessary to first calibrate the sensors for the user. With these movements, we want to determine what the sensor range is of every sensor.

The user will start in the most common position, the standing position. Here the user will let his arm rest near his body with an open hand. It's necessary to not flex any muscles in this position. This position is the most relaxed and therefore can be used to determine the minimum values. See figure 12 for a reference.

Afterwards the user will perform several biceps curls to determine the maximum values for each sensor. When performing the biceps curls it's necessary to flex both your upper arm and forearm. Since there will be the most force on the sensors while your muscles are flexed. Try to do as many movements as possible, try to turn your wrist from time to time and even open and close your hand while doing this. See the exercise figures for reference.

4.3 Learning movements

The focus of this thesis lies around the elbow of the arm in the sagittal plane and thus only movements that'll be possible in that plane will be used. This'll mostly be bicep curls but there'll be slight differences to determine if it's possible to detect the position of the wrist (neutral, supination and pronation) and the status of your hand (open or closed) with only using the HID-sensors. There are twelve exercises in total, six with the arm next to the body and six with a forward stretched arm.

Therefore, I'll only list the first six exercises. The six exercises when the forward arm is stretched are a copy of those with the arm next to the body.

Figure 12 arm next to the body

Figure 11 arm stretched forward

a) Exercise 1

The user starts in the standard position with his **hand open** and the **wrist** in the **neutral** position. Afterwards he will perform a biceps curl and hold the flexed position for two seconds before going back to the standard position.

Figure 13 Exercise 1 begin and end position

Figure 14 Exercise 1 middle position

b) Exercise 2

The user starts in the standard position with his **hand closed** and the **wrist** in the **neutral** position. Afterwards he will perform a biceps curl and hold the flexed position for two seconds before going back to the standard position.

Figure 16 Exercise 2 start and end position

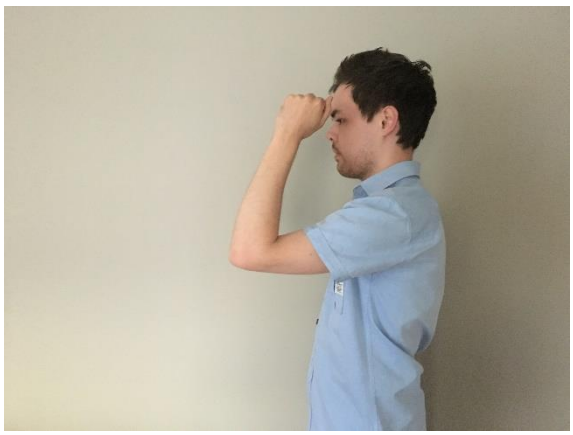


Figure 15 Exercise 2 middle position

c) Exercise 3

The user starts in the standard position with his **hand open** and the **wrist** in the **pronation** position. Afterwards he will perform a biceps curl and hold the flexed position for two seconds before going back to the standard position.

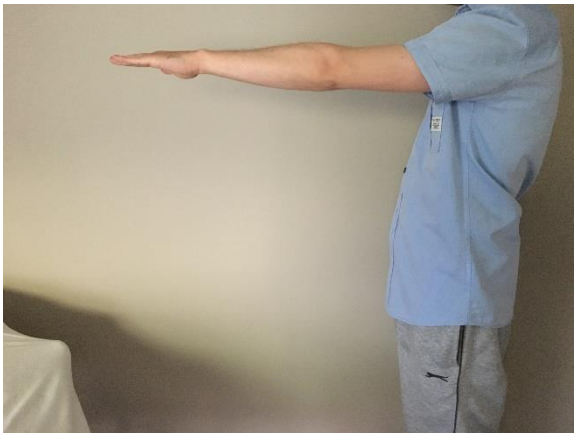


Figure 17 Exercise 3 begin and end position

Figure 18 Exercise 3 middle position

d) Exercise 4

The user starts in the standard position with his **hand closed** and the **wrist** in the **pronation** position. Afterwards he will perform a biceps curl and hold the flexed position for two seconds before going back to the standard position.

Figure 20 Exercise 4 begin and end position

Figure 19 Exercise 4 middle position

e) Exercise 5

The user starts in the standard position with his **hand open** and the **wrist** in the **pronation** position. Afterwards he will perform a biceps curl and hold the flexed position for two seconds before going back to the standard position.

Figure 21 Exercise 5 begin and end position

Figure 22 Exercise 5 middle position

f) Exercise 6

The user starts in the standard position with his **hand closed** and the **wrist** in the **supination** position. Afterwards he will perform a biceps curl and hold the flexed position for two seconds before going back to the standard position.

Figure 24 Exercise 6 begin and end position

Figure 23 Exercise 6 middle position

5 Data acquisition board

5.1 Introduction

The HID sensors are FSR sensors from the FSR400 family. The resistive value of the sensor decreases when force is applied to the sensor. These resistive changes need to be digitalized in order to create suitable data for the machine learning algorithms. The data acquisition board fulfils this role and will be explained in the next subsection.

5.2 Principle

The block diagram can be found in the figure below. It consists of sixteen sensor circuits that are connected to a multiplexer. The resistive changes are converted to a voltage signal with an opamp circuit which is then send to the multiplexer after going through a filter. Which sensor data is available at the output of the multiplexer will be decided by the CAN node. The board foresees sixteen sensors but only eleven sensors are used. The sensor values are later converted to a digital signal by 11-bit ADC's present on the CAN node. A datasheet of the specific sensor is provided in the attachments.

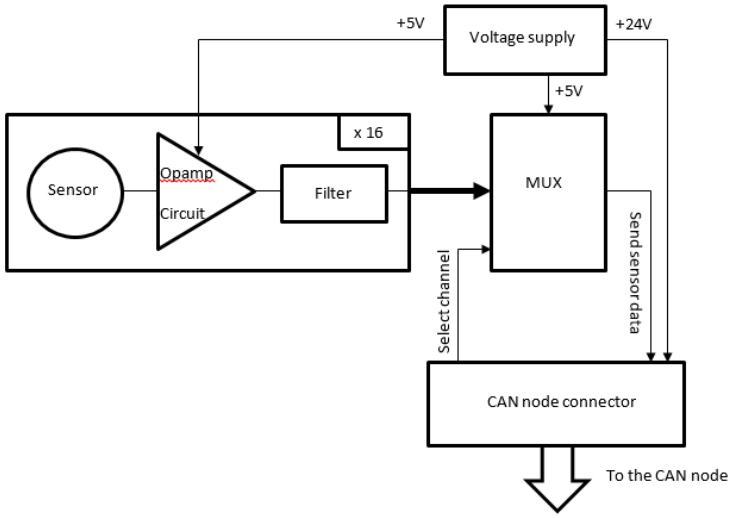


Figure 25 Block diagram HID-board

5.3 Calibration of the sensors

The HID-board is designed in such a way that the initial value (no pressure on the sensor) and sensitivity can be adjusted. This allows for on the fly adaptation and personal fitting. The possibility of changing these parameters is a huge boon to us. After doing several measurements on different persons, it became clear that these two parameters need to be adjusted for every person and even for different exercises.

Different persons have different muscles: density, placement, fat percentage, …. All these factors influence the pressure on the sensors which in their turn influence sensor range. Doing the same exercise but one where the wrist is in the neutral position and the other in pronation, even then the sensor range is different. Therefore, it is a necessity to first do some movements in advance, to measure the absolute highest pressure that can be put on the sensors. This'll allow us to define the sensor range.

6 Transferring data to work station

6.1 Introduction

In the previous chapter, the data was available at the CAN node. The next step is to send the data to the UCM via the CAN bus. The CAN bus itself was already available to me, thus the data was already present at the UCM. The last step is to get the data from the UCM to the workstation. This will be accomplished by implementing a CAN listener that will “listen” to a specific channel on the CAN bus where the data is present.

6.2 CAN bus

I won’t explain the CAN bus protocol in detail in this thesis but it is necessary to know the structure of the package that is sent to the CAN bus.

The maximum payload that can be send at once with CAN bus is 8 bytes. The value of one channel is 12-bit, so each channel occupies 1,5 bytes. It’s clear to everyone that sixteen channels can’t be sent in one whole package. To send all sixteen values, multiple packages are needed. In this case, we opted to send all sixteen values over four packages. Each package containing four channels with their respective identifier. The identifier has a value from 0 to 15 (the number of channels), which corresponds to 4 bits (0,5 bytes) for each identifier. To illustrate the complete package, see figure below.

Package1

ID 1	Value 1	ID 2	Value 2	ID 3	Value 3	ID 4	Value 4

Package2

ID 5	Value 5	ID 6	Value 6	ID 7	Value 7	ID 8	Value 8

Package3

ID 9	Value 9	ID 10	Value 10	ID 11	Value 11	ID 12	Value 12

Package4

ID 13	Value 13	ID 14	Value 14	ID 15	Value 15	ID 16	Value 16
4bit	12bit	4bit	12bit	4bit	12bit	4bit	12bit

Figure 26 CAN-bus data structure

6.3 CAN listener

The CAN listener which is present on the workstation needs to connect to the user control module (BBB) via ethernet to access the data on the CAN bus. To do that three things are needed:

- The IP address of the UCM;
- The port number;
- The CAN bus channel where the data is available.

When connection is established, the CAN listener will only listen to the specific CAN bus channel. Whenever a new package is present on that channel, the Can listener will see that change and take the values from it. It then stores the four values from each package in an array until it has sixteen values in chronological order so that the first element contains the first value, etc...

There is a mechanism implemented so that there can be no overlap between full package 1 and full package 2. A “full package” are the four sub packages together. So, for example: the last eight values from full package 1 and the first eight from full package 2 can never be together in one array.

There is also a second mechanism implemented, if there is even one package missing from a full package, then all other sub packages from the same full package will be thrown away. Using an incomplete full package will result in non-useable data.

When sixteen values from the same full package are available, then these values will be written away into a csv file for experimenting purposes. Later, in the final product these values won’t be written away but will be used in a real-time application. A good thing to note is that the data (16 packages) will be supplied at 50Hz.

7 Machine learning

7.1 Introduction

To get started with machine learning, the program of choice is MATLAB. The reason being is that I'm familiar with it and has predefined functions that are useful to get started.

In this chapter, I'll explain my thought process on how I did things and corrected myself where needed. The main goal is to get answers to several questions that'll determine if machine learning can be a real benefit to us. Several of those questions are:

- Can arm movements be classified? If yes how many movements?
- Are the machine learning models person specific?
- Can models be reused after taking off the suit?
- Which machine learning model performs the best?
- Which exercises are most useful for gathering data?
- How many times does an exercise need to be performed?

At the end of this chapter these questions will be answered by the experiments that were conducted.

7.2 Problem definition

When the user is performing the exercises it's easy for us humans to see when the user's arm is going up, down or standing still for example, but a computer does not have that luxury. Our task is to provide data for the computer to distinguish those specific events. This can be done by pre-labelling the data by holding a different button for each event. Each event will have a different label so that it's easy for the computer to know when an event started or ended. Afterwards the computer will be searching for patterns in the supplied data that is unique to an event to distinguish them.

The learning exercises mentioned in chapter 4 are chosen to make the different events easier to classify. The events that I'll be looking for are:

- The direction of how the arm is moving (upwards, downwards or standstill);
- The status of the user's hand (open or closed);
- The position of the user's wrist (neutral, supination, pronation).

In total, there are 24 different classes that we want to identify. I didn't immediately start to try going for all 24 classes but I split them up in the three parts above. First, I tried to detect the direction of the arm (four classes) and later added the status of the user's hand to it (two new classes, eight in total). After that I added the position of the wrist to it because this added the most classes and was harder to detect (three new classes, 24 in total). The number of new classes are always multiplied by the number of old classes.

7.3 Analyzing the data

An example of raw data from one of the twelve exercises is given in the following figure. There are five channels with a value of 1900+, these channels are put to their maximum value. This is a self-made decision because those five channels are not used. There are eleven sensors in total and thus only eleven channels are needed.

1 channel1	2 channel2	3 channel3	4 channel4	5 channel5	6 channel6	7 channel7	8 channel8	9 channel9	10 channel10	11 channel11	12 channel12	13 channel13	14 channel14	15 channel15	16 channel16
95	159	129	118	1962	1960	99	115	428	458	1962	1963	1963	521	864	547
94	158	129	116	1962	1961	102	115	438	460	1964	1964	1963	523	871	555
97	160	132	121	1962	1961	101	114	455	462	1961	1961	1961	521	871	550

Figure 27 RAW-data instances

Later, the channels that are not in use will be thrown away because they don't hold any valuable data. In the following subsections, I'll first explain how the data was split in different events. After that I'll explain how the data was prepared.

The raw data that is being supplied via csv files holds the values of the force sensitive resistors. As mentioned before there are eleven force sensitive resistors in use to detect muscle movement but all sixteen channels are imported. The values have a range from -30 to around 1980. The minimum value for each channel can be adjusted by a trimmer present on the data acquisition board. The trimmers are set so that there are no negative values. There is also a mechanism in the software to convert negative values as a failsafe. After all the values and labels are read in, they need to be separated. The values of the sensors are input data and the labels are output data.

The following two figures will give a better visual understanding of how the data looks like for each channel for one exercise (the exercise is repeated 5 times). Of course, the data will look a bit different for each exercise but it'll give a general idea.

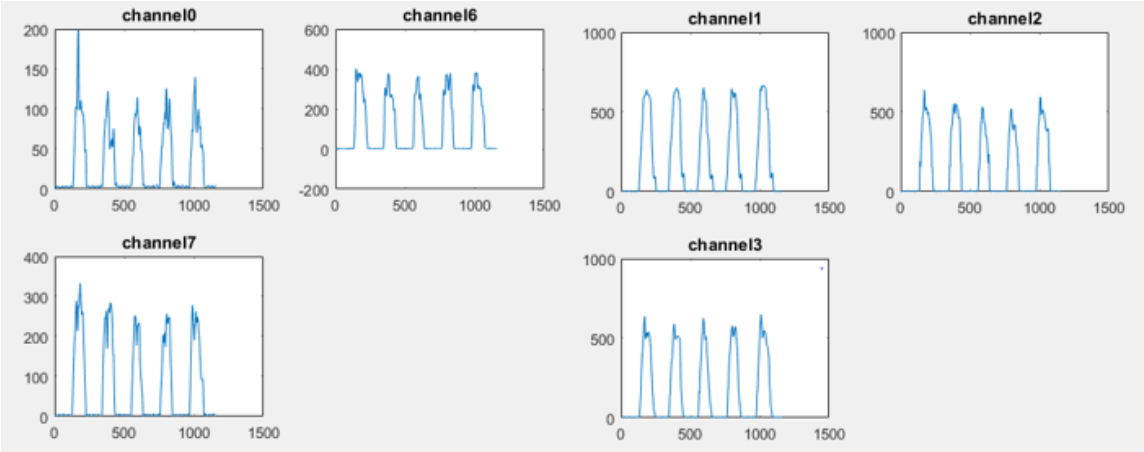


Figure 28 Values from sensors on the biceps (left) and the triceps (right)

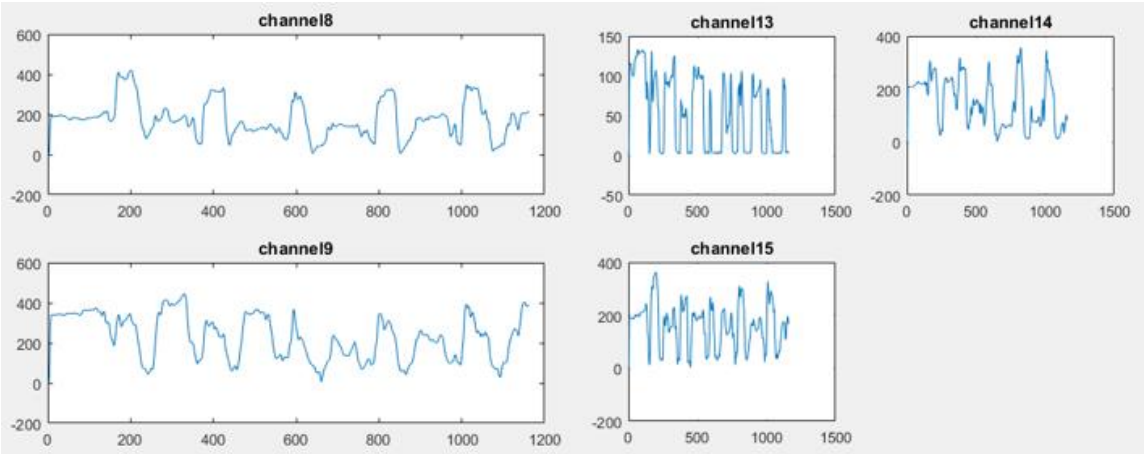


Figure 29 Values from sensors on the forearm ventral side (left) and forearm dorsal side (right)

A list of channel vs muscle below

Table 8 Sensor placements

Channel	Muscle
1	Inner side of biceps
2	Outer side of triceps
3	Middle of triceps
4	Inner side of triceps
5	Outer side of biceps
6	Middle of biceps
7	Outer ventral side of forearm
8	Inner ventral side of forearm
9	Inner dorsal side of forearm
10	Outer dorsal side of forearm
11	Middle of dorsal forearm

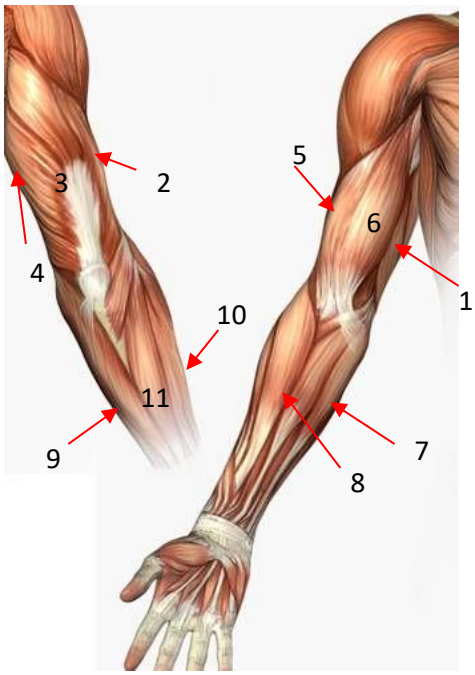


Figure 30 Placement sensors on human arm (Photo Credit: L Bucklin, licensed to About.com, Inc.)

7.4 Preparing the data

The first step in preparing the data is to segment it in events. One data row is only a fraction of an event and multiple data rows can describe a whole event. As mentioned earlier the pre-labelling defines the beginning or the end of an event. When this label switches in value, we know that an event has ended and a new one has begun. Now that it is known when an event has started or ended, it's time to segment them.

A single event now has multiple data rows to describe itself. Next up is calculating useful features from those data rows so that a single event can be described by a single data row. The features calculated in this case are:

- The mean value of every channel;
- The first derivative of every channel;
- The energy of every channel (sum of squares).

Note: the first derivative will be called speed from here on out for easier understanding and typing. It's not exactly the same but there is a connection between them.

In total, there are 33 features for a single event. Together with those features, there needs to be a label. The label will specify which event we're talking about.

Table 9 Classification labels

Labels	Direction only	Direction and hand status
1	Upward movement	Upward hand closed
2	Downward movement	Downward hand closed
3	Standstill down	Standstill down hand closed
4	Standstill up	Standstill up hand closed
5		Upward hand open
6		Downward hand open
7		Standstill down hand open
8		Standstill up hand open

The direction only column describes the four major events. The arm can be moving up or down or it can standstill at two different places. One where the arm is stretched and one where the arm is bend (after moving your arm upwards). Afterwards these four head events are split up into eight events which also describe the status of the hand. The labels for direction, hand status and wrist together are not listed. They are the same as for the direction and hand status labels but multiplied by three, that makes a total of 24 classes. The first eight labels are for the neutral wrist position, the second eight labels are for the pronation wrist position and the last eight are for the supination wrist position.

The figure below gives a representation of how the data is transformed from a lengthy table (5000+ instances for one exercise) to a scatterplot of the extracted features. This figure only shows the relation between two of the 33 features.

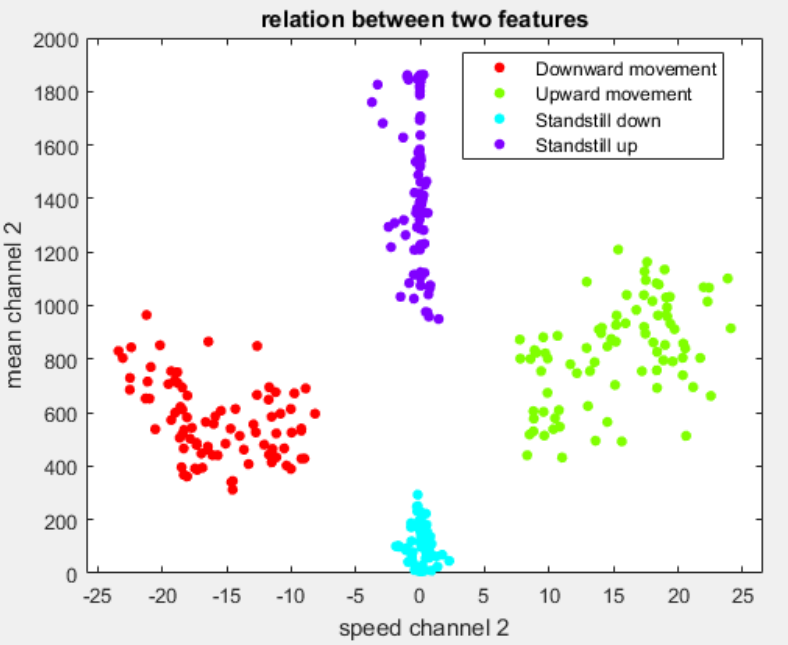


Figure 31 Scatterplot extracted features, direction labels

The figure describes the relation between the speed of channel two and the mean value of channel 2. The label used are those of the direction only. As you can see these two features hold valuable data to classify these movements.

Of course, there are more labels than those four. The following figure shows the same two features and their relation but with eight labels. The status of the hand has been added to the equation.

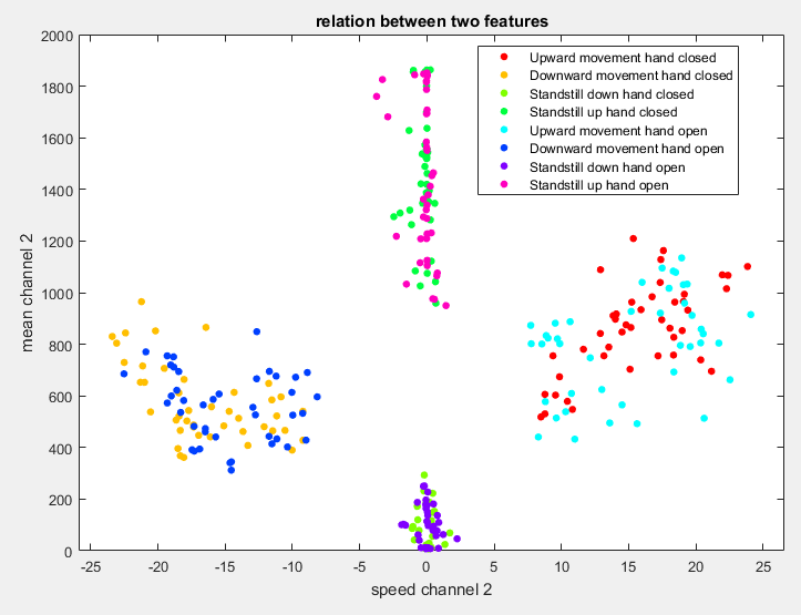


Figure 32 Scatterplot extracted features, direction and hand labels

With the addition of labels, the two features can't tell the difference between the status of the hand. This is to be expected and thus more features are needed. When the wrist labels are added, even more groups will be created and distinguishing those is more difficult.

These scatterplots will give you a first peak at how some algorithms will make their decisions to classify different labels.

7.5 Evaluating algorithms

7.5.1 List of used algorithms

In the following table, you'll find the algorithms that I've used to conduct the experiments.

Table 10 List of algorithms used

Decision tree
Bagged decision tree
Linear SVM
Quadratic SVM
Cubic SVM
Gaussian SVM
K-NN K = 1
K-NN K = 3
Cosine K-NN
Cubic K-NN
Weighted K-NN

This list can be divided in three major algorithms: Decision trees, SVM and K-NN. The specific names of the algorithms define the kernel that is used to calculate and help the classification process.

These algorithms were chosen because most of them are relatively easy to interpret and to see if different kernels have an impact on the accuracy.

In the following subchapters, I'll go over the different questions asked in the beginning of chapter 7. With asking all these questions, different experiments needed to be set up and will be clarified in their respectively parts. I'll only fully explain everything what I did once in the first question with one machine learning algorithm but the results of each model will be given. The questions that come afterwards will have a briefer and to the point explanation, mostly mentioning the differences between experiments and some mistakes that I've made. By doing this the explanation will be shorter and more to the point.

7.5.2 Can arm movements be classified?

This is the first question that needs to be asked. Can arm movements be classified? Or in other words: Is the current sensor setup equipped enough to detect all the movements we want to monitor? To answer this, I've done all the twelve exercises that I thought would be useful to the experiment and the result are represented below.

First, training and test data is needed to train and evaluate an algorithm. In this experiment, I've done each exercise about ten times to make sure enough data is present. From here I've split the whole data set in 80% training and 20% testing. I've made sure that each exercise is available in the training and test set.

Now to train the algorithm. For explanatory purposes, I've chosen the decision tree. The decision tree is very simple to understand and is also somewhat more relatable to human thinking but this is subjective.

In MATLAB, the training of a classification decision tree algorithm is made simple for you with a single function called fitctree. This function allows you to fit your data and many other things that you need to specify as parameters of the function.

One of those parameters is cross-validation, it is a must. Cross validation allows you to create models that are less prone to overfitting. Overfitting your data can be a potential trap in most cases of machine learning. The gist of it is that your model will look at too many details than needed. By using cross-validation you'll split your training data in K-folds, where K is given by the user. The parameter K will determine how many times you want to split your training data in equal parts. Then the model will train itself with K-1 parts of the training data and use that last part to test itself. This process will happen K times, so it tested itself against every fold of data. The final model will be an average of the K models. The following figure gives a representation of cross-validation.

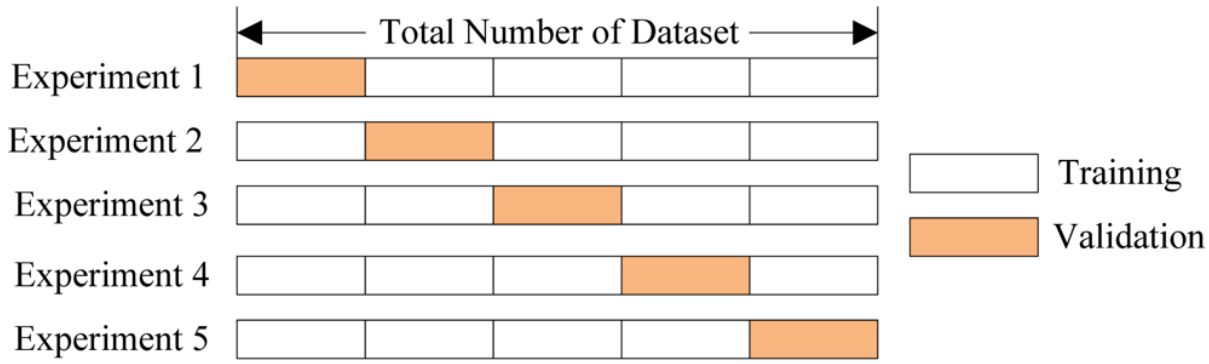


Figure 33 Cross validation (picture taken from PhilChang at stackoverflow[35]**)**

Once the model has trained itself, it is time to test it against the testing data that was left out of the training data. This is done by the predict function in MATLAB with specifying the model and the test data. To see the results, I've compared the actual labels with the predicted labels and calculated the accuracy of every algorithm. The numbers are given below for each algorithm.

Table 11 Accuracies all algorithms - experiment 1

Model	Acc direction lbl (%)	Acc direction and hand lbl (%)	Acc direction, hand and wrist lbl (%)
Decision tree	97.6	94.4	70.8
Bagged decision tree	100	96.1	95.2
Linear SVM	100	93.7	92.2
Quadratic SVM	100	95.8	91.6
Cubic SVM	100	96.5	91.6
Gaussian SVM	100	94	93.7
Knn K=1	100	97.9	97
Knn K=3	100	95	95.5
Cosine Knn	100	96.5	93.1
Cubic Knn	100	94.7	95.5
Weighted Knn	100	95	96
Average	99	95.4	92

When looking initially at these results, one can only say that this is a success. It is possible to classify all 24 classes but further analysis is required to see if these accuracies will last in certain situations.

Note: the last column with the wrist labels were added at the end of the thesis. At first there was a misunderstanding and were thus removed from the other experiments.

In the following figure the three decision trees for every label set is given to give a visual of which decisions were taken.

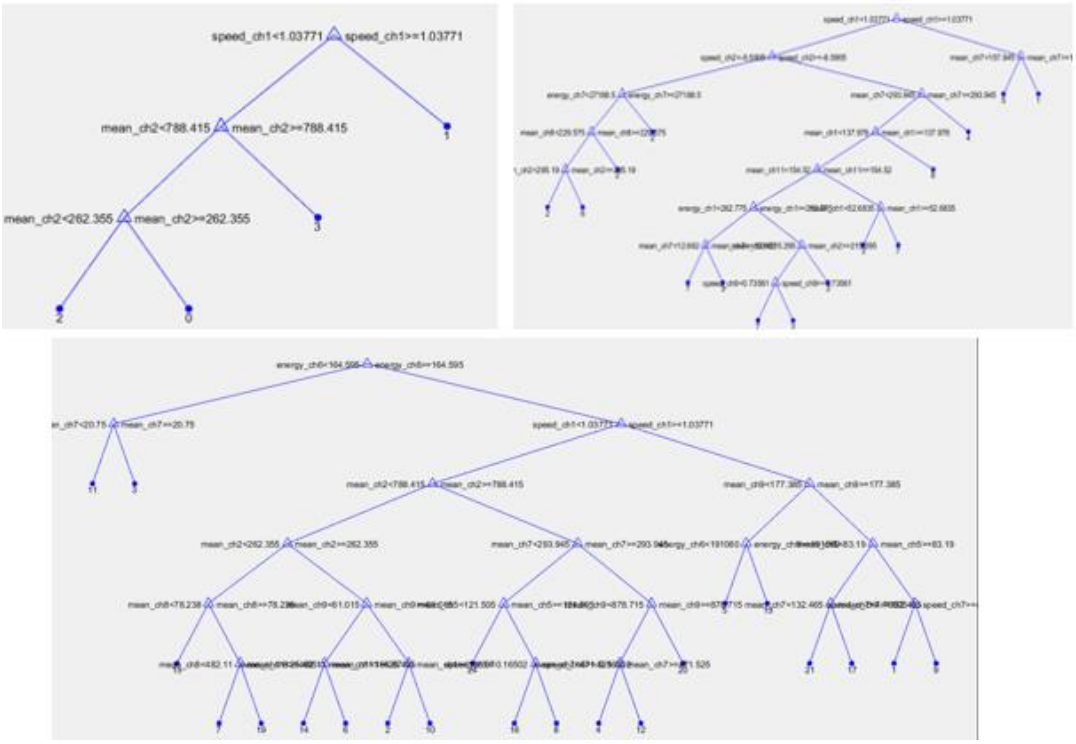


Figure 34 Decision tree models - experiment 1

Naturally this is only for this data set but the experiment was done over many data sets and there is still something to take away from it. For example, whatever data set it was certain features came back to classify data:

- Speed channel 1
- Speed channel 2
- Mean channel 2
- Mean channel 7
- Mean channel 8
- …

These features hold valuable data to split certain classes. By taking a closer look at these features and the scatterplots that correspond with them it's easier to see these decisions. Let's take a look at the top right decision tree in figure 20. This is the decision tree for direction of the arm and the status of the hand labels which classifies eight classes. By putting the corresponding scatter plots with the decision tree.

The figure below gives the first two decisions or nodes in the upper right corner of the decision tree.

If speed channel 1 > 1 the it is class 1 or 5 and if mean channel 7 > 158 then it is class 1. Class 1 and 5 both represent the upward movement of the arm. The difference between them is the status of the hand. In class 1 the hand is closed and in class 5 the hand is open. Therefore the first assumption that can be made is that the “speed” of channel 1 can classify arm movement and the mean of channel 7 can classify hand status. Naturally this is not true because then everything would be classified with only these parameters. But a general hypothesis can be made from this. Are the “speed” features commonly used as classification for arm movement and the mean parameters used for hand status?

To find this out I’ve studied some of the decision trees and came to a conclussion that this is the case.

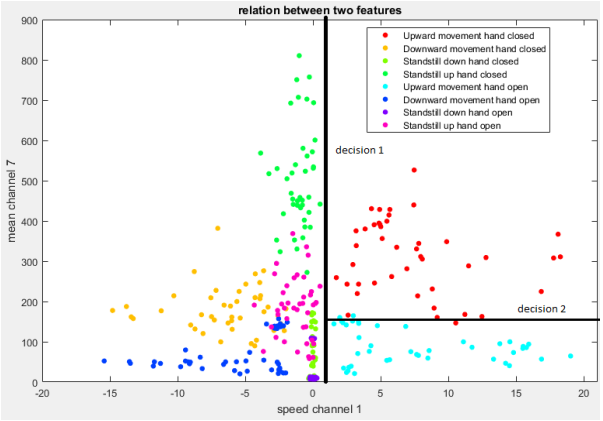


Figure 35 Example decision making

“Speed” of a channel is mostly used to split the different arm movements. In the test setup there are three arm movements: upward movement, downward movement and standstill. To separate these three, two “speed” decision have to be made, this comes back in every decision tree. An even bolder statement is that only or 95% of the time the “speed” features of channel 1 or 2 is chosen. To split the hand status in the movements, the algorithm makes use of mostly the mean features and energy features from the sensors attached to the forearm.

7.5.3 Are the machine learning models person specific?

This is an important question because this will translate in having only one model for everyone or having one model for every person. To answer this, I've collected data from several people and tested the training model of the first person against the whole dataset of another person. The results are shown in the following figure.

Table 12 Accuracies person specific experiment

Models of person 1	Acc person 1 (%)	Acc person 2 (%)	Acc person 3 (%)	Acc person 4 (%)
Decision Tree	97	41	52,7	36,3
Bagged Tree (30)	100	51	60,8	39,7
Linear SVM	96	46	57,2	32,9
Quadratic SVM	99	45	51,7	33,1
Cubic SVM	99	47,6	56,2	42,1
Medium gaussian SVM	99	49,2	59,3	39,7
K-NN K=1	100	41,7	61,2	44,4
K-NN K=3	96	42	60	41
Cosine K-NN K=3	96	42,5	57,4	41,9
Cubic K-NN K=3	97	40,7	56	41,9
Weighted K-NN K=3	100	42	60	42,3
Average	98,1	44,4	57,5	39,6

The accuracy drops tremendously when used for another person. This is an indicator that every person will need another model.

Accuracy is good method to see how precise a model can be but it doesn't tell where the model goes wrong. An excellent method to use for showing where the model fails is a confusion matrix.

I've made a confusion matrix for the decision tree model from person1 that contains the prediction of every data instance from the other persons in the figure below.

Table 13 Combined confusion matrix decision tree models

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8
Class 1	38	0	29	0	121	0	1	0
Class 2	0	22	58	0	0	107	1	1
Class 3	0	0	165	0	0	6	0	0
Class 4	12	36	22	55	3	11	4	46
Class 5	9	0	27	0	154	0	0	0
Class 6	0	4	50	0	0	135	1	0
Class 7	0	0	169	0	0	3	0	0
Class 8	8	28	25	41	4	15	2	67

The confusion matrix shows that the classification of the hand status doesn't go well in some cases. Class 1 and Class 5 represent the upwards movement of the arm with hand closed or open

respectively. Same for Class 2 and Class 6 or Class 3 and Class 7. Another case is that Class 4 and Class 8 are smeared out over all other classes.

The problem for Class 4 and Class 8 lies in that they define standstill while the elbow is bend. When bending your elbow and coming to a standstill you can sometimes have a peak in the values or the values start dropping. This phenomenon is caused by relaxing certain muscles or maybe even flexing the muscles when coming to a standstill. This can cause variations in certain channels which translates to changes in features, especially the "speed" features. With this change, the data points of class 4 and 8 which is standing still ("speed is almost 0") are sometimes classified as moving and sometimes not because it does not happen every time.

The figure below shows this phenomenon in the RAW-data of channel 2 (blue), a sensor on the triceps. The red lines represent the label switching. The data between the red lines are the data that is used to calculate the features of standstill_up, which means that you've bend your elbow and are now standing still. In this period data values can drop between 5 to 15%. When this drop is high enough, the first derivative ("speed") feature that is calculated is positive or negative enough to be confused with moving the arm up or down.

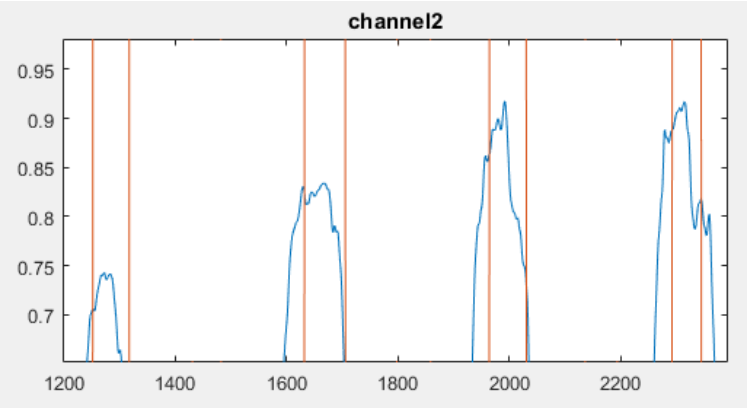


Figure 36 RAW-data smearing out explanation

A solution for this problem is to use another algorithm which is less prone to this phenomenon. This problem is more noticeable when using a decision tree than SVM or K-NN for example. Another solution is to use bagged trees, using multiple decision trees with a slightly different view on the matter. This makes the decision tree less prone to overfitting and this smearing out. A last solution that comes to mind is to use new features. One where the energy of the first derivative is measured, I define energy as the sum of squares. Big value changes in the first derivative mean that the arm is definitely moving and the energy of them will be even bigger. While smaller movements like what happens in this phenomenon are separated from them because their energy is way less than that of bigger movements.

The hand status problem lies with the mean features. In most cases mean features are used to separate the data points of Class 1 and 5 or the other pairs (2 and 6, 3 and 7, 4 and 8). A possible solution for the hand status problem is converting the absolute mean values to relative mean values. If the sensor belts are attached in a slightly different position or the pressure on the sensor belts is different, then the absolute values are different. This can cause problems for the models because they based their decisions on other absolute values.

A small conclusion with these results is that every person needs a different model. The possible solutions mentioned still need to be tested if there is any improvement. When those results are available I'll be able to give a definite answer.

7.5.4 Can person specific models be reused after taking off the suit?

After the results of the last experiment this would be a logical question to ask. The answer to this question will definitely be useful. The outcome will determine if the user who buys the exo-skeleton needs to train his suit one time or every time he wants to use it.

a) Person 1

The sensor belts are fitted to the first person so that they feel more comfortable and hopefully have good results. The accuracy of each dataset against the models of one specific dataset is given below.

Table 14 Person1 session specific accuracies

Models session 1	Acc ses 1	Acc ses 2	Acc ses 3	Acc ses4	Avg ses 2-4
Decision Tree	94,4	71,7	59,5	70,3	67,2
Bagged Tree (30)	96,1	70,8	82	80,5	77,8
Linear SVM	93,7	48,9	81,3	69	66,4
Quadratic SVM	95,8	52,4	77,4	69	66,3
Cubic SVM	96,5	55,7	76	71,4	67,7
Medium gaussian SVM	94	77	73,4	79	76,5
K-NN K=1	97,9	63	81	69,5	71,2
K-NN K=3	95	57,5	82	65	68,2
Cosine K-NN K=3	96,5	63,8	78	70,5	70,8
Cubic K-NN K=3	94,7	58,5	80,3	66,3	68,4
Weighted K-NN K=3	95	57,5	82	65,2	68,2
Average	95,4	61,5	77,5	70,5	

The results don't seem favorable in this experiment.The decrease in accuracy is more than anticipated.

Up till now I've discussed the problems with the results of the decision tree but after seeing the scores of that model, it's clear that it has it flaws and therefore it'll be better to discuss this matter with a better algorithm. In this case, I'll be using the bagged decision tree. As stated in the last experiment, bagged decision trees are less prone to the smearing out of Class 4 and 8. This algorithm can be seen as an upgraded version of the normal decision tree. By looking at the confusion matrix of the "upgraded" model, we'll be able to see where the classification goes wrong. The confusion matrix presented below represents the combined matrix 's of session 2, 3 and 4.

Table 15 Person 1 combined confusion matrix bagged tree model

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8
Class 1	128	0	11	0	46	0	0	1
Class 2	0	141	15	1	0	26	3	0
Class 3	0	0	111	0	0	0	57	0
Class 4	0	0	7	152	0	0	0	27
Class 5	5	0	2	0	165	0	9	0
Class 6	0	25	6	0	0	126	23	1
Class 7	0	0	3	0	0	0	160	0
Class 8	0	0	0	36	0	1	11	133

The average 19% drop in accuracy for the bagged tree model is due to the classification of the hand status. Like the other experiment, the hand labeling is a problem. The same solution that was stated in the previous experiment can be applied here.

b) Person 2

The same experiment will be repeated but with a different person. This person is brawnier than the previous person and has a lower fat percentage. The person's arm is thus thicker and has more muscle.

First off are the accuracy results of the algorithms. These accuracies are the results of testing session 1 on the models of session 1 and testing session 2 on the models of session 1.

Table 16 Person 2 session specific accuracies

Models session 1	Acc session 1 (%)	Acc session 2 (%)
Decision Tree	78,7	50,8
Bagged Tree (30)	83,6	54,8
Linear SVM	79,9	45,4
Quadratic SVM	82,5	41,5
Cubic SVM	80,6	47,5
Medium gaussian SVM	78,4	40,2
K-NN K=1	80,6	55,4
K-NN K=3	78	54
Cosine K-NN K=3	78	54,3
Cubic K-NN K=3	77,6	53,3
Weighted K-NN K=3	79,9	54,2
Average	79,8	50,1

On average, the results are lower than with person 1. By looking at the confusion matrix from the bagged tree model, the conclusion is that the problem lies with classifying the hand status.

Table 17 Person 2 confusion matrix bagged tree model

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8
Class 1	33	0	0	0	2	0	0	0
Class 2	0	27	1	0	0	7	0	0
Class 3	0	0	31	0	0	0	1	0
Class 4	0	1	0	26	0	0	0	8
Class 5	5	0	0	0	28	0	0	0
Class 6	0	8	0	0	0	23	2	0
Class 7	0	0	1	0	0	0	31	0
Class 8	0	0	0	8	0	0	0	25

Wrong classification of the hand status is dependent on the mean features and energy features of the sensor placed at the forearm. This has been proven already by looking at the different decision tree models in subsection 7.5.2. When looking at the RAW-data from both persons for every exercise, a big difference can be noticed.

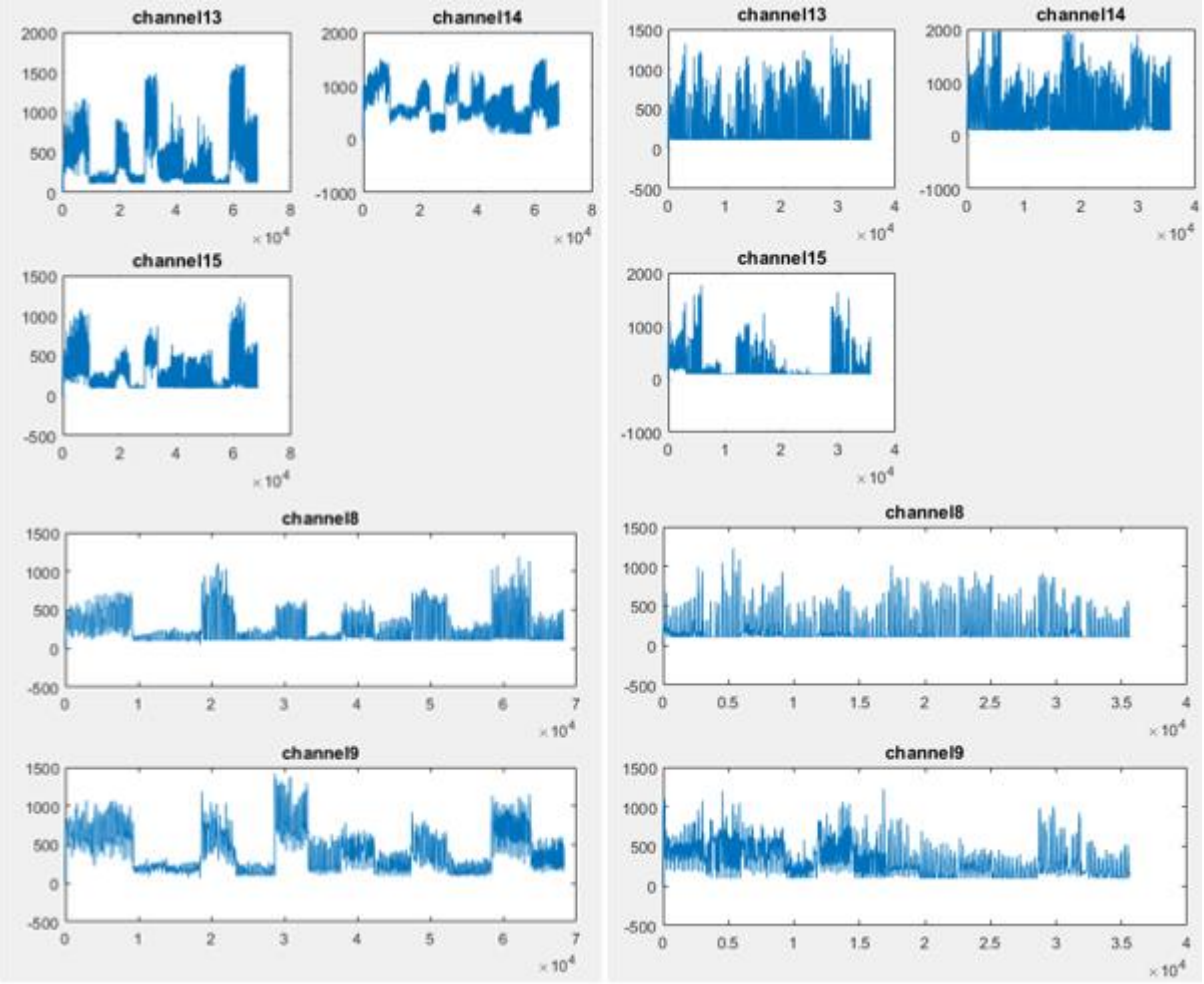


Figure 37 RAW-data comparison person 1 and 2 from the forearm sensors

The sensor values fluctuate more in person 1's dataset than in person 2's. Fluctuation is a good sign because that is what makes it possible to detect the hand status. This fluctuation can be caused by multiple reasons. The top reasons are:

- The sensor belts were fitted to person 1 so a thicker arm adjusts sensor placement;
- The tightness of the sensor belts;
- Fat percentage;
- Muscle.

Keep in mind that these graphs show you data from 12 exercises at once. In person 1's data you can clearly see when a new exercise begins while with person 2 you cannot. Another thing to remember is that in both cases the exercises are arranged in the same order, there is also a pattern in the hand status of the exercises which is as follows: closed, open, closed, open, closed, ….

To find out if it is the tightness of the sensor belts I conducted a second session with this person but asked him to lessen the tightness on the sensors in the next set. Note that it can't be too loose or else the sensors won't be kept in place. Essentially the idea is good but the second set had to be done directly after the first set and semi keeping the sensor belts on. The second set that was conducted happened after the sensor belts were taken off. The scores of the second experiment are stated below but they are not accurate to define if tightness is a leading factor in lowering the accuracy.

Table 18 Person 2 session 2 all model accuracies

Models session 2	Acc session 2 (%)
Decision Tree	74,3
Bagged Tree (30)	82,3
Linear SVM	73
Quadratic SVM	81,5
Cubic SVM	82
Medium gaussian SVM	77,5
K-NN K=1	82,3
K-NN K=3	81,2
Cosine K-NN K=3	81,5
Cubic K-NN K=3	81,1
Weighted K-NN K=3	81,5
Average	79,8

Table 19 Person 2 confusion matrix bagged tree

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8
Class 1	25	0	0	0	7	0	0	0
Class 2	0	21	0	0	0	11	0	0
Class 3	0	0	24	0	0	0	6	0
Class 4	0	0	0	29	0	0	0	3
Class 5	3	0	0	0	28	0	0	0
Class 6	0	8	0	0	0	23	0	0
Class 7	0	0	2	0	0	0	28	0
Class 8	0	0	0	4	0	0	0	27

The accuracy has stayed the same but there is no way to know for sure if this is due to tightness of the sensor belts. By taking the sensor belts off and replacing them, the positioning of the sensors relative to the muscles are changed and can result in different values.

A small conclusion: it's clear that the models need to be retrained after each session with these configurations if we want decent results. The main problem is the classification of the hand status which depends on the absolute mean sensor values, the classification of the arm direction is excellent.

At the end of the thesis the same experiment was conducted on person 1 but with relative mean values. The results are listed in the table below. By converting the mean values to relative, the classification of the hand status went better than before. This can be seen in the increase of accuracy in every model. This change made it possible to get a more reliable result on the hand status.

Table 20 Person 1 session specific accuracies with new feature

Models session 1	Acc sess 1	Acc sess 2	Acc sess 3	Acc sess 4	Averag sess 2-4
Decision Tree	91,3	74,4	65,5	71,4	70,4
Bagged Tree (30)	96,7	79,6	81,7	82,6	81,3
Linear SVM	93,7	73,5	82,9	80,3	78,9
Quadratic SVM	95,8	77,4	82,4	79,2	79,7
Cubic SVM	95,8	77,4	82,7	81,6	80,6
Gaussian SVM	94,3	86,2	73,4	81,3	80,3
K-NN K=1	98,2	74,6	85,3	80,3	80,1
K-NN K=3	97	70,8	86,3	80,3	79,1
Cosine K-NN K=3	97,3	77	83,9	78,8	79,9
Cubic K-NN K=3	97,3	71,3	83,7	79,2	78,1
Weighted K-NN K=3	97	70,8	86,3	80,3	79,1
Average	95,9	75,7	81,3	79,6	

Table 21 Person 1 confusion matrix bagged tree with new feature

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8
Class 1	165	0	10	0	10	0	0	1
Class 2	0	151	17	1	0	17	0	0
Class 3	0	0	117	0	0	0	51	0
Class 4	0	0	4	155	0	0	0	27
Class 5	16	0	9	0	151	0	4	1
Class 6	0	22	12	0	0	127	20	0
Class 7	0	0	13	0	0	0	150	0
Class 8	0	0	2	23	0	0	7	149

If we compare these results with the previous ones, it's clear that the accuracy between sessions has been improved between 4% to 14%. Converting absolute mean values to relative mean values certainly is an improvement but at the end it's still not enough. This is a healthcare application and around 80% is too low.

7.5.5 Which exercises are most useful for gathering data?

Up till now the person always did the 12 exercises but this can take a long time. To reduce the time spent training, I'd like to know if there are any redundant exercises. This experiment will train models on some exercises and will then be tested on the exercises that were left out.

The first test will be if half of the exercises will be enough to predict the other half. This'll be done twice where the half's will be switched. If the results aren't favorable then there is no point in reducing the exercises even further. The accuracies for every model and the confusion matrixes of the bagged tree model are shown. These are the results of the direction of the arm only.

Table 22 Person 1 accuracies useful exercises experiment

Models	Testset second half (%)	Testset first half (%)
Decision Tree	96,7	91,4
Bagged Tree (30)	100	87,3
Linear SVM	91	80
Quadratic SVM	95	85,8
Cubic SVM	99,7	89
Medium gaussian SVM	75,5	81
K-NN K=1	91,4	77,2
K-NN K=3	97	68
Cosine K-NN K=3	92,7	62,7
Cubic K-NN K=3	88,4	61,8
Weighted K-NN K=3	97	69,5
average	93,1	77,6

Table 23 Confusion matrix direction labels useful exercises experiment first half

	Class 1	Class 2	Class 3	Class 4
Class 1	77	0	0	0
Class 2	0	77	0	0
Class 3	0	0	71	0
Class 4	0	0	0	77

Table 24 Confusion matrix direction labels useful exercises experiment second half

	Class 1	Class 2	Class 3	Class 4
Class 1	86	0	0	0
Class 2	0	86	0	0
Class 3	0	0	80	0
Class 4	28	0	15	43

The results indicate that half of the exercises is already enough for predicting the direction of the arm. Although the second half of the exercises had trouble with classifying the arm standing still while the elbow was bend. The reason why was explained in chapter 7.5.3.

After seeing these results then maybe it's even possible to reduce the number of exercises even further **if** you only want to classify the direction of the arm. Let's first look at the results of the eight label models.

Table 25 Accuracies all models useful exercises direction and hand labels

Models	Testset second half (%)	Testset first half (%)
Decision Tree	72,2	65,4
Bagged Tree (30)	77,8	78,1
Linear SVM	70,5	62,7
Quadratic SVM	70,5	67,8
Cubic SVM	64,9	69,2
Medium gaussian SVM	46,7	61,8
K-NN K=1	63,2	64,2
K-NN K=3	64,9	54,7
Cosine K-NN K=3	60,9	53,2
Cubic K-NN K=3	56,6	48,8
Weighted K-NN K=3	64,9	58
average	64,8	62,2

Table 26 Confusion matrix direction and hand labels useful exercises experiment first half

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8
Class 1	38	0	0	0	0	0	0	0
Class 2	0	36	0	0	0	2	0	0
Class 3	0	0	12	0	0	0	23	0
Class 4	0	0	0	38	0	0	0	0
Class 5	16	0	0	0	23	0	0	0
Class 6	0	6	0	0	0	33	0	0
Class 7	0	0	0	0	0	0	36	0
Class 8	0	0	0	20	0	0	0	19

Table 27 Confusion matrix direction and hand labels useful exercises experiment second half

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8
Class 1	43	0	0	0	1	0	0	0
Class 2	0	43	1	0	0	0	0	0
Class 3	0	0	34	0	0	0	7	0
Class 4	0	6	0	26	0	0	0	12
Class 5	0	0	0	0	42	0	0	0
Class 6	0	19	1	0	0	22	0	0
Class 7	0	0	26	0	0	0	13	0
Class 8	0	0	0	0	0	1	0	41

The accuracy drops tremendously for the eight label models when not all exercises are done. The status of the hand is the problem here. There is no point in reducing the number of exercises even further if these results aren't good.

Moving on to the last test is trying one of the exercises. I've done this with two exercises separately to see if one predicts less accurate than the other. If this is true then it is in our best interest to stick with half of the listed exercises.

Table 28 Accuracies direction label trained on one exercise

Models	Trained on exercise 1(%)	Trained on exercise 3 (%)
Decision Tree	89,4	73
Bagged Tree (30)	100	84
Linear SVM	89,4	84
Quadratic SVM	93,4	86,4
Cubic SVM	91,9	90,1
Medium gaussian SVM	63	65,4
K-NN K=1	89,8	79,1
K-NN K=3	91,9	81,9
Cosine K-NN K=3	92,3	78,8
Cubic K-NN K=3	89,2	69,6
Weighted K-NN K=3	91,9	81,3
average	89,3	79,4

The accuracy differs 10% on average and that is not a good sign. The percentage in general dropped to the 90% or even 80% and that is a bit low for this application. It also depends on which algorithm you choose but that'll be decided by all the experiments and not this one alone.

A small conclusion to this is that all exercises are needed for the hand status and three are enough for the direction of the arm. The recommended three exercises are the ones where the wrists are in different positions: neutral, supination and pronation.

Reducing the number of exercises is not the only way to shorten the learning period. Till now every exercise was repeated ten times to make sure that there were enough data points. The next experiment will try to reduce the repetitiveness.

7.5.6 How many times does an exercise need to be performed?

Reducing the number of repetitions can greatly reduce the time needed to train the algorithm. This experiment will divide one session dataset so that the training data will have less repetitions. This experiment was only performed on person 1 because the sensor belts were fitted to him and had the most reliable data.

The next figure shows you the results of the experiment that was conducted. First, the number of exercises was reduced to 7-8 times then, 5-6, 3-4 and last 1-2 times. The reason being that there is not a single number is that an exercise was done 11 or 9 times instead of 10. Losing count of how many times you've repeated an exercise was common with every test person.

Table 29 Accuracies repetition experiment

Models	1-2reps	3-4reps	5-6reps	7-8reps	Average
Decision Tree	80,9	89,4	91,3	91,7	88,3
Bagged Tree (30)	81,5	94,6	96,4	92,3	91,2
Linear SVM	66,8	91,3	93,7	90,5	85,6
Quadratic SVM	61,7	92,5	95,8	95,2	86,3
Cubic SVM	68,7	91	95,8	93,5	87,3
Gaussian SVM	43,9	86,5	94,3	88,1	78,2
K-NN K=1	64,9	91	98,2	98,8	88,2
K-NN K=3	64,9	87,7	97	97	86,7
Cosine K-NN K=3	66,8	89,8	97,3	97	87,7
Cubic K-NN K=3	56,6	85,3	97,3	95,9	83,8
Weighted K-NN K=3	65,6	88,3	97	97	87,0
Average Accuracy	65,7	89,8	95,8	94,3	

The decline starts to happen at repeating the exercise for only 3-4 times for most algorithms. The confusion matrix of the gaussian SVM model is shown below to show were the mistakes are starting to happen.

Table 30 Confusion matrix bagged tree repetition experiment

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8
Class 1	43	0	0	0	0	0	0	0
Class 2	0	43	0	0	0	0	0	0
Class 3	0	0	37	0	0	0	2	0
Class 4	2	0	0	40	0	0	0	1
Class 5	11	0	0	0	31	0	0	0
Class 6	0	10	0	0	0	32	0	0
Class 7	0	0	10	0	0	0	29	0
Class 8	0	0	0	9	0	0	0	33

As you can see the mistakes start with the prediction of the hand status. There are still algorithms that stay above the 90% range with only 3-4 repeats, so deciding how many repeats are needed is also dependent on the model.

To make sure that these results are reliable, the experiment is done a second time on the same person but on a different dataset. The results are shown in the following table.

Table 31 Accuracies all models repetition experiment

Models	1-2 rep (%)	3-4 reps (%)	4-5 reps (%)	7-8 reps (%)	Average
Decision Tree	87,4	97,2	94,6	95,8	93,8
Bagged Tree (30)	93	96,2	98,7	96,9	96,2
Linear SVM	80	86,4	93,4	91,7	87,9
Quadratic SVM	85,1	91,1	95,5	94,8	91,6
Cubic SVM	85,8	93,9	96,7	92,7	92,3
Gaussian SVM	87	93,4	93,4	96,9	92,7
K-NN K=1	95,3	96,2	98,3	96,9	96,7
K-NN K=3	76,8	94,8	97,1	95,8	91,1
Cosine K-NN K=3	78,3	96,2	97,5	95,8	92,0
Cubic K-NN K=3	77,8	94,8	96,7	96,9	91,6
Weighted K-NN K=3	93,2	94,8	97,5	95,8	95,3
Average Accuracy	85,4	94,1	96,3	95,5	

The results indicate that the accuracy doesn't drop when performed on another measurement set. Just as with the first measurements, the accuracy drops when the number of reps go below three.

A small conclusion for this experiment is that the algorithms can create a reliable model after doing every exercise at least three times.

7.6 Selecting the best algorithm

Now that all the experiments are done and the results are known, it's time to choose an algorithm that's most suited to our needs. Because this is a healthcare application, reliability is of the utmost importance. This means that the algorithm needs to perform well even in sub optimal conditions like:

- Not enough exercises done;
- Not enough repetitions for one exercise;
- Positioning of the sensors not optimal;
- Etc.

The experiments that were done on this chapter give some light on that. Looking back at the results accuracy was always a decent indicator for that but it's also important to know where it goes wrong. Two algorithms with the same accuracy can have different outcomes. The most important thing for these algorithms is that the direction of the arm is perfectly predicted. The status of the hand is also important but knowing when to drive the actuators is more important. Hand status can influence the intensity of driving the actuators but not when.

I've not specified every confusion matrix of every algorithm because the length of it will be immense but the decision of the best suited algorithm in this project was based on:

- None or minimal "smearing" in vertical and horizontal direction aka direction prediction (mentioned in chapter 7.5.3.);
- Overall accuracy in the experiments.

I've concluded that Bagged trees are the best option for this application. This algorithm was the most reliable even in sub optimal conditions. Bagged trees can be explained as multiple decision trees but with a slightly different view on the problem. Therefore, it's also less prone to overfitting data. It also did excellent on the prediction of the arm direction as seen in the confusion matrixes in the experiments.

8 Before and after

At the start of the thesis only two sensors located at the wrist were used to detect human arm movement. Values were directly read and send to the actuator to drive the exo-skeleton arm. These two sensors detected when the wrist touched one side and would act accordingly. The problem was that this method caused stuttering. The wrist would touch the sensor and the arm moves so that the sensor doesn't touch the wrist. This happens during the whole movement and needed a solution. Adjusting the parameters for the arm was also a pain to do.

Now at the end, eleven sensors are used to do this. Six located at upper arm and five at the forearm. A HID board was already developed but was tested and implemented in this thesis. The existence of the HID board allows up to sixteen sensors to be used to detect human arm movement. Eleven of those sixteen are in use right now six are used for the upper arm (biceps and triceps) and the other five are present on the dorsal and ventral side of the forearm.

Communication between the HID board and a workstation has been established by implementing a CAN-bus listener on the workstation. Sensor values are transported to the CAN-Bus on the UCM and the workstation listens to a specific channel to gather the necessary data. At the moment, the data is stored in csv files for testing purposes.

A learning period has been created with three to twelve exercises dependent on the number of labels that need to be predicted. Initial labelling is also implemented when the exo-skeleton user is doing his exercises so it's known when he was performing certain exercises.

Scripts have been developed to extract the necessary features and to add additional labels necessary for the machine learning algorithms. Another script has been developed to test algorithms and track certain characteristics like accuracy and the confusion matrix.

9 Future work

9.1 Data acquisition board

The initial values of the sensors and the sensitivity are controlled by hardware and limits the possibility of tuning on the fly. The hardware has been adjusted to a certain movement executed by person 1 so that the sensor values have a maximum range. While performing different exercises some data is lost because values go over the limit or are optimized. Transforming the hardware into software will allow us to do this. Adjusting certain parameters so that an optimized range is achieved is possible with software instead of hardware.

9.2 Convert MATLAB to Java

Extracting features, labelling and creating machine learning models are realized in MATLAB on a workstation. Using MATLAB on the UCM is expensive and takes a lot of memory which is not suited in the final product. Ultimately these MATLAB scripts need to be converted to JAVA code so that the UCM can do this instead of a separate workstation.

9.3 Real-time predicting

Predicting can only be done when data is available in csv format. Models can be created but not implemented in a real-time application because there is no script for it yet. To implement the models in real-time, it first needs to be converted to JAVA code where the UCM can work with.

9.4 Improving algorithms for hand status

When the user takes the suit off and puts it back on, new models have to be created to guarantee high accuracy and comfort to the user when trying to predict the hand status. Hand status is predicted by the features that are extracted from the sensors at the forearm. At the moment, the data that comes from these sensors is not optimal due to several factors, some of which we have no control over. The following factors can influence the data values of these sensors:

- Positioning of the sensors relative to the muscles;
- Tightness of the sensor belts;
- Muscle density;
- How well the sensor belt fitted to the person.

Fitting the sensor belts is a matter of manufacturing. The diameter of the user's arm must be known and the belts should be created to them accordingly.

Tightness of the sensor belts influences the starting value of the sensor and the sensitivity. It's good that the sensor is tightly attached to the user but too much will cause the initial value to be too high

and thus decreasing our range and sensitivity. A difference in tightness between sessions can also influence certain features so that the accuracy from the models drop. Controlling the tightness of the sensor belts will help increase the accuracy between sessions.

The positioning of the sensors relative to the muscles is no problem for a nurse, doctor, etc… but to a technician or a normal user this is a problem. The values coming from the sensors will be different when they are in a different place than before on the human arm. A possible solution for this is to also attach the sensor belts to the exo-skeleton itself but then there is still the problem of the user placing it's arm differently to fit into the suit. A clear solution hasn't been thought off yet.

There is no way of controlling the muscle density because this is person specific. More muscle means that the arm expands more when doing exercises and this means increased activity on the sensors. Which translates in adjusting the hardware on the HID-board. Another effect of having more muscle is that the tightness of sensor belts can be reduced a little. This phenomenon came fort when gathering data with a brawnier person.

10 Conclusion

It's possible to detect movement from the human arm and hand in the sagittal plane with this technology. Classifying the direction of the arm and hand status is certainly possible but still needs some fine tuning. Fine tuning can contain feature reduction and/or feature optimization.

The learning period that is developed right now has enough exercises which cover basic human arm movement. It also has been 'trimmed' to reduce the time needed to fully train the algorithm.

10.1 Experimental growth

After conducting the first measurements without initial labeling, it became clear to me that I had no definitive way to exactly pinpoint the location where a movement started or ended. This problem was solved by implementing initial labeling via the keyboard of the workstation. I explained to the test person that it's important to only move on my command and pressed the right key's for labelling the movement stages.

Another issue that was found experimentally is that the learning period must be done in one go. Taking off the sensor belts and reapplying them can influence the sensor values and thus the features, as seen in one of the experiments. Factors that change with reapplying the sensor belts are:

- Positioning of the sensor relative to the muscles;
- Tightness of the sensor belt.

While doing experiments on other persons it also became clear that the overall accuracy is lower with the same sensor belts. This has to do with the change in arm thickness and also muscle density.

Literature list

[1] W. COMmeto, AAU, BIOT, COM, HJALP, MTD, UGAV, UoL, “Axo-suit: assistive exoskeleton suitable for elderly persons,” 2015.

[2] “How to Draw a Laptop, Step by Step, Stuff, Pop Culture, FREE Online Drawing Tutorial, Added by Dawn, March 11, 2011, 6:21:49 am.” [Online]. Available: http://www.dragoart.com/tuts/7657/1/1/how-to-draw-a-laptop.htm. [Accessed: 17-May-2017].

[3] “BOSCH CAN Specification,” 1991.

[4] Boston Dynamics, “BigDog.” [Online]. Available: https://www.bostondynamics.com/bigdog. [Accessed: 06-Jun-2017].

[5] K. Jahanbakhsh, “What machine learning algorithms are used for internet advertising? - Quora,” *January 22*, 2017. [Online]. Available: https://www.quora.com/What-machine-learning-algorithms-are-used-for-internet-advertising. [Accessed: 06-Jun-2017].

[6] A. Geitgey, “Machine Learning is Fun Part 6: How to do Speech Recognition with Deep Learning,” *December 24*, 2016. [Online]. Available: https://medium.com/@ageitgey/machine-learning-is-fun-part-6-how-to-do-speech-recognition-with-deep-learning-28293c162f7a. [Accessed: 06-Jun-2017].

[7] A. Munoz, “Machine Learning and Optimization.”

[8] M. Mehryar, R. Afshin, and T. Ameet, *Foundations of machine learning*. 2012.

[9] Mitchell Tom, “The Need for Biases in Learning Generalizations,” 1980.

[10] M. van Otterlo and M. Wiering, “Reinforcement Learning and Markov Decision Processes,” Springer Berlin Heidelberg, 2012, pp. 3–42.

[11] DnI institute, “Building Predictive Model using SVM and R – DnI Institute,” *September 13*, 2015. [Online]. Available: http://dni-institute.in/blogs/building-predictive-model-using-svm-and-r/. [Accessed: 05-Jun-2017].

[12] T. M. Mitchell, “Decision tree learning,” *Decis. tree Learn.*, p. 74, 1997.

[13] J. Zhu, “Machine Learning: Decision Trees.”

[14] J. Brownlee, “Bagging and Random Forest Ensemble Algorithms for Machine Learning - Machine Learning Mastery,” *April 22*, 2016. [Online]. Available: http://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/. [Accessed: 05-Jun-2017].

[15] S. Schaal and C. C. Atkeson, “From Isolation to Cooperation: An Alternative View of a System of Experts.”

[16] Wikipedia, “Naïve bayes classifier,” *March 30*, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Naive_Bayes_classifier.

[17] J. Brownlee, “Naive Bayes for Machine Learning - Machine Learning Mastery,” *April 11*, 2016. [Online]. Available: http://machinelearningmastery.com/naive-bayes-for-machine-learning/. [Accessed: 05-Jun-2017].

[18] R. A. FISHER, “THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS,” *Ann. Eugen.*, vol. 7, no. 2, pp. 179–188, Sep. 1936.

[19] T. S. Körting, “(5) How kNN algorithm works - YouTube,” *february 18*, 2014. [Online]. Available: https://www.youtube.com/watch?v=UqYde-LULfs. [Accessed: 05-Jun-2017].

[20] J. Brownlee, “K-Nearest Neighbors for Machine Learning - Machine Learning Mastery,” *April 15*, 2016. [Online]. Available: http://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/. [Accessed: 05-Jun-2017].

[21] Wikipedia, “Linear regression,” 2017. [Online]. Available: https://en.wikipedia.org/wiki/Linear_regression.

[22] nasacj, “gradient descent.” [Online]. Available: http://nasacj.net/?author=1. [Accessed: 05-Jun-2017].

[23] S. Reid and R. Tibshirani, “Sparse regression and marginal testing using cluster prototypes,” *Biostatistics*, p. kxv049, Nov. 2015.

[24] A. N. Tikhonov, “Об устойчивости обратных задач,” *Dokl. Akad. Nauk SSSR*, vol. 39, no. 5, pp. 195–198, 1943.

[25] J. Brownlee, “Linear Regression for Machine Learning - Machine Learning Mastery,” *March 25*, 2016. [Online]. Available: http://machinelearningmastery.com/linear-regression-for-machine-learning/. [Accessed: 06-Jun-2017].

[26] LISA lab, “Logistic function,” *April 21*, 2017. [Online]. Available: http://deeplearning.net/software/theano/tutorial/examples.html. [Accessed: 05-Jun-2017].

[27] J. Brownlee, “Logistic Regression for Machine Learning - Machine Learning Mastery,” *April 1*, 2016. [Online]. Available: http://machinelearningmastery.com/logistic-regression-for-machine-learning/. [Accessed: 05-Jun-2017].

[28] Brownlee Jason, “4-steps to get started in machine learning: the Top-Down strategy fir beginners to start and practice,” *March 3*, 2014. [Online]. Available: http://machinelearningmastery.com/4-steps-to-get-started-in-machine-learning/. [Accessed: 17-Nov-2016].

[29] Brownlee Jason, “How to define your machine learning problem,” *December 23*, 2013. [Online]. Available: http://machinelearningmastery.com/how-to-define-your-machine-learning-problem/. [Accessed: 17-Nov-2016].

[30] Brownlee Jason, “Quick and Dirty Data Analysis for your Machine Learning Problem,” *February 24*, 2014. [Online]. Available: http://machinelearningmastery.com/quick-and-dirty-data-analysis-for-your-machine-learning-problem/. [Accessed: 17-Nov-2016].

[31] Brownlee Jason, “How to Prepare Data For Machine Learning,” *December 25*, 2013. [Online]. Available: http://machinelearningmastery.com/how-to-prepare-data-for-machine-learning/. [Accessed: 17-Nov-2016].

[32] Brownlee Jason, “How to Improve Machine Learning Results,” *December 30*, 2013. [Online]. Available: http://machinelearningmastery.com/how-to-improve-machine-learning-results. [Accessed: 17-Nov-2016].

[33] J. Brownlee, “How to Evaluate Machine Learning Algorithms - Machine Learning Mastery,” *December 27*, 2013. [Online]. Available: http://machinelearningmastery.com/how-to-evaluate-machine-learning-algorithms/. [Accessed: 20-Nov-2016].

[34] J. Brownlee, “How to Improve Machine Learning Results - Machine Learning Mastery,” *December 30*, 2013. [Online]. Available: http://machinelearningmastery.com/how-to-improve-machine-learning-results/. [Accessed: 20-Nov-2016].

[35] P. Chang, “python - how to implement walk forward testing in sklearn? - Stack Overflow,” *August 12*, 2015. [Online]. Available: https://stackoverflow.com/questions/31947183/how-to-implement-walk-forward-testing-in-sklearn. [Accessed: 05-Jun-2017].

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
learning algorithms for sensor interpretation on an exo-skeleton

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,

Bonné, Ruben

Datum: **6/06/2017**