Czech Technical University in Prague
Faculty of Electrical Engineering

Department: Department of Cybernetics
Study program: Open Informatics
Specialisation: Artificial Intelligence and Computer Science

Artificial Intelligence for the Robust Analysis of Piezoelectric Biosensors

Umělá inteligence pro robustní analýzu signálu z piezoelektrických biosenzorů

BACHELOR'S THESIS

Author:	Lukáš Frána
Supervisor:	Ing. Vratislav Fabián, Ph.D.
Year:	2021

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Frána Lukáš**	Personal ID number:	**483787**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Specialisation:	**Artificial Intelligence and Computer Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Artificial Intelligence for the Robust Analysis of Piezoelectric Biosensors

Bachelor’s thesis title in Czech:

Umělá inteligence pro robustní analýzu signálu z piezoelektrických biosenzorů

Guidelines:

1) Make a literature search for artificial intelligence methods usable for signal analysis of piezoelectric sensors.
2) Design and implement selected artificial intelligence methods for robust signal analysis of piezoelectric sensors.
3) Compare the selected methods and evaluate the proposed methods in terms of practical applicability.

Bibliography / sources:

[1] Janshoff, Andreas, Hans-Joachim Galla, and Claudia Steinem. "Piezoelectric mass-sensing devices as biosensors—an alternative to optical biosensors?." Angewandte Chemie International Edition 39.22 (2000): 4004-4032.
[2] Russell, Stuart, and Peter Norvig. "Artificial intelligence: a modern approach." (2002).

Name and workplace of bachelor’s thesis supervisor:

Ing. Vratislav Fabián, Ph.D., Department of Physics, FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Nicholas Scott Lynn, Jr., Ph.D., Institute of Physics of the Czech Academy of Sciences, Prague

Date of bachelor’s thesis assignment: **10.01.2021** Deadline for bachelor thesis submission: **13.08.2021**

Assignment valid until: **30.09.2022**

Ing. Vratislav Fabián, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt Student’s signature

Author statement for undergraduate thesis

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date 12. 8. 2021

....
Lukáš Frána

Acknowledgements

First of all, I would like to express my gratitude to my thesis supervisor, Ing. Vratislav Fabián, PhD. He has been a constant source of encouragement and insight during my research.

I would like to thank to Viktor Procházka for helping me overcome my procrastination.

Special thanks go to the staff of the Institute of Physics and especially RNDr. Hana Lísalová, Ph.D., who maintained a pleasant and flexible environment for my research. I would like to express special thanks to Nicholas S. Lynn, PhD., who helped me with numerous problems and professional advancements.

Finally, my greatest thanks go to my family members, for their infinite patience and care.

Abstract/Abstrakt

Piezoelectric biosensors can be used for the detection of toxic materials in a complex mass. Here, various techniques are discussed for signal analysis of these sensors. Methods for data preparation, noise reduction and filtering will be introduced. The output of the data preprocessing will be used as an input for classification algorithms based on the artificial intelligence.

This thesis discusses the use of artificial intelligence (AI) to improve the performance characteristics of QCM biosensors. More specifically, it will be shown that AI can be used to classify positive and negative samples based on the changes in resonant frequency.

Keywords: artificial intelligence, machine learning, quartz crystal microbalance, support vector machine, random forest, k-nearest neighbours

Piezoelektrické biosenzory lze využít k detekci toxických materiálů ve komplexních vzorcích. Diskutovány budou různé techniky pro analýzu signálu z těchto senzorů. Budou představeny metody pro přípravu dat, redukci šumu a filtrování. Výstup z předzpracování dat bude použit jako vstup pro klasifikační algoritmy založené na umělé inteligenci.

Tato práce pojednává o využití umělé inteligence (UI) ke zlepšení výkonnostních charakteristik QCM biosenzorů. Konkrétněji se ukáže, že UI lze použít ke klasifikaci pozitivních a negativních vzorků na základě změn rezonanční frekvence.

Klíčová slova: umělá inteligence, strojové učení, mikrováhy z křemenných krystalů, metoda podpůrných vektorů, náhodný les, k-nejbližších sousedů

Contents

List of Tables

List of Figures

Introduction

The diagnosis and management of public health is aided by the detection of biomarkers in a variety of bodily fluids. Given the development of the recent Covid-19 pandemic, measuring the presence of biological agents is now perhaps more important than ever. A number of newly developed biosensing platforms represent a rapid and inexpensive alternative to currently accepted diagnostic methods; a gold standard example of these is the widely used polymerase chain reaction (PCR) test to detect nucleic acid remnants present in persons positive with Covid-19 [1]. However, the PCR methods are expensive and time consuming. The result is known after approximately 4 hours, and they require a laboratory with trained personnel.

One such novel and promising alternative method for Covid-19 diagnosis is the quartz crystal microbalance (QCM) biosensor [2], which is a piezo-like crystal that oscillates at a resonant frequency (in the MHz range). The transduction component of the biosensor acts to create an electric signal that can be used to detect subtle changes in the state of the mass present on the surface of the crystal: indicated by a change in the resonant frequency. When made functional with a biocomponent (e.g., antibodies or DNA), these QCM sensors allow for the quantification of the presence of a wide variety of biomarkers and, because of their high sensitivity, have gained significant attention in the past decade [3].

The path from electrical signal generated on the device to the classification of positive and negative samples begins with data collection. The QCM electronics sends the measured values through the serial port to the computer, which saves them in real-time. The raw (unprocessed) QCM biosensor output consists of two quantities, amplitude and phase, which are both provided as a function of frequency. Due to a number of problematic factors (e.g., poor electrical connections), the data output from this biosensor tends to be noisy. Filtering and classifying these data via standard algorithms is difficult, as there are a significant number of complicated artifact types. Removing these problematic data points helps to reduce noise, which makes later analysis easier and increases overall sensing performance.

Figure 0.1: openQCM device

Artificial intelligence methods [4, 5] creates an abstraction upon complex and slow algorithms and maintains high potential to increase the performance of QCM biosensors. Supervised machine learning algorithms [6] can learn classification parameters from manually labeled data, and then use these parameters to make predictions on previously unseen data. These algorithms are well researched and have been shown to have good results on data from a broad selection of domains.

However, using time series of different lengths is highly impractical for this sort of application. Because of this, the first half of this thesis is focused on data preparation. Because the QCM is capable of creating a few million values in every measurement, it is first necessary to decide which data improves the performance of pathogen detection. Choosing the wrong properties of the collected data points can slow down the speed of execution and decrease the precision of predictions.

The final part of this thesis is on the **machine learning (ML)** [7] itself, which operates on top of a dataset prepared using the methods discussed in the 1st half. Three important classifiers will be introduced - support vector machine, random forest and k-nearest neighbours - along with auxiliary methods for splitting the entire dataset on (a) a first set determined for building the classifier and, (b) a second set used for evaluating its performance or a method for comparing models between themselves. The results are discussed at the end as well as the influence of filters.

The methods developed in this thesis were then applied to real data concerning samples collected from a variety of surfaces within public transportation vehicles, which were tested for the presence of biological material related to Covid-19 [8]. Some of the measurements were removed due to non-standard measuring process and were not suitable for analysis. The samples were collected in a safe manner in compliance with approved hygienic methods. These samples were analyzed at the Institute of Physics of the Czech Academy of Sciences in a specialized laboratory. These methods are applied towards an experimental openQCM device (figure 0.1), an open hardware/software platform developed in Italy and based on the Teensyduino chip. Specifically, they are applied using the Python language [9] and Jupyter notebooks [10], along mainly with the popular packages NumPy [11], SciPy [12], SciKit learn [13]. Python is an interpreted language and it is used worldwide for data analysis, and Jupyter notebook is an interactive way to write Python code with graphical interface.

Chapter **1**

Making a signal

1.1 Raw data format

The openQCM device communicates with the computer through its serial port. The data collection for one point in time starts when computer sends three values: **starting frequency** f_{start}, **stop frequency** f_{stop}, and **frequency step** f_{step}. Hence, the device will start measuring both phase and amplitude at f_{start}, after which a frequency counter will increase by the value of f_{step} until the counter is equal or greater to f_{stop}. The number of steps is $N = (f_{stop} - f_{start})/f_{step} + 1$. At the end, the temperature measured on the chip is sent. The frequencies f_{start} and f_{stop} are chosen in the calibration process. This whole cycle is repeated every 600 to 800 ms. It cannot be an exact number due to the nature of electronics inside the QCM device and its firmware/software. These values are saved in test files, then converted to NumPy format and finally saved in binary format for more convenient usage. The number of points N is in these measurements always 501, which is the defualt number set by the manufacturer of openQCM device.

The **resonant frequency (RF)** can be calculated from both phase and amplitude arrays expressed as $A = \{A_1 \dots A_N\}$ and $P = \{P_1 \dots P_N\}$ respectively. The RF corresponds to a point on the x-axis (frequency) where either the phase or amplitude is the highest. Both quantities have similar properties, and same methods discussed below can be applied on them; however, the hardware approach of how is the phase measured in the device itself is not ideal to work with it as it is not temporally stable. Here I consider the RF only obtained from the amplitude vs. frequency graph.

$$rf = \operatorname*{argmax}_i Q_i,$$

where $i \in \{1 \dots N\}$.

The signal data calculated by the equation above are not optimal to work with directly (see figure 1.2). The reason for this is the discrete nature of the data, where the position corresponding to the maximum of the array Q will always be a multiplication of f_{step}. In reality, however, the actual RF can exist between the measured steps, because the physical phenomena represented by Q as a function of f is continuous and the probability that $\operatorname{argmax}_x f(x)$, where $x \in \mathbb{R}$, will be at one specific rf is close to 0. Thus it is important to introduce a method of how to determine the actual RF with sub-f_{step} resolution.

In theory, measurements of RF should not change if the mass on the crystal is not changed. However, as previously mentioned, the raw data suffers from **noise** [14], or more specifically, multiple measurements of amplitude at the same frequency will not return the same value. This is caused by characteristics of electronics, electromagnetic radiation, temperature and other external influences. To partially mitigate this issue, the device measures amplitude 32 times at every frequency, and then exports the arithmetic mean (equation 1.2) of those values.

Among other more complex ways how to measure the noise, computing the standard deviation in unchanged (constant) environment provides information about the noise in the signal. The standard deviation of an $x = (x_1, \ldots, x_n)$ is

$$std = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}, \tag{1.1}$$

where

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1.2}$$

is known as arithmetic mean.

1.1.1 Vizualization

Figure 1.1 shows example plots of normalized phase/amplitude vs. frequency. Since the RF is computed only from its peak, the signal can be normalized in two steps

$$Q_{prenorm} = (Q - min(Q))$$

and

$$Q_{norm} = \frac{100 \cdot Q_{prenorm}}{max(Q_{prenorm})}.$$

This ensures that the signal is always in the interval $[0, 100]$. The signal maximum will be near the center of a properly calibrated crystal, decreasing in a similar fashion on each side. The phase has a wider peak, decreasing slower on both sides, thus it will also have bigger noise [15]. The amplitude decreases from its maximum faster and thus will have lower noise. It can be seen that the two maximums (highlighted with cyan dots) are not at the same frequency. This difference is due to the physics of the acoustic waves generated on the crystal, a topic that is beyond the scope of the analysis presented here. Because resonant frequency is always calibrated around 10 MHz, it is normalized by subtracting aforementioned 10 MHz.

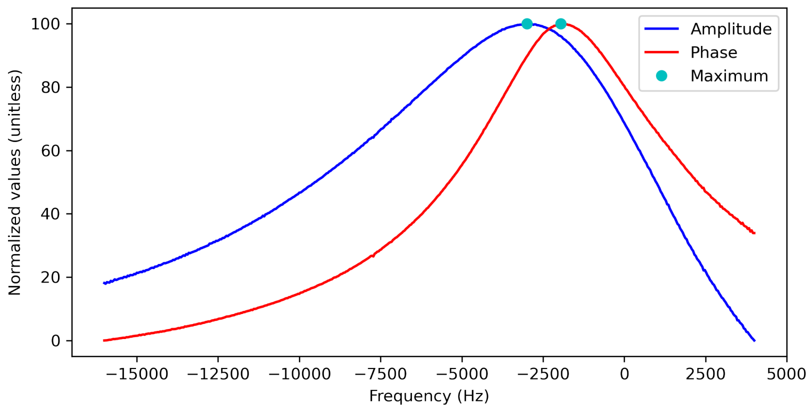


Figure 1.1: Amplitude/phase (normalized, unitless) vs frequency (Hz after subtracting 10 MHz)

For every time point, data similar to that shown in figure 1.1 is used to extract the RF. As a first approach RF was calculated as the frequency corresponding to the absolute maximum of Q. This time-series data, caculated from the amplitude, is plotted in figure 1.2. If not explicitly mentioned, only a part of the measurement will be shown.

It can clearly be seen that the data is divided by discrete 40 Hz differences. That makes perfect sense, because the maximum is taken directly from values in the f_{step} raw data array. Analysis of data calculated in this manner would be difficult, if not impossible, due to the high noise. The next section introduces a method to reduce the noise to a value below the f_{step} resolution.

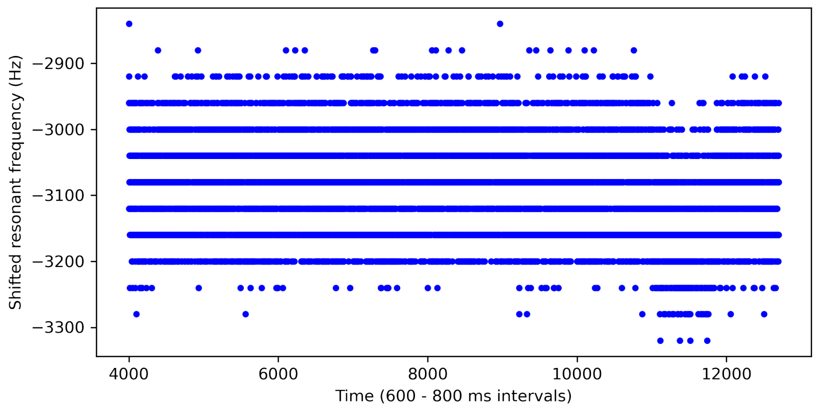


Figure 1.2: Raw resonant frequency vs time (y-axis reported as RF - 10 MHz, and time reported as the index from the beginning of measurement)

1.2 Noise reduction

There exist several methods for reducing the signal noise in sensors based on a resonant signal. One method is to fit a portion of the measure points to a suitable function using least squares polynomial regression [16]. This method is fast enough to be implemented in real-time, and furthermore, provides good results. The only requirement is to define several parameters related to the fitting process algorithm, specifically in the selection of data to be fitted as well as the polynomial used for fitting. The entire frequency interval does not have to be used - the computational power can be saved while achieving better results with fitting only a selected portion of data around the maximum. In addition to this, one must guess a function to fit the points to, although this is easier with respect to data selection. There are two options that can used to select such data to be used in curve fitting.

1. Let a number N_p be defined as the number of points from the maximum to each side of the peak. This results in total of $2 * N_p + 1$ points to be used for fitting. This method is very simple to both understand and code, and is also fast to compute. On the other hand, it has several drawbacks. The value N_p must be set by hand or computed from the data. With a fixed number of the measure points for each signal point this is not an issue, but that is not always the case. For example, we can take a narrower interval and use only 251 values (rather than the default 501), which increases the speed of gathering the measure points. In this example $N_p = 50$ would work well for 501 data points (fits only around 10% of all points), whereas it might not work as well when using 251 points (fitting around 20%). In the latter case there is twice as much data being fitted. Even choosing fixed percentage portion of the data does not necessarily solve this problem, as it does not reflect the nature of different signals.

2. Another option is to set a threshold thr, which serves to select data above a critical y-value for fitting. This can be accomplished by defining a signal boundary B_Q, expressed in percentage and computed as

   $$B_Q = (\max Q - \min Q) \cdot thr + \min Q.$$

   Every point above B_Q boundary is then selected for the curve fitting. The number of picked points varies as it cannot be predicted how many values will be high enough to overcome this boundary. This approach works better, as it allows for a dynamic selection of fitted points.

1.2.1 Least squares

The least squares method is used to approximate the solution of a system where a number of equations is higher than a number of unknowns. This approach works by minimizing the sum of the squares of the differences between an expected value (predicted by a model) and the actual value. Here, using a least squares fit with a second order polynomial, the sum of the squares can be expressed mathematically as

$$S(\beta) = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} ((\beta_2 x_i^2 + \beta_1 x_i + \beta_0) - y_i)^2,$$

where $\beta = (\beta_0, \beta_1, \beta_2)^T \in \mathbb{R}^3$ are the polynomial coefficients from the formula $f(x) = \beta_2 x^2 + \beta_1 x + \beta_0$, n is number of discrete data points, and r_i is residuum: the difference between the actual y-value and the approximated one. Finding the minimum of the sum S is an optimization problem. Clearly, S is a convex function with smooth derivatives. One way to find the minimum of S is to find a point where the gradient of S is zero, that is

$$\frac{\partial S}{\partial \beta_j} = 2 \sum_{i=1}^{n} r_i \cdot \frac{\partial r_i}{\partial \beta_j} = 2 \sum_{i=1}^{n} r_i \cdot x^j = 0, \ j \in \{0, 1, 2\}.$$

This system of equations can be solved numerically with various methods, the most common of which is the gradient descent method. The optimal solution can also be found analytically using the Vandermonde matrix, which is defined as follows

$$V = (v_{i,j}), \ v_{i,j} = x_i^{j-1}, \ i \in \{1, \dots, n\}, j \in \{1, \dots, m\},$$

where n is the number of data points and m is the degree of the polynomial to fit increased by one. The optimal coefficients β that minimize S can then be computed as

$$\beta = (V^T V)^{-1} V^T y.$$

The inverse of $V^T V$ can be calculated because the Vandermonde matrix is guaranteed to be nonsingular due to the use of distinct data points.

After choosing the right measure points to be fitted, one need to take a look at the specific functions (polynomials) for curve fitting. The parabola (a 2nd order polynomial in form of $f_{fit}(x) : ax^2 + bx + c = 0$) is an obvious choice and provides a satisfactory approximation of the peak shape. Naturally, both the measured points and the 2nd order polynomial have only one local maximum, which applies to both phase and amplitude. Once the parameters a, b, c for parabola are found, an analytical solution can be used to find its argument maximum - corresponding to the RF - by using the first derivative of f_{fit} equal to zero. This is expressed in the formula $x_{max} = -2a/b$, whereby the resonant frequency is thus rf $= x_{max}$.

The results of this fitting method are shown in figure 1.3, which shows a drastic improvement over the raw data (figure 1.2). The time-series trend of the signal (determining if there is any change of resonant frequency) is now visually apparent, which is not the case in the data shown in figure 1.2.

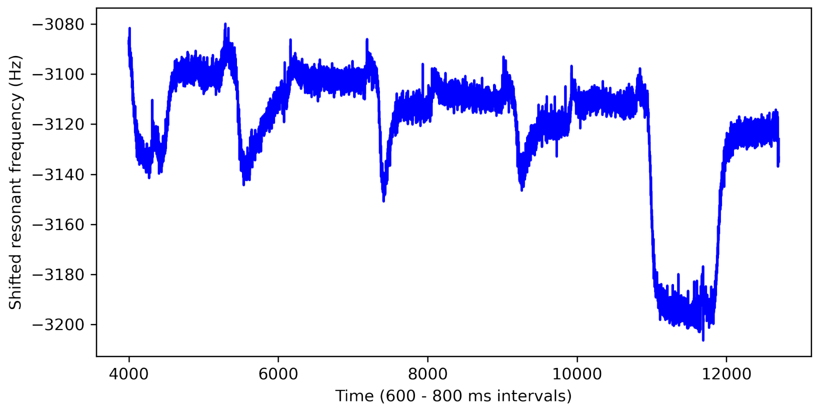


Figure 1.3: Fitted ($thr = 0.95$) resonant frequency vs time (y-axis reported as RF - 10 MHz, and time reported as the index from the beginning of measurement)

The noise in the interval $[6400, 7000]$ calculated via equation 1.1 was 2.601. The next sections details the use of several filters that help to reduce the noise, which will be always calculated from the same interval and will be always rounded to three decimal places.

1.2.2 Savitzky-Golay filter

The Savitzky-Golay filter [17] is curve smoothing method that is based on the use of least squares polynomial regression. It replaces the original y-value in the series of points with the new value obtained from the fit. The regression is performed on successive sub-sets (windows) of adjacent data points: a process known as convolution. Its goal is to increase the precision of the data via reducing the signal noise while simultaneously maintaining the signal tendency.

The filter takes two parameters, $m, p \in \mathbb{N}$. The value p is the polynomial order, and is typically low-degree. The value m is an odd number and defines the window length. Starting with an input set of $\{(x_j, y_j)\}, j = 1..m$, the algorithm takes the point at the index i and, using $\frac{m-1}{2}$ points on each side (number of points sums to m), performs least squares regression using a p-th order polynomial fit as f_j to those points, and then computes a new $Y_j = f_j(x_j)$. The set of Y_j values is the smoothed output.

Figure 1.4 shows application of SG filter with window length of 33 and polynomial order of 2. The noise for this data is 0.635 (same interval as above), which shows great improvement compared to the noise in the unfiltered data.

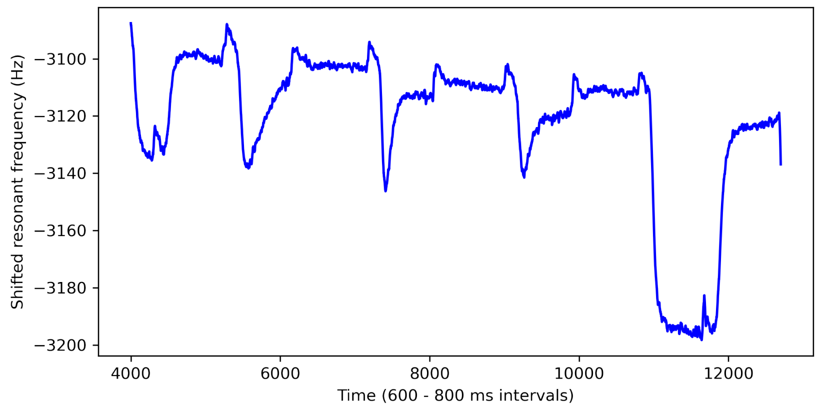


Figure 1.4: SG filtered resonant frequency vs time, applied to the same data set in fig. 1.3 and using coefficients of $m = 33$, $p = 2$ (y-axis reported as RF - 10 MHz, and time reported as the index from the beginning of measurement)

1.2.3 Median filter

One of the disadvantages of the Savitzky-Golay filter is that the underlying least squares method is highly sensitive to outliers. The points far from the others affect the polynomial fit much more, because the distance here is squared. This can result in new artificial created artefacts in the signal. A way to remove these outliers is through the use of a so-called median filter.

The median filter [18] works with a single parameter, the window size of length m. Instead of fitting the points in the window to polynomial, it sorts them and then outputs the median value of the data in the window. One of the advantages is that median filter does not create new values, but rather repeats previously seen value. This filter works well to remove outliers, which will likely be near the edges of the sorted values.

Figure 1.5 shows the applied use of the median filter to the unfiltered data. The median filter lowers the noise level to 0.607, which is slightly better than that calculated by the SG filter.

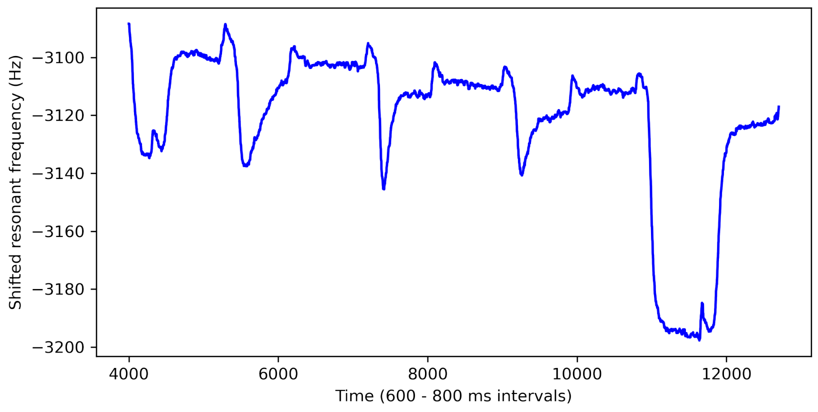


Figure 1.5: Median filtered resonant frequency vs time, applied to the same data set in fig. 1.3 and using a window size of 33 (y-axis reported as RF - 10 MHz, and time reported as the index from the beginning of measurement)

1.2.4 Combining filters

The best results can be achieved by combining both filters. In this approach a median filter is first applied to remove problematic outliers, after which a SG filter is used to smooth the curve. The result of this approach can be seen in figure 1.6. This approach leads to a noise of 0.567, which is more than 4.5× lower than the unfiltered one. If not specifically mentioned, this combined filter will be used in the plots from now on.

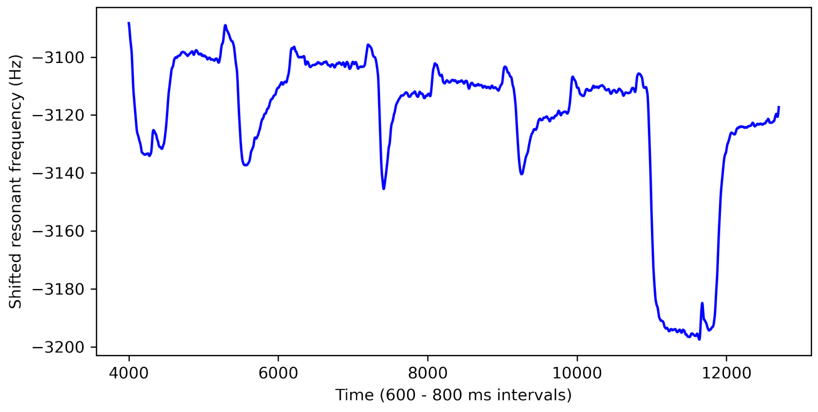


Figure 1.6: Resonant frequency with combined filters vs time (y-axis reported as RF - 10 MHz, and time reported as the index from the beginning of measurement)

CHAPTER 2

Preparation for classification

One of the most popular methods to assign data into one of two states (e.g., positive and negative samples) is to use **a machine learning (ML)** [19]. This method requires a small subset (also known as labels y) of all elements to be manually classified by more complex (and slower) algorithms or alternatively, by humans. This subset is then divided into two parts: **a training set and a testing set**. The ML algorithm can "learn" parameters from the training set by creating **a model** and then **the predictions** (results labeled by a model) are tested against known observations in the testing set.

2.1 Labeled plot of resonant frequency

The experimental protocol for measurement is divided into several stages. The preparation of the chip, including the addition of bioreceptors and the conditioning of the surface film, takes up to 50% of the total analysis time (varies from measurement to measurement). Discussion of this portion of the signal is beyond the scope of this thesis and furthermore, not needed for classification, as the clinical sample is not involved.

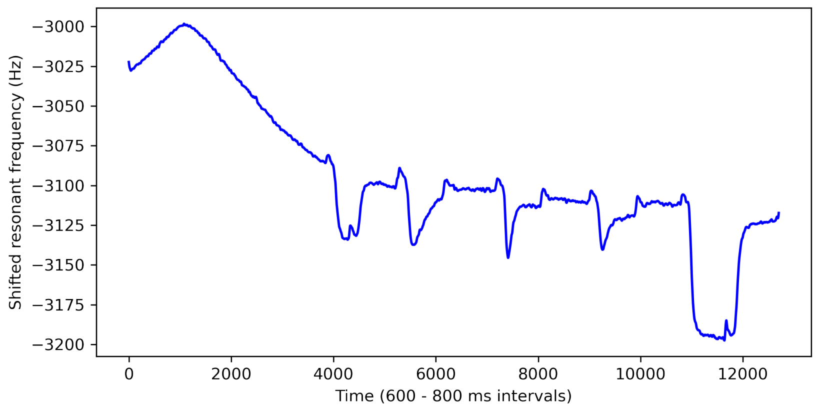


Figure 2.1: Example measurement

The other half does react with clinical samples and thus is the topic of interest for this thesis. Data taken from a single clinical measurement is shown in figure 2.1, this data (i.e. a sensorgram) represents the injection of five clinical samples into the sensor (each surrounded by injections of control buffer solution). These five injections can be seen as the "valleys" in the sensorgram, leading to drops in the signal by 30-100 Hz. These valleys correspond to when control buffer (non reactive on the surface) is exchanged with a clinical sample which may contain the targeted pathogenic material - in this case biological material composing the viral structures causing Covid-19. The first valley is a control that is used for determining **the baseline**. The second one is an injection with a negative control: a sample that certainly does not contain pathogen. The third and fourth valleys correspond to tested samples collected from a public transport vehicle, and the last valley is a positive control: a sample that definitely contains coronavirus.

The biggest issue with the data available to this thesis is that the device did not save any information about when certain events (such as change the liquid, start of the pump and others) occurred. The intervals adjacent to each valley must be manually selected by humans, because in some cases measurements followed a different procedure than normal, which is not suitable for later analysis. The manual labeling of these intervals is time consuming. When this process of injecting samples in the laboratory will be automated, an operating software can easily save information when the injection happened (followed by change in the RF). Here we define the data for segments of interest as $S_i, i = 1..5$, where due to the collection protocol, the length of each segment was not constant (due to non-constant experimental protocols).

It would be useful to have an automated method that would incorporate segmentation of each sensorgram (the RF over time) into a preprocessing phase. This approach would need to be universal, however, is very difficult as the experimental results do not always go as planned. For example, if a bubble appear during the injection the RF will change in an unexpected fashion. In addition, the process of measurement is constantly changing at the Institute of Physics according to the project needs.

For this thesis, segmentation by hand is in place until there exists a standardized way of pre-selecting these data. Figure 2.2 shows manually selected intervals (referred as **the segments**) of the plot. Each segment starts with the red line and ends with the black one.

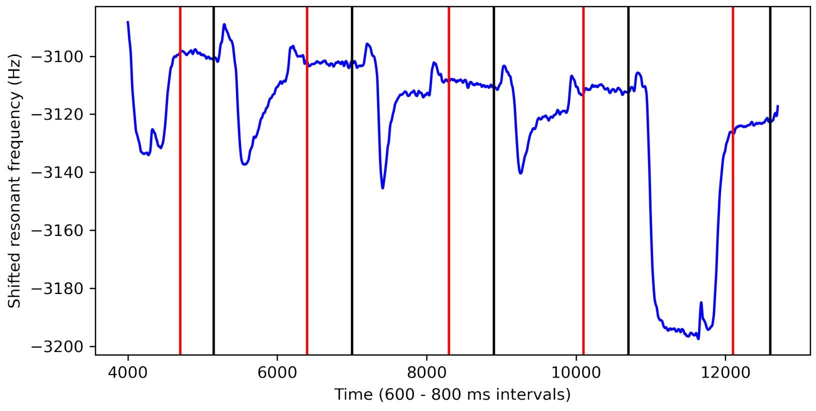


Figure 2.2: Part of the measurement with the segments

2.2 Feature vectors

In order to classify data, one must first determine **the feature values** (sometimes only referred as features) that form **the feature vectors** used in classification. A feature value is typically a number that represents a property of the observation. In this case, several types of feature values are used. The requirement is that the feature vectors must have the same length even if the corresponding segments has different lengths.

The segments should ideally remain constant over time, but a drift, caused by changes in temperature and other uncontrollable causes, is often present. It can be approximated as an linear function in form of $cx + d = 0$ using the least squares regression. This method gives two parameters c_i and d_i, where $i = 1..5$ is index of the segment. The parameter c_i represents the slope in the interval, while the parameter d_i is mainly affected by calibrated resonant frequency. Every chip has different resonant frequency due to inconsistencies in the factory process. In order to compare two measurements with different calibrated frequency, the parameter d_i must be normalized by subtracting arithmetic mean of previous segment $i - 1$. This will also help to reduce effect of linear drift.

Another appropriate feature values are arithmetic mean of the values $mean_i = \frac{1}{n}\sum_{j=1}^{n} S_{i,j}$, the minimum $\min S_i$ and the maximum $\max S_i$, where $i = 1 \ldots 5$ is the index of segment and n is its size. All of these values are also normalized against previous segment.

To sum up, the feature values are c_i, $d_i - mean_{i-1}$, $mean_i - mean_{i-1}$, $min_i - mean_{i-1}$, and $max_i - mean_{i-1}$. Since negative and positive controls are always known (second and last segment, respectively), they naturally create two classes for classification. These values create five dimensional feature vectors.

If there is a negative sample, the difference in the mean of S_i between two following segments should be close to zero. In other case (a positive sample) a change is observed. The negative control is compared with the baseline segment, which only exists as an reference for the negative control and it is not compared with any other interval. The last comparison is between the positive control and the previous segment with an unknown sample from a public transport vehicle. For the last comparison it does not matter if the unknown sample was positive or negative, as there was an injection of control solution between each sample injection: the change in pathogenic mass before and after the final injection is almost entirely due to the addition of mass during the final injection (where the rate at which pathogenic material falls off the chip is relatively small) [20]. The idea presented here is thus to use only positive and negative control to train a ML algorithm.

2.3 Training and testing set

Definition 2.3.1 (Dataset)**.** A dataset is a list of feature vectors commonly described as an matrix X.

Definition 2.3.2 (Labels)**.** A labels are a list containing only 1 or 0 (for two classes) commonly defined as a vector y.

Definition 2.3.3 (Length of dataset)**.** The dataset length (and also length of labels) is defined as n.

Definition 2.3.4 (Classification)**.** Classification analysis is when the outcome of a model is a discrete number (class 1 or 0) to which the data belongs.

Definition 2.3.5 (Regression)**.** Regression analysis is when the outcome of a model is a real number (e.g., price of an asset or amount of the pathogen on the chip).

Positive and negative feature vectors are separated into training set, which is used for training machine learning algorithm, and testing set, which is used for determining the overall quality of algorithm. The training set is usually bigger than the testing set.

To sum up, the standard procedure is similar to following

1. split the dataset into a training and a testing set,
2. train a model using only the feature vectors from the training set,
3. compare predictions of a model with known observations from the testing set.

2.3.1 Evaluation of a model

Performance of a trained model can be evaluated by using the **mean squared error (MSE)**. If predictions are close to the observed data, the MSE would be lower. This can be expressed as [21]

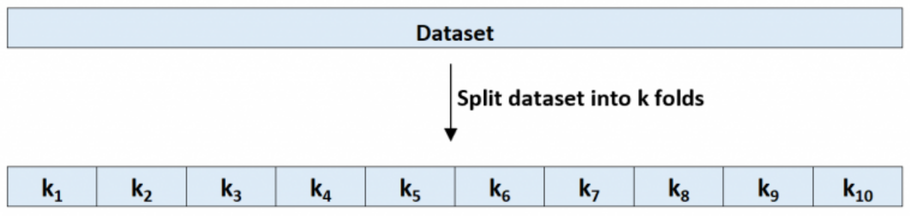
$$MSE = \frac{1}{n}\sum (y_i - f(x_i))^2,$$

where n is a size of training set, y_i is a i^{th} observation and $f(x_i)$ is prediction of a model based on feature vector x_i. The MSE measures the performance of predictions of a model based on previously unseen data. The issue here is that the training and the testing set can be easily affected by **a selection bias**. Only by splitting the dataset in a different fashion, the model would produce diverse accuracy. This inconsistency could result in an inadequate and unrealistic assumption of the MSE.

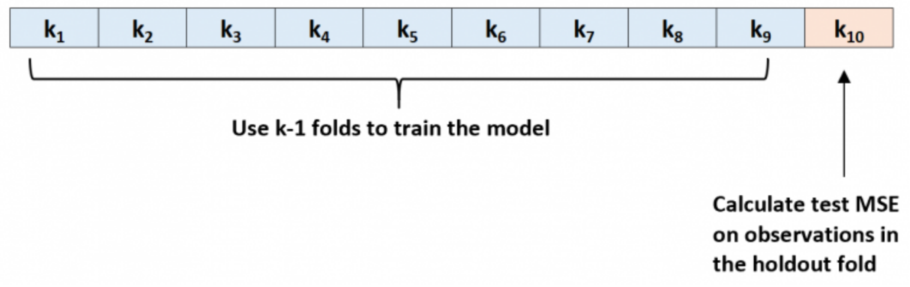
2.3.2 k-fold cross-validation (CV)

A popular method for decreasing the selection bias is k-fold cross-validation [22]. It is a statistical method that estimates the precision of prediction of a model. It is commonly used in machine learning predictive modeling due to its simplicity and reliable results with a lower bias [23]. The procedure is made up of following steps [21]:

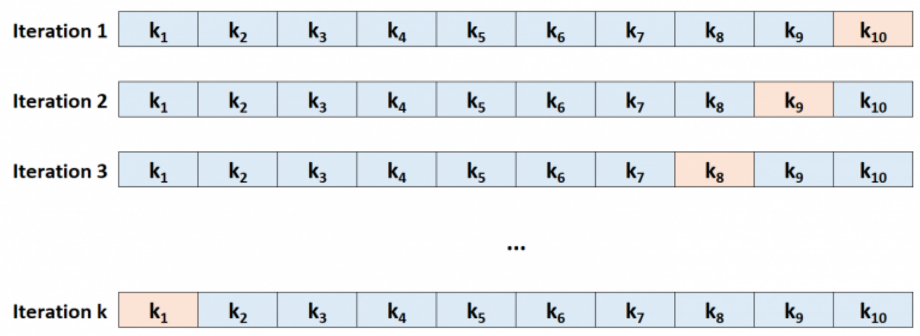
1. Randomly shuffle the dataset and divide it into k groups (or folds) with a similar size.



2. Choose one holdout (testing) set and train the model on the rest of the folds. Calculate the MSE


k1
k2
k3
k4
k5
k6
k7
k8
k9
k10
Use k-1 folds to train the model
Calculate test MSE on observations in the holdout fold

3. Iterate through the folds k times using a different holdout set each time



4. Take the average of the measured MSEs

As the result, the overall MSE is calculated as

$$MSE_{overall} = \frac{1}{n} * \sum MSE_j,$$

where k is number of folds and MSE_j is the j^{th} iteration of MSE. The standard deviation is often included with the mean on MSE. Values of k are usually 5 or 10 [24]. Lower values of k lead to lower variance, albeit at the expens of higher bias. Similarly, the higher variance comes with higher k, but gives lower bias. This phenomenon is known as bias-variance tradeoff [25]. There are numerous modifications of k-fold cross validation. One of them is called **leave-one-out** cross-validation, where $k = n$, so that every feature vector creates its own holdout set [23]. Other variations of the algorithm include stratified, nested or repeated k-fold CV.

Chapter 3

Machine Learning methods

Definition 3.0.1 (Statistical machine learning)**.** Statistical ML finds predictive function based on statistics and functional analysis of the data.

Definition 3.0.2 (Empirical risk)**.** The error on the training dataset is called the empirical risk. This is always known.

Definition 3.0.3 (Supervised learning)**.** Supervised learning uses known labels to train algorithm that later predict outcomes as accurately as possible.

Definition 3.0.4 (Unsupervised learning)**.** An unsupervised learning algorithm discovers patterns for clustering or association problems without prior knowledge of the data. It does not need labels to be trained.

There are several suitable algorithms to analyse the regions. Artificial neural networks are currently popular, but they require high amount of training data, which cannot be obtained in this case. Suitable classifiers for this application are support vector machine, adaboost, random forest, and k-nearest neighbors. These four will be explained in detail and then compared between themselves.

3.1 SVM (Support vector machines)

Definition 3.1.1 (Hyperplane)**.** In a D-dimensional space V, a hyperplane is a subspace with dimension $n-1$. In context of SVM, a hyperplane is an affine subspace which divides V into two half spaces. A hyperplane can be defined mathematically by

$$\mathbf{w}^T x - b = 0,$$

where $\mathbf{w} \in \mathbb{R}^D$ is a normal vector to the hyperplane and $b \in \mathbb{R}$ is the bias.

Support vector machine [26, 27] is a machine learning algorithm used for both classification and regression, but the former use is more widely approached. This supervised learning model was developed by Vladimir Vapnik and his colleagues at AT&T Bell Laboratories. It falls under the category of statistical machine learning. SVM maps feature vectors from a training set to points in a space and then finds a hyperplane that divides the set into two categories. This approach is similar to a perceptron, but SVM also maximizes the gap (hard-margin) between the two classes. It can work not only with linear classification (given by the hyperplane), but also with non-linear one using the so called **kernel trick** [28] that map input vectors to a higher dimension. Soft-margin can even work on non-linearly separable data.

3.1.1 Kernel trick

When a dataset is not linearly separable in the original feature space, the datapoints can be mapped to a higher-dimensional space where a new separating hyperplane can be sought. However, this can become computationally expensive if approached in a naive way. This problem can be solved by using the kernel trick.

To determine an SVM classifier's parameters the data must be transformed to a matrix of scalar products of the individual datapoints. Let $\theta : X \to V$ be a feature map, where X is the original feature space and V if the new higher dimensional space. Instead of first computing $\theta(x)$ for each individual datapoint x and then $\langle\theta(x), \theta(x')\rangle_V$ for each pair of data points, explicit formula for a function K given by

$$K(x, x') = \langle\theta(x), \theta(x')\rangle_V$$

can be derived. Such function is called a kernel function. Using the optimized expression for K can reduce computational cost significantly.

One of most used kernels, **the radial basis function kernel (RBF)** or Gaussian kernel, is defined as

$$K(x, x') = exp(-\frac{\|x - x'\|^2}{2\sigma^2}),$$

where $\|x - x'\|^2$ is a squared Euclidean distance, $x, x' \in X$, σ is free parameter (sometimes expressed as $\gamma = \frac{1}{2\sigma^2}$. This kernel is used in the code as it performs well on a given dataset.

3.1.2 Hard-margin

A hard-margin classifier in linearly separable data finds two parallel hyperplanes that separates classes so the distance between them is maximized. The region between these two boundaries is called the margin. The maximum margin hyperplane lies exactly between them. Assuming a normalized dataset, the equations that describe these boundaries are

$$\mathbf{w}^T x - b = 1,$$

and

$$\mathbf{w}^T x - b = -1.$$

Anything above (or below) is sample of class with label 1 (or -1 respectively). The distance between boundaries is $\frac{2}{\|\mathbf{w}\|}$. Thus, in order to maximize this number, one must minimize the parameter $\|\mathbf{w}\|$ with constraints

$$\mathbf{w}^T x - b \geq 1, \text{ if } y_i = 1$$

and

$$\mathbf{w}^T x - b \leq -1, \text{ if } y_i = -1.$$

A simpler form is then

$$y_i(\mathbf{w}^T x - b) \geq 1, \text{ for } i \in 1..n.$$

This problem falls within the category of optimization problems and can be expressed as

$$\text{Minimize } \|w\| \text{ subject to } y_i(\mathbf{w}^T x - b) \geq 1, \text{ for } i \in 1..n.$$

After finding the w and b parameters, the classification for given x is done by

$$x \mapsto sgn(\mathbf{w}^T x - b),$$

where $sgn(\cdot)$ is the sign function.

3.1.3 Soft-margin

Hard-margin can be extended to cases when the data are not linearly separable by using a hinge loss function in the form of

$$\max\{0, 1 - y_i(\mathbf{w}^T x_i - b)\}.$$

This function returns 0 if the data point is correctly classified (i.e., lies on the correct side of the margin.

The optimization task then changes to

$$\left[\frac{1}{n}\sum\{\max 0, 1 - y_i(\mathbf{w}^T x_i - b)\}\right] + C\|w\|^2,$$

where C is the trade-off between correctly classifying the data point and increasing the margin size.

3.1.4 Properties

Both hard- and soft- margins shows that SVM always finds the global optimum on a training set. The size of a feature vector is not limited and can work with infinite-dimensional spaces. Only $\mathbf{w}$ and b must be known to perform a prediction, which is fast to compute, and trained model uses a little of storage space. In addition to this, the algorithm can be modified to use more than two classes or be used for regression.

3.2 Random forest

The member of ensemble learning algorithm is random decision forest [29, 30], which can serve as a classifier as well as regressors. It constructs a multiple decision trees when learning and outputs a class with majority of votes from each tree, when used in classification. Random forests are in a way extension of the decision trees, does not suffer from overfitting and thus generally perform better. The first algorithm of this type was developed by Tin Kam Ho in 1995 using the random subspace method. In 2006, the term ”random forests” was registered as an trademark by Leo Breiman and Adele Cutler, who developed an extension of Ho’s algorithm. They require almost no configuration and are simple to use. Their predictions are reasonable across variety of applications.

3.2.1 Decision tree

Decision tree goes from root, uses branches for different observations (based on outcome of a test in node) and leaves for final decision. One way to think about decision trees is to treat them as an series of conditional statements. They are not only popular in machine learning, but also in operations research or decision analysis. Single decision tree with high depth tend to overfit as it has low bias, but high variance, and it is sensitive to noise.

3.2.2 Bagging

A way to decrease variance, but with the same bias is a method called bagging (or bootstrap aggregating). Choose a natural number B and select random sample of an X with replacement B times. Then fit the trees to selected samples. This could be expressed as an iterative two step algorithm

For $b = 1..B$

1. Take portion of training set X and y named X_b, y_b respectively
2. Build a decision tree f_b from X_b and y_b

Last step is to take unseen examples $x' = X$
$\{X_b\}$ and predict them by taking the mode of trained trees (also known as taking the majority rule). These trees are not correlated as they are not trained from the same training set. This method is called bootstrap sampling. The B parameter can be found using the cross-validation.

3.2.3 Random forests

Random forests use very similar technique as the bagging one, but they use modified learning algorithm that selects a random subset of features. This process is called feature bagging. A reason for doing this is that in some cases one or more feature values are strong predictors. Hence, they will be selected in many trees and that trees become correlated.

The feature bagging use only a portion of features from a feature vector. If feature vector has length p, then the idea is to use only $\sqrt{p}$ (rounded down) of features, but the number depend on properties of the classified data.

3.3 k-nearest neighbors

Definition 3.3.1 (Manhattan distance)**.** The Manhattan distance is distance between two points. The distance d of points $x, y \in \mathbb{R}^D$, where D is dimension of the space in Cartesian coordinates, is calculated as

$$d(x,y) = |x-y| = \sqrt{\sum_{i=1}^{D} |x_i - y_i|}.$$

Definition 3.3.2 (Euclidean distance)**.** The Euclidean distance in Euclidean space is distance between two points. The distance d of points $x, y \in \mathbb{R}^D$, where D is dimension of the space in Cartesian coordinates, is calculated as

$$d(x,y) = |x-y| = \sqrt{\sum_{i=1}^{D} (x_i - y_i)^2}.$$

The k-nearest neighbors algorithm [31, 32] was developed by Evelyn Fix and Joseph Hodges in 1951 and later extended by Thomas Cover. This non-parametric method is used in classification as well as regression. In classification, the outcome is based on majority of the k nearest neighbors and will be assigned to the most common class among them. The number k is typically small. This method depends on the distance of the feature vectors in space. Normalizing the datasets can drastically improve performance of a model. Neighbors can be assigned with weights showing importance of an neighbor, so the closer neighbor is, the more important is. Commonly used function to compute weights is $\frac{1}{d}$, where d is distance between input vector and its neighbor. The k-NN is sensitive to noise in training data as it could create a local cluster of wrong classified feature vectors.

The distance metric could be Euclidean distance (for continuous data), Hamming distance (for discrete data) or others. Algorithms Large Margin Nearest Neighbor or Neighborhood component analysis can learn metric distance and improve overall model accuracy.

One problem of the majority voting based on the neighbors is when one class has significantly more members than the other class. An input has much higher probability to have more samples of the first class in its neighborhood. One way to overcome this issue is to use weightening function that takes into account distance between the input and its neighbors. Another approach is to apply self-organizing map (SOM).

3.3.1 Selection of k

The parameter k depends on input data and can be properly selected by heuristic functions. The higher the k is, the lower significance of noise is, but the boundaries between classes are less distinct. Special case is when $k = 1$. This method is called the nearest neighbor algorithm. In binary classification, the k should be odd number to avoid problems with equal votes. The k value can can selected empirically by using the bootstrap method.

3.4 Results

Every method was tested using the k-fold algorithm with random shuffling in order to lower the selection bias. The parameter k was set to 10. A leave-one-out approach could produce better results, but at the price of potential overfitting. The results (in percentage) are rounded to two decimal places and may vary through the section, because the feature vectors are shuffled. The number of measurements used in classification is 53.

These three algorithms were chosen because they represent a unique point of view in the machine learning world. The SVM tries to separate the two classes from each other by drawing an hyperplane between them. The random forests construct multiple decision trees based on observations and then opts for the class with most trees that predicted it. On the other hand, the k-NN looks at the points clustered around an input to be classified and decides on the basis of which class has the majority of the neighbors to belong to.

The implementation was left to the SciKit learn Python library, which provides additional classifiers such as AdaBoost, Gaussian Process, and others. However, even if using the library, one should provide correct parameters for the methods. They were chosen by evaluating different parameters and then compared between themselves. The function from the library accepts more parameters than the ones discussed here. Some of them were omitted, because their default values are already reasonable or are too specific to be tweaked and mentioned here. If not mentioned, the default values provided by the library were used.

Figure 3.1 shows the manually selected segments for 40 measurements (of 53 total). The overall dataset should be larger, but the process of measuring in the laboratory is demanding both on personnel as well as time; the size of dataset will be bigger in the future. It can be seen that the sensorgrams for each plot are cleary different from each other. One can see drifts in the resonant frequency that are unique to each sensorgram.

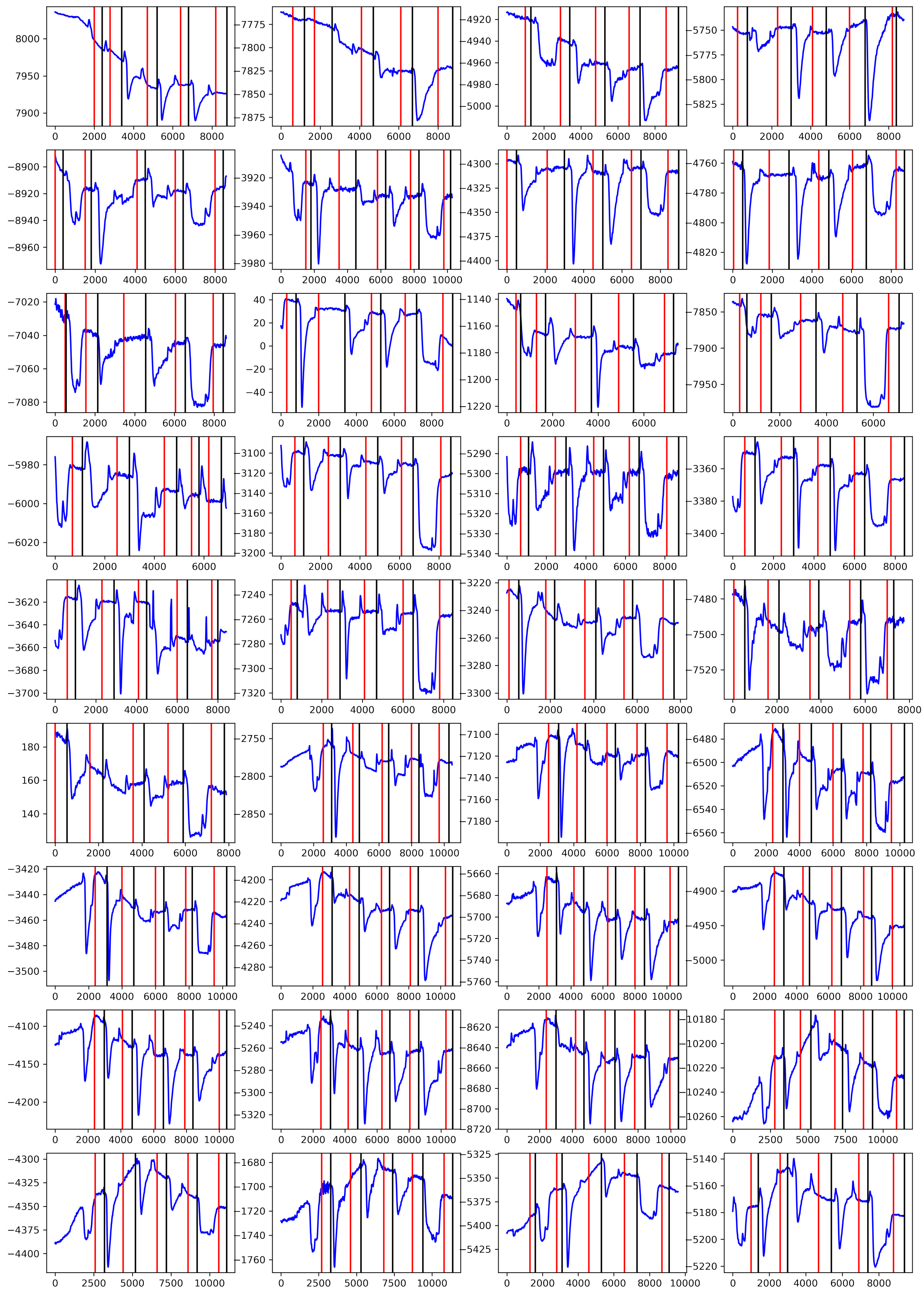


Figure 3.1: Overview of 40 datasets along with manually selected segments

3.4.1 Parameters for the SVM

As was already stated, the SVM accepts C parameter as a cost of wrongly classified points (applies only to soft-margin used here) and kernel function, which is one of radial-based (RBF), polynomial (poly) or linear. The C parameter behaves similarly throughout different kernels, with $C = 50$ as the optimal parameter. The best combination is with the RBF kernel.

Table 3.1: MSE on testing set with various SVM parameters

Kernel / C	0.5	1	5	10	50	100
RBF	9.04 %	9.04 %	9.04 %	8.21 %	**6.2 %**	8.54 %
Poly	15.4 %	14.66 %	12.04 %	11.09 %	9.99 %	7.6 %
Linear	13.05 %	15.05 %	15.05 %	15.05 %	15.05 %	15.05 %

3.4.2 Parameters for the random forest

Random forest takes number of trees as the only input here. The classification is considerably good in every case, but the $n = 25$ trees preforms the best. The random forest is highly sensitive to randomly generated values used in the algorithm, so the results are not stable and change with every execution.

Table 3.2: MSE on testing set with various random forest parameters

10	25	50	100	150
8.51 %	**6.69 %**	9.5 %	10.14 %	9.54 %

3.4.3 Parameters for the k-NN

The results when using Manhattan or Euclidean distance are close to each other, however the distance for the former is slightly better. A much bigger impact on the quality of classifier is the number of neighbours; looking at only one nearest neighbour leads to insufficient performance. Apart from that, the rest of the results are similar, but for the best precision one should use $k = 5$ neighbours. This value was superior when using both distances.

Table 3.3: MSE on testing set with various k-NN parameters

Distance / k	1	3	5	7	9	11
Manhattan	15.43 %	8.05 %	**6.61 %**	7.51 %	10.57 %	7.34 %
Euclidean	16.25 %	6.77 %	6.68 %	7.67 %	9.2 %	7.09 %

3.5 Comparison of the best results

3.5.1 Unfiltered resonant frequency

Even when using the segments with the unfiltered signal (resonant frequency vs time) the accuracy is at least 70 % as seen in table 3.4. The SVM and the random forest have similar results on the testing set. The k-NN is slightly worse. The MSE follows the same trend as the error. The low error on training set is due to the use of data that has already been seen; this number is usually lower than the error on testing set, but is also less significant.

Table 3.4: Error on unfiltered RF

Algorithm	Testing set	Training set
SVM (RBF kernel, $C = 50$)	29.0 %	19.38 %
Random forest (25 trees)	27.36 %	0.00 %
k-NN ($k = 5$, Manhattan)	26.45 %	19.08 %

Table 3.5: MSE on unfiltered RF

Algorithm	Testing set	Training set
SVM (RBF kernel, $C = 50$)	10.9 %	3.8 %
Random forest (25 trees)	7.73 %	0.00 %
k-NN ($k = 5$, Manhattan)	7.81 %	3.68 %

3.5.2 Filtered resonant frequency

The error rates on the signal filtered with median filter and then with Savitzky-Golay filter are lower. The biggest change can be seen in the SVM classifier. The relative improvement is 24 %. For this filtered data, the random forest performs very similarly to the unfiltered data. The k-NN experienced 10 % increase of classifying precision. The errors on the training set are almost negligible.

Table 3.6: Error on filtered RF

Algorithm	Testing set	Training set
SVM (RBF kernel, $C = 50$)	22.0 %	18.24 %
Random forest (25 trees)	26.73 %	0.31 %
k-NN ($k = 5$, Manhattan)	23.73 %	18.77 %

Table 3.7: MSE on filtered RF

Algorithm	Testing set	Training set
SVM (RBF kernel, $C = 50$)	6.2 %	3.35 %
Random forest (25 trees)	9.11 %	0.00 %
k-NN ($k = 5$, Manhattan)	6.61 %	3.54 %

Conclusion

Usefulness and future research

Quality and quantity of dataset

Clearly, better results would be given via the use of a larger dataset. However, the laboratory measurements are slow to process, so the expected size of dataset is a few hundred samples at most. This calls for models, those used in this thesis, that can be trained on small datasets. Another problem discovered by this thesis is that knowledge of change in the resonant frequency are necessary to avoid manual segmentation: not only is manual segmentation time consuming, but it can also introduce an additional bias.

It is recommended to use standardized measurements in order to increase the accuracy of predictions. With a performance over 70%, this method can be compared to antigen tests of the Covid-19 presence [33]. On the other hand, the specialists in the laboratory can examine the curve of the resonant frequency more thoroughly: analysis of similar data revealed that the precision of the QCM crystals can be as good as a PCR test. This analysis conducted by humans requires expertise which should be incorporated with the machine learning algorithms and together provide fully automated, fast, and reliable performance.

Lower level of the noise

Another goal of the additional research would be to further lower the noise. For this adjustments to both the hardware and software portions of the sensor are required. There are aspects of the electronics that are outside of the scope of this the thesis. However, the experts from Institute of Physics claims that the noise will be lower and with combination of software tweaks, the results could be close to the performance of the PCR tests. One of the troubles with current approach is that the curve is drifting mainly because of the changes in the temperature, which results in significant shifts in the resonant frequency. A method how to minimize this effect was discussed, but there is definitely room for improvement.

The temperature compensation will be addressed not only in the software, but mainly in the electronics itself.

Regression

The classification can only decide the class (positive or negative) of the sample. However, the information about an amount of the present pathogen is very insightful. In this regard regression algorithms can be used for non-binary decisions. Their output is a real number, in this case a quantity of a measured pathogen. All of the methods presented here can be modified to serve as an regressors. The whole interval of real number can be split into smaller sub-intervals. Each sub-interval then describes a category. The prediction then falls into one of four or five categories depending on "how much positive" the sample is. This approach is called an histogram.

Multi-class classifiers can be used in the case that the output would belong to one of the intervals (the case of the histogram). These classifiers do not have binary output, but rather have several classes to which the predicted data belongs.

Results evaluation

The characteristics of quartz piezoelectric biosensing crystals were discussed, as well as how these sensors can be adjusted to detect coronavirus in complex samples. Devices built on this basis could help control the coronavirus spread, which is currently a global problem, by functioning as a cheap alternative to commonly used antigen or even PCR tests. The effectiveness of the methods used in this thesis surpasses 70% accuracy on the test dataset, i.e. QCM devices clearly have the potential to outcompete antigen tests in regards to precision.

The path from a raw text output of an experimental QCM to a decision whether a sample is positive or negative has been described; this includes an explanation of the data format and the process of converting the data into a form suitable for AI analysis. A reliable method for computing the resonant frequency was introduced, as well as how to overcome the challenges encountered in their usage. To reduce the level of noise, a portion of the raw data was fitted to a parabola using the least squares regression. A Savitzky-Golay filter, combined with a median filter, then proved to reduce the noise even more.

In the current state, the data preparation phase is perhaps where most improvement can be done. Hyperparameters in this process were chosen experimentally by hand, further improvement is to be expected in an attempt of more sophisticated calibration. In addition, all of the noise filtering methods used in this thesis are based on the least squares method. Using more rigorous statistical methods may reduce noise even more.

Before the analysis itself, the data was splitted into the training and the testing set with the help of the k-fold cross-validation algorithm to mitigate selection bias. The eligible features formed the feature vectors used in the learning phase of the artificial intelligence model. The mean squared error was defined as a mean of squared ratios of the wrongly predicted data points by the model. This value was then used to compare the accuracy of the classifiers. All of these methods are commonly used to ensure robustness of the final model.

The support vector machines divides points in space into two categories with the use of the kernel trick and soft-margin, which leads to some points to be wrongly classified. The random forest algorithm works in a way as to build a number of decision trees and, by a majority vote, decides if the sample is positive. The last algorithm, k-nearest neighbor, looks at its neighbors and classifies the sample based on its observations. All of these methods showed considerable results and performed with less than 30% error rate.

For a continuation of this research, the data in this thesis suggest that in order to improve accuracy, more samples need to be gathered and in much better form. In the available data, individual experiments are often inconsistent with events occurring at largely different times, which adds variance to the noise observed and makes classification considerably harder. Additional features on which to train the classifier could also be supplied, but this would have to be followed by additions to the available dataset to avoid overfitting. Adjustments in the electronics of the QCM device should decrease noise and provide clearer signal with less noise. With a large enough dataset, the classification can be substituted by regression algorithms that would not only show if the sample is positive, but also provide information how much of the pathogen is present in the sample.

Bibliography

[1] Lan Lan et al. “Positive RT-PCR test results in patients recovered from COVID-19”. In: *Jama* 323.15 (2020), pp. 1502–1503.

[2] Andreas Janshoff, Hans-Joachim Galla, and Claudia Steinem. “Piezoelectric mass-sensing devices as biosensors—an alternative to optical biosensors?” In: *Angewandte Chemie International Edition* 39.22 (2000), pp. 4004–4032.

[3] Renee L Bunde, Eric J Jarvi, and Jeffrey J Rosentreter. “Piezoelectric quartz crystal biosensors”. In: *Talanta* 46.6 (1998), pp. 1223–1236.

[4] Stuart Russell and Peter Norvig. “Artificial intelligence: a modern approach”. In: (2002).

[5] John McCarthy. “What is artificial intelligence?” In: (2007).

[6] Sotiris B Kotsiantis, I Zaharakis, P Pintelas, et al. “Supervised machine learning: A review of classification techniques”. In: *Emerging artificial intelligence applications in computer engineering* 160.1 (2007), pp. 3–24.

[7] Michael I Jordan and Tom M Mitchell. “Machine learning: Trends, perspectives, and prospects”. In: *Science* 349.6245 (2015), pp. 255–260.

[8] World Health Organization et al. “Coronavirus disease 2019 (COVID-19): situation report, 73”. In: (2020).

[9] Sebastian Raschka. *Python machine learning.* Packt publishing ltd, 2015.

[10] Thomas Kluyver et al. *Jupyter Notebooks-a publishing format for reproducible computational workflows.* Vol. 2016. 2016.

[11] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[12] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[13] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[14] Cynthia Dwork et al. "Calibrating noise to sensitivity in private data analysis". In: *Theory of cryptography conference*. Springer. 2006, pp. 265–284.

[15] Thomas H Lee and Ali Hajimiri. "Oscillator phase noise: A tutorial". In: *IEEE journal of solid-state circuits* 35.3 (2000), pp. 326–336.

[16] Sanford Weisberg. *Applied linear regression*. Vol. 528. John Wiley & Sons, 2005.

[17] Ronald W Schafer. "What is a Savitzky-Golay filter?[lecture notes]". In: *IEEE Signal processing magazine* 28.4 (2011), pp. 111–117.

[18] BI Justusson. "Median filtering: Statistical properties". In: *Two-Dimensional Digital Signal Processing II*. Springer, 1981, pp. 161–196.

[19] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[20] Todd M Squires, Robert J Messinger, and Scott R Manalis. "Making it stick: convection, reaction and diffusion in surface-based biosensors". In: *Nature biotechnology* 26.4 (2008), pp. 417–426.

[21] Zach. *An Easy Guide to K-Fold Cross-Validation*. `https://www.statology.org/k-fold-cross-validation/`. [Online; accessed 08-May-2021]. 2020.

[22] Tadayoshi Fushiki. "Estimation of prediction error by using K-fold cross-validation". In: *Statistics and Computing* 21.2 (2011), pp. 137–146.

[23] Jason Brownlee. *A Gentle Introduction to k-fold Cross-Validation*. `https://machinelearningmastery.com/k-fold-cross-validation/`. [Online; accessed 08-May-2021]. 2018.

[24] Gareth James et al. *An Introduction to Statistical Learning*. 2013.

[25] Mikhail Belkin et al. "Reconciling modern machine-learning practice and the classical bias–variance trade-off". In: *Proceedings of the National Academy of Sciences* 116.32 (2019), pp. 15849–15854.

[26] William S Noble. "What is a support vector machine?" In: *Nature biotechnology* 24.12 (2006), pp. 1565–1567.

[27] *Support-vector machine*. `https://en.wikipedia.org/wiki/Support-vector_machine`. [Online; accessed 12-August-2021]. 2021.

[28] Bernhard Scholkopf. "The kernel trick for distances". In: *Advances in neural information processing systems* (2001), pp. 301–307.

[29] Gérard Biau and Erwan Scornet. "A random forest guided tour". In: *Test* 25.2 (2016), pp. 197–227.

[30] *Random forest*. `https://en.wikipedia.org/wiki/Random_forest`. [Online; accessed 12-August-2021]. 2021.

[31] Leif E Peterson. "K-nearest neighbor". In: *Scholarpedia* 4.2 (2009), p. 1883.

[32] *k-nearest neighbors algorithm*. `https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm`. [Online; accessed 12-August-2021]. 2021.

[33] Bo Diao et al. “Accuracy of a nucleocapsid protein antigen rapid test in the diagnosis of SARS-CoV-2 infection”. In: *Clinical Microbiology and Infection* 27.2 (2021), 289–e1.

Appendix

- thesis.pdf - the thesis itself
- thesis.zip - contains LaTeX source files
- notebooks.zip - contains various notebooks used in this thesis