A Machine Learning approach to Predict Respiratory Rates from ECGs

Steven Maharaj

October 24, 2019

Supervisor: Roger Rassool

Abstract

Measuring the respiratory rate is an important part of the initial and continuing assessment of any unwell patient. Direct methods are usually very invasive so instead, ECGs can be used to infer the breathing rate. We investigate various machine learning models to predict the respiratory rate from ECG data. Meaningful features for each learner are extracted using traditional signal processing techniques.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Measuring the respiratory rate (the number of complete breathing cycles) is an important part of the initial and continuing assessment of any unwell patient. Continuous monitoring of respiratory rate can be done manually by patient observation, but is extremely time consuming and labour intensive. Alternative methods such as the monitoring of temperature, humidity or CO2 levels, or the measurement of movement, volume and tissue concentrations seek to automate measuring respiratory rate. However, these automated methods are often invasive for the patient [1]. Clinicians often use the limited information obtained from pulse oximeters. Oximeters use red and infrared light to detect the difference in absorption between oxygenated haemoglobin and deoxygenated haemoglobin to provide a measure of oxygen saturation and heart rate [2]. These devices do not directly measure respiratory rate and will only detect inadequate respiration when hypoxia (a deficiency in oxygen) occurs.

Using an electrocardiogram (ECG) is a potential alternative to monitor a patient's respiratory rate. An ECG is a measurement of the hearts electrical activity. ECGs are non-invasive, painless test that yield quick results. It is known that ECGs contain a respiratory signal [3]. An algorithm that could reliably and accurately detect the respiratory signal in a patient's ECG would be extremely useful, especially if it could be run on low-cost hardware such as a mobile phone.

In this report we construct meaningful ECG features using signal processing techniques. The meaningful features are fed into various machine learning algorithms to predict the respiratory rate in terms of a frequency. Chapter 2 will outline general biological concepts of ECGs and Respiratory measurements. Chapter 3 out lines the structure of the data a lists the required software. Chapter 4 details how we extract meaningful features using signal processing techniques and outline a machine learning architecture to predict respiratory rates as a frequency. Chapter 5 discusses the effectiveness of each learner proposed in Chapter 4. Lastly, we conclude with a report summary and possible directions for further research.

Chapter 2

Biological Preliminaries

2.1 Electrocardiogram (ECG)

An ECG is the electrical activity of the heart muscle as it changes with time. Cardiac muscles contract in response to electrical depolarisation of the muscle cells [4]. The ECG is measured by placing a series of electrodes on the patients skin. This electrical signal is amplified and recorded to yield the result shown in Figure 2.1.

Each component labelled in Figure 2.1 is associated with a biological event. In a normal heartbeat, the electrical activity starts in a small cluster of pacemaker cells. When the impulse activates the atria (upper chamber of the heart), it produces a small peek called the P wave. Next it activates the ventricles (the lower chambers of the heart). This makes up the largest part of the ECG signal (because of the greater muscle mass in the ventricles) and this is known as the QRS complex. The final T wave is a recovery period as the impulse reverses and travels back over the ventricles [4]. Lastly, the intervals between R peaks take the name RR intervals.

2.2 Respiratory measurements

There are many existing approaches for measuring a patient's respiratory rate. In this report we consider chest and abdominal respiratory signals obtained using inductance plethysmography, doronasal airflow measured using nasal thermistors, and SpO2 oxygen saturation. An inductance plethysmography are chest or abdominal belts that measure the fluctuations in thoracic circumference with respiration. Nasal thermistors are usually thermocouple sensors, nasal prongs or face masks which measure fluctuations in temperature with respiration. SpO2 oxygen saturation are monitored with differential paramagnetic sensors, fiber optic fluorescence-based oxygen sensors, or gas analysis systems [5]. Figure 2.2 shows the various respiratory measurements described above. In an inductance plethysmography a peak corresponds to an inhale while troughs correspond to an exhale. In the nasal thermistor readings a peak corresponds to an exhale while troughs correspond to an inhale [6]. Also note that the inductance plethysmographies appear to be noisier than the nasal thermistors. The Sp02 conveys information about an apnea event whereby the patient's blood oxygen saturation drops. For example, a patient stops breathing while snoring.

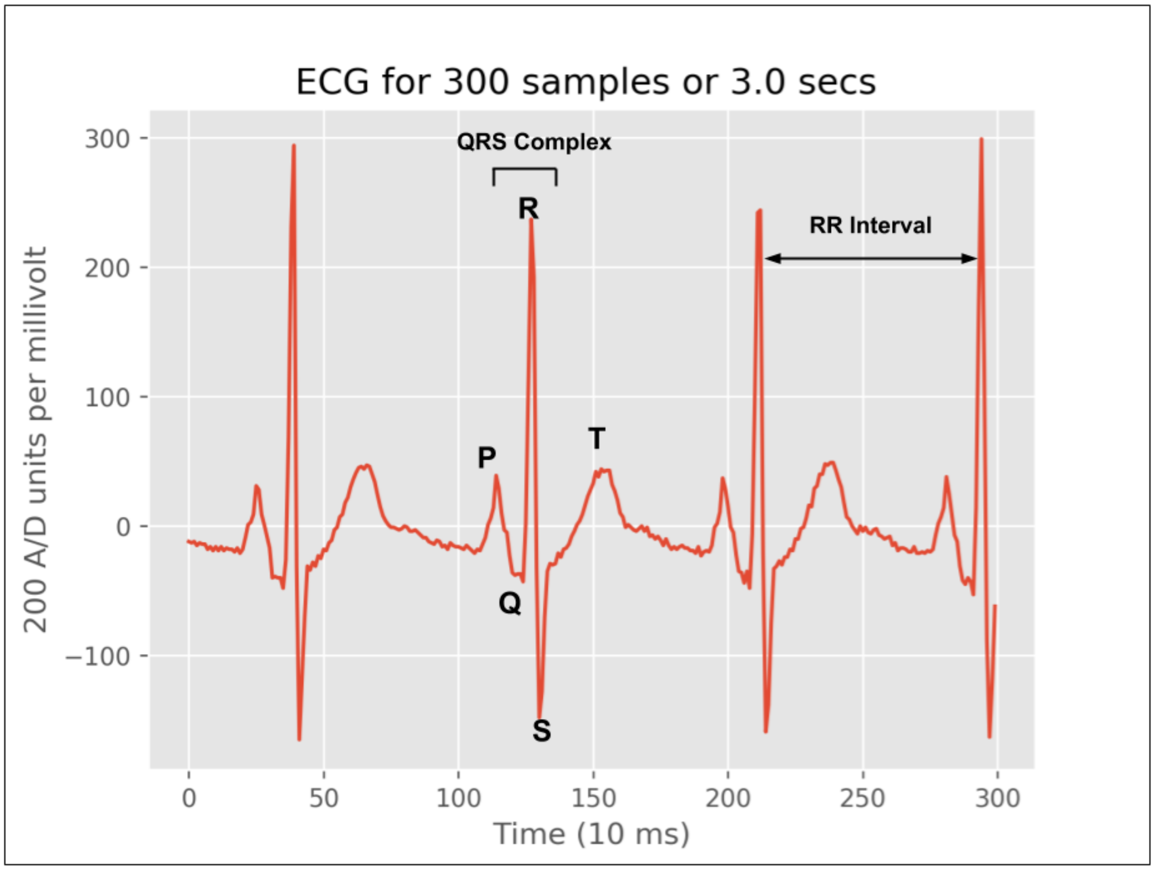


Figure 2.1: A normal (healthy) ECG. The Major components of an ECG are the P wave, the QRS Complex, the T wave and the RR interval.

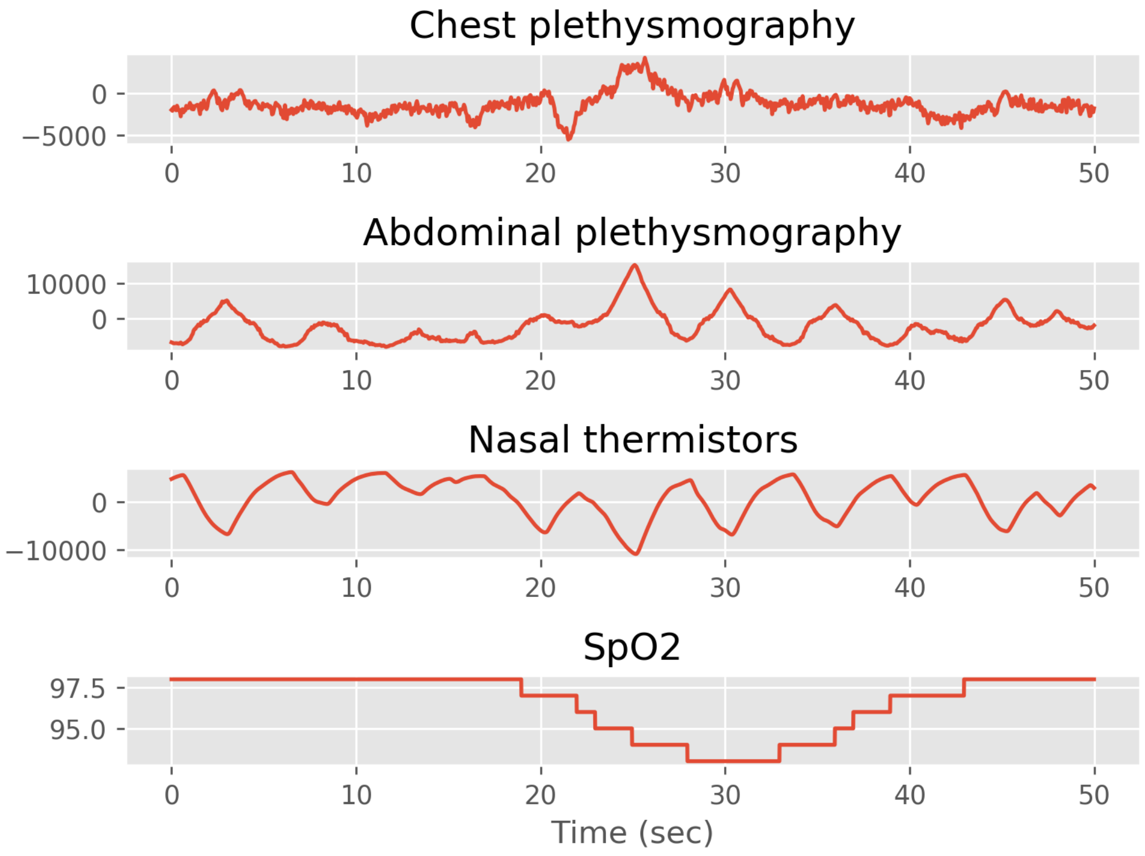


Figure 2.2: Four types of respiratory measurements: chest (RespC) and abdominal (RespA) respiratory inductance plethysmography, nasal thermistors (RespN) and SpO2 oxygen saturation.

Chapter 3

Materials

3.1 Data Description

We use the Apnea-ECG Database set provided by PhysioNet [7]. The Apnea-ECG Database was originally used by Penzel *et al.* [7] to model sleeping disorder such as sleep apnea. In 2000 the data was released to the public as a competition. The objective was to identify instances of apnea.

"The data consist of 70 records, divided into a learning set of 35 records (a01 through a20, b01 through b05, and c01 through c10), and a test set of 35 records (x01 through x35). Recordings vary in length from slightly less than 7 hours to nearly 10 hours each. Each recording includes a continuous digitised ECG signal, a set of apnea annotations and a set of machine-generated QRS annotations. In addition, eight recordings (a01 through a04, b01, and c01 through c03) are accompanied by four additional signals (Resp C and Resp A, chest and abdominal respiratory signals obtained using inductance plethysmography; Resp N, doronasal airflow measured using nasal thermistors; and SpO2, oxygen saturation). The files with names of the form `rnn.dat` contain the digitized ECGs (16 bits per sample, least significant byte first in each pair, 100 samples per second, nominally 200 A/D units per millivolt)." [7]. Files names of the form `rnnr.dat` contain the respiratory data. In the `rnnr.dat` Resp C is every fourth entry, Resp A is every fourth entry offset by 1, Resp N is every fourth entry offset by 2, SpO2 is every fourth entry offset by 3.

3.2 Software

For this project we use the programming language python. Python is an open source general purpose language which uses an interpreter to execute code. Since it is interpreted it allows for dynamic typing at the cost of speed (compared to statically typed languages such as C, C++ or Go). However, due to its recent rise in popularity over the last decade, libraries such as numpy, pandas and scipy make numerical and scientific programming more user friendly with the added bonus of speed since most operations from these libraries are implemented in C.

In order for the reader to reproduce the results of this project a package named ECG has been open sourced on Github https://github.com/StevenMaharaj/ecg.git. Documentation is provided for each function in the ECG package.

Chapter 4

Method

4.1 Data prepossessing

4.1.1 Data parsing

Files containing ECG and respiratory measurements are binary files with 16 bits per sample. For a specified offset and number of events (samples) the binary file can be read using the `read_ecg()` function inside the the ECG package. The `read_ecg()` loops over the binary file for the specified number of events reading two bytes (16 bits) for each iteration. The process of reading the data is shown in Figure 4.1.

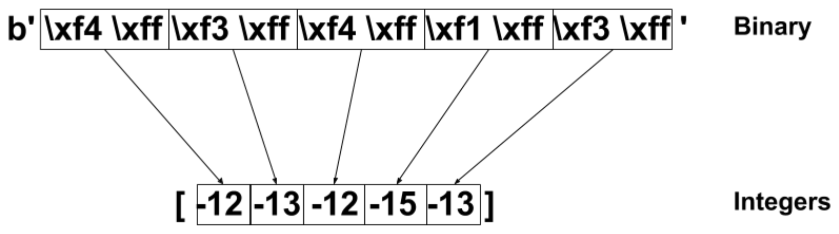


Figure 4.1: The convention of binary to integers for the first 5 samples of the file `a01.dat`
.

Once the ECG data is successfully parsed into a list (or numpy array) of 16 bit integers we extract useful features which could be later used to find the respiratory signal. The data is sampled at every R-peak. The R-peaks were found using the scipy function `find_peaks()` [1]. Features such as the R-peak heights, R-R intervals and R-peak heights changes are calculated as shown in Table 4.1.

The respiratory data is also parsed using the `read_ecg()` function to a numpy array. This numpy array was then split into four sub-arrays, each for the different types of respiratory measurements described in Chapter 3. From a graphical comparison of the respiratory data (see Figure 2.2) it was concluded that Resp N (doronasal airflow

[1]The author's numpy implementation of locating the R-peaks can be found in the module `my_find_peaks.py`.

R-peak location (sec)	R-peak height (A/D units)	R-R interval	R-peak height change
0.39	294	0.0	0
1.27	237	0.88	-57
2.12	244	0.85	7
2.94	299	0.82	55
3.77	340	0.83	41
4.65	344	0.88	4

Table 4.1: Re-sampling the ECG data (`a01.dat`) at the R-peak locations along with extracting R-peak heights, R-R intervals and R-peak heights changes

measured using nasal thermistors) contained the least noise. This is not including the Sp02 measurement since the Sp02 measurements only provide information about apnea events. Next, the respiratory data should be compared to the ECG data in order to infer the respiratory signals from the ECG data. For example one may re-sample the respiratory data at the R-peaks from Table 4.1 and look for a relationship between the ECG features and the re-sampled respiratory data.

In addition, RespN appears to be oscillating at a constant frequency but the centre of the waveform varies too much. This will affect our analysis since we only wish to consider the respiratory rate and not the absolute strength of the respiratory signal. In the next subsection we will address this concern.

4.1.2 Data filtering

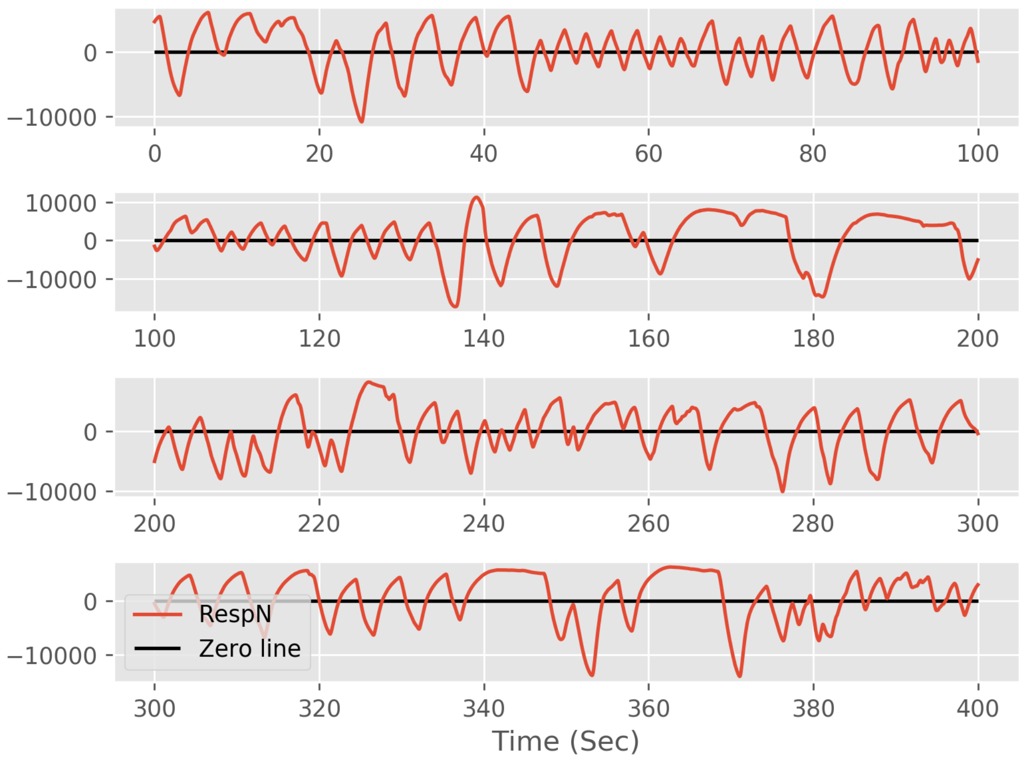
We introduce a band pass filter to centre the respiratory data around zero and eliminate other frequencies far from the respiratory signal (essentially noise).

RespN contains the signal from the patient's breathing rate along with other higher and lower frequencies. High frequencies correspond to small oscillations in the waveform while low frequencies make the centre (relative to the breathing rate) deviate from zero. The effect of low frequencies is shown in Figure 4.2a. Over a 400 second period the breathing rate is not always centred at zero. This could cause problems as an algorithm may confuse an inhale and exhale. For example, in Figure 4.2a between 0 and 20 seconds a section of RespN is all above the zero line. Hence, if zero is used as a reference point to differentiate inhales and exhales, a trough sampled in this section (between 0 and 20 seconds) may be mistaken for an exhale. Note that picking a different reference point will cause the same confusion in another section of the data. The solution to this problem is to filter out the low frequencies.

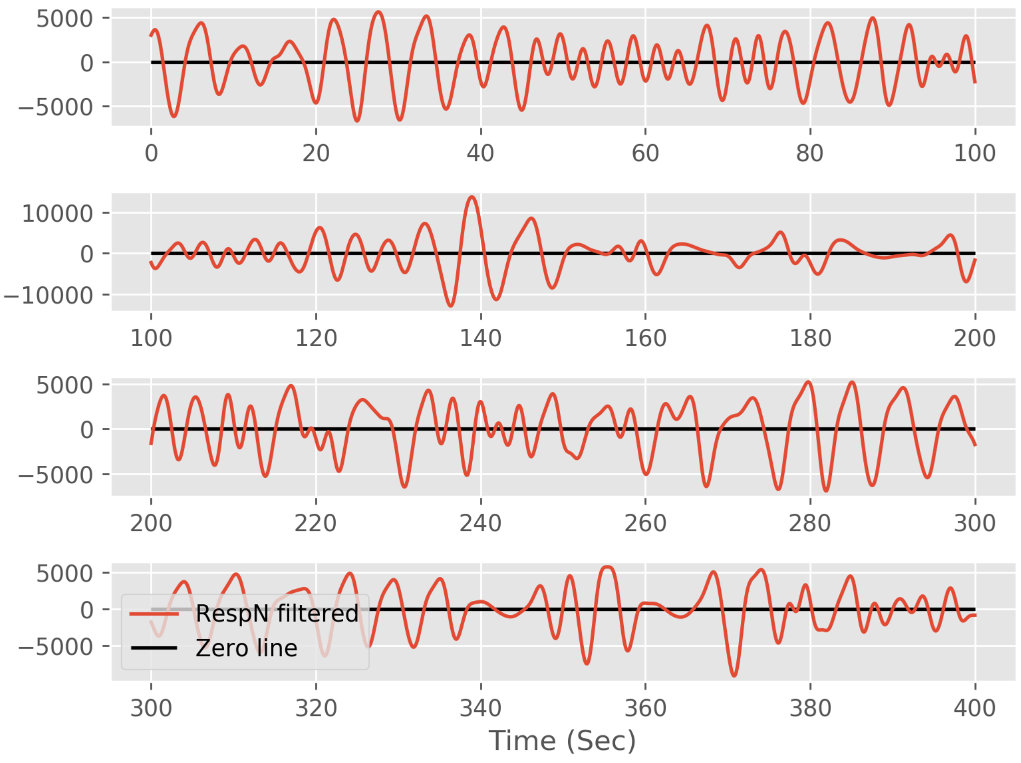
RespN was passed through a band-pass butter worth filter. The lower cut off frequency was 0.1 Hz and the higher cut off frequency was 0.5 Hz. Choosing these frequencies assumes that the duration of a patients breath (the beginning of and inhale to the beginning of the next inhale) is between 2 seconds and 10 seconds. Also note that the range from 0.1 Hz to 0.5 Hz was chosen because Mirmohamadsadeghi *et al.* [8] used a band-pass filter with these frequencies in there previous work with ECG data to predict respiratory signals. Figure 4.2b illustrates the used of the band-pass filter to centre RespN.

4.1.3 Fourier transform

A Fourier transform is a signal processing technique which decomposes a function on the time domain into a function in the frequency domain. Since we are working with cyclic data it is appropriate to incorporate Fourier transforms



(a) RespN



(b) Filtered RespN

Figure 4.2: (a) RespN and (b) Filtered RespN over 400 seconds. The red line is RespN and the black line is the zero line. The file `a01r.dat` was used.

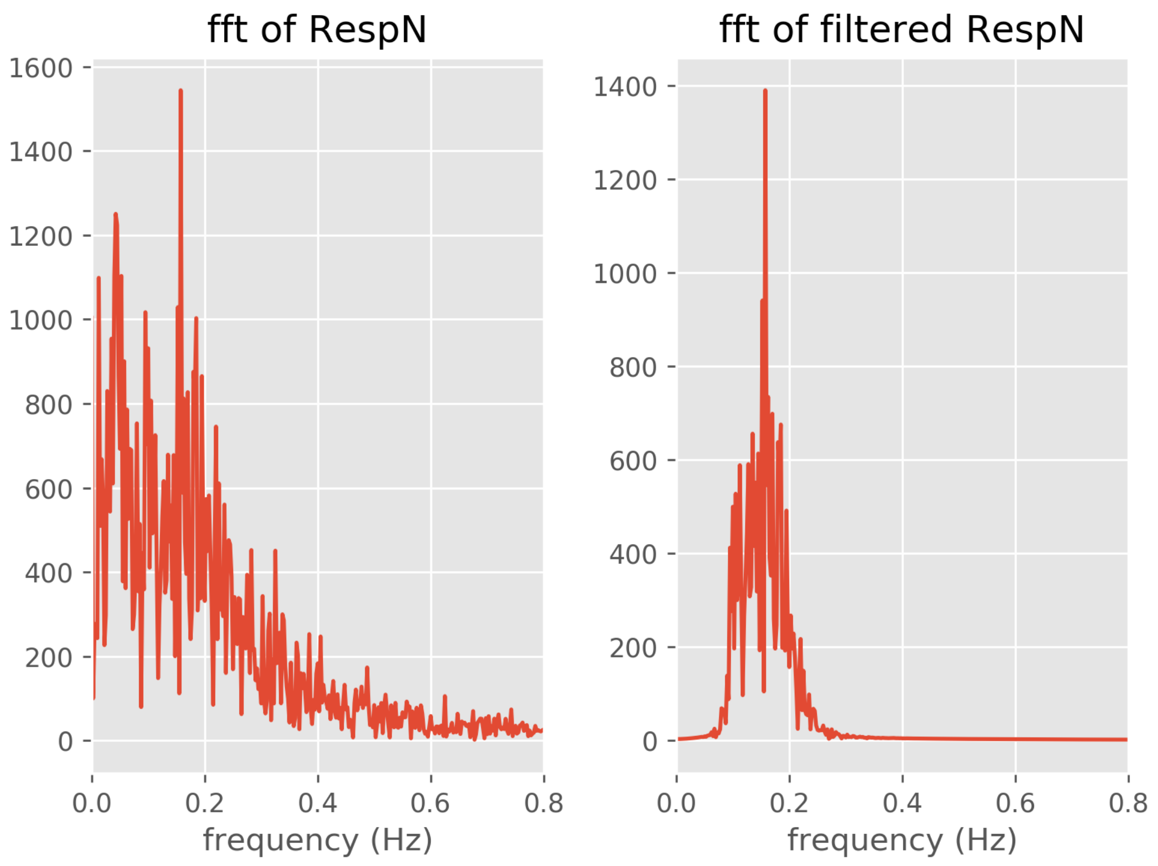


Figure 4.3: The left figure is and FFT of RespN. The right figure is an FFT of fitered RespN

of the ECG data as a feature of our model. However, the transform involves continuous integrals but in practice computers and digital processing systems can only work with finite sums. This problem is over come by making use of a fast Fourier transform (FFT) which is an algorithm that computes the discrete Fourier transform [9]. Figure 4.3 shows the results of an FFT applied to the RespN and filtered RespN. We see that the band-pass filter was able to eliminate noise in the frequency domain. The figure on the right more clearly shows the dominant frequency of RespN which is a little bit less than 0.2 Hz. Hence, it is reasonable to conclude that for the window of time in Figure 4.3 (400 seconds) the breathing rate was little bit less than 0.2 Hz (a little more than 1/5 of a breath per second).

4.2 Modelling

We propose various machine learning models that seek to identify the respiratory frequency from the ECG data. Data files a01 through a04, and c01 through c03 are split into 20 second windows (batches of 2000 data points). All windows are fed into the band-pass filter described in subsection 4.1.2. From the filtered ECG data we extract the mean R-R intervals and identify the dominate frequency using a FFT from subsection 4.1.3. Note that Appendix A provide some mathematical justification as to why the RR-intervals are included as input features. From the filtered respiratory data we also extract the dominate frequency. All data are stacked into one large data set and each 20 second window will be assumed to be independent of each other.

A proposed learner will seek to predict the dominate frequency of a filtered respiratory window given the mean R-R interval and dominate frequency of a filtered ECG window. All prediction from the proposed learner will be evaluated using the mean absolute error (MAE). Note that we are constructing pre-trained models. That is the learner will be trained on the large data set then the learned model parameters are saved to disk. At a later time they can be deployed without needing to re-train the model.

The learners considered in this report are Random Forests, Elastic Net Regressions, Support Vector Machines (SVMs) and Neural Nets. Appendix B provides a short overview of each learner. Visually, the model set-up is given by the following word equation.

$$\text{Respiratory frequency} = \text{learner}(\text{ECG frequency}, \quad \text{R-R intervals}).$$

Also, we will define our benchmark learner to be that the dominate frequency of a filtered respiratory window is equal to the dominate frequency of a filtered ECG window.

Chapter 5

Results and Discussion

Table 5.1 provides information about the accuracy and CPU time for each learner. We see that the benchmark accuracy is already reasonably low at 0.1 Hz. On a closer inspection of the data set the dominate frequency for some of the filtered respiratory windows is equal to (or very close to) its corresponding ECG window. Thus a simple FFT would provide a somewhat effective prediction for the respiratory frequency at a very low computational cost. However, in order to improve accuracy we had to resort to more complex models.

All learners achieved the same accuracy up to 0.001 Hz and beat the benchmark. The random forest achieved the highest accuracy but the elastic net was the fastest learner. Since the accuracy's of each learn are very close one could argue that that the elastic net is the best learner since it is approximately 20 times faster than the second fastest learner (the SVM).

Compared to the other learners the elastic net is relatively simple (a lower number of trainable parameters). Therefore, it is more suitable to train on data sets with small sample sizes. Recall, that the data set are sampled at 100 Hz for approximately 10 hours. Thus we approximately have $100 \times 3600 \times 10 = 3{,}600{,}000$ samples per data set. However, the learners from Table 5.1 use aggregating statistics from each non-overlapping 20 second window. Hence, the sample size is reduced to approximately 1800 samples per data set. Also, the number of train samples are reduced once the data has been split into train validation and test sets.

Given a small data set, very complex models such as the neural net are not appropriate (unless provided with more data) since they achieve roughly the same accuracy at an enormous computational cost. The computational speed and memory usage are important factors when choosing a model to predict the respiratory frequency. Ideally we would like to deploy our model on a light weight device such as a mobile phone or a hospital monitor. Even

Learner	MAE (Hz)	CPU time
benchmark	0.1363	62 μs
ElasticNet	0.0831	2.43 ms
Random Forest	0.0820	129 ms
SVM	0.0836	50 ms
Neural net	0.0829	1min 17s

Table 5.1: (Middle column) The MAE each learner. (Right column) CPU time for each learner to be trained and predict the respiratory frequency on a test data set.

though we constructed pre-trained models, fast predictions are still required. Appendix C describes a prototype web application which could ideally be written for light weight devices.

Chapter 6

Conclusions

6.1 Report summary

In this report we investigated various machine learning models to predict the respiratory rate from ECG data. In order to construct meaningful inputs for a learning algorithm, ECG features needed to be extracted. The learners under investigation had two input features: the dominate filtered ECG frequency and the mean RR-interval. In order to extract the dominate filtered ECG frequency the data was parsed through a band-pass filter and a FFT mapped the data on to the frequency domain. The dominate filtered ECG frequency was extracted for each 20 second window. In order to extract RR-intervals a peak finding algorithm would seek the R-peaks and record the distance between each R-peak (forming the RR-intervals). The mean of the RR-intervals for each 20 second window was extracted.

Four different learners were assessed on the basis of accuracy and computational efficiency. It was concluded that the elastic net was the best learner since all learners achieved approximately the same accuracy but the elastic net was more computationally efficient than the other learners.

6.2 Further research

We will list possible research directions to extend the work in this report.

- The inclusion of more input features: Other input feature such as the R-peak height could be used as a potential input to a learner. In order to construct meaningful features, one should further review the literature and perform data analysis seeking to find relationships between the ECG data and the respiratory data.
- Overlapping windows: Using overlapping windows will significantly increase the size of the data set. Hence, more complex model such as the neural net could achieve a much higher accuracy.
- Model deployment: Further details can be found in Appendix C.
- A full grid search of the hyper-parameter space: In order to select the best model, hyper-parameters need to be optimised. A naive grid search can be found in the Ipython notebook `grid_seach.ipynb`.

Appendix A

Feature Correlations

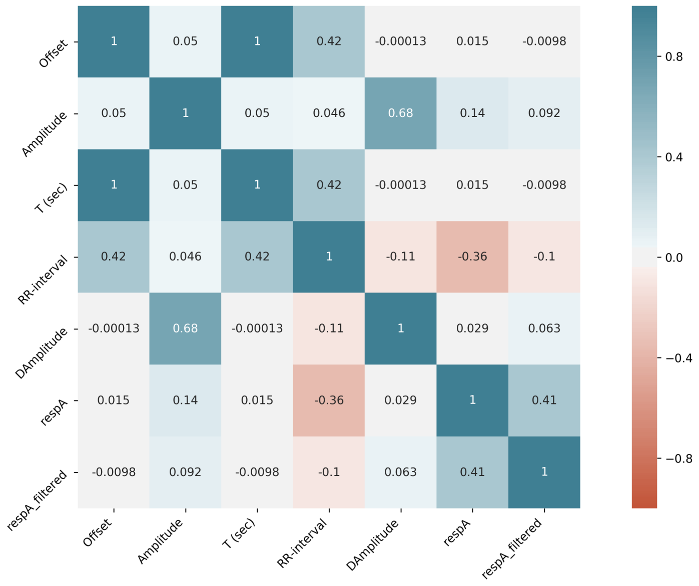


Figure A.1: A heat map of correlations of the re-sampled data at the R-peaks. For one hour of data from the file `a01.dat`

.

Figure A.1 shows a heat map of correlations of the re-sampled data at the R-peaks. We observe that the RR-intervals are somewhat correlated with the respiratory data. Thus, along with previous investigations [3] and the observed correlation it is appropriate to include the RR-interval as an input feature to a learner.

Appendix B

Learner Descriptions

In this appendix we provide a short outline of each learner used in this report. Refer to and Bishop [10] for details about the mathematics of each learner and refer to Geron [11] for python implementations of each learner.

B.1 Random Forest

A random forest is a collection of decision trees (a crude learner with low variance but high bias). It averages (in the case of regression) all the predictions from the decision trees to yield the prediction of the random forest (Bootstrap aggregating). We used a random forest with 100 trees each with a maximum depth of 3.

B.2 Elastic Net Regression

An Elastic net regression is a regularised regression model whereby the regulariser term is a linear combination of a Ridge (L2 norm) and a LASSO (L1 norm) regulariser. We used a penalty term of 0.5 and the L1 ratio was 0.5.

B.3 Support Vector Machine

A Support Vector Machine constructs a set of hyper-planes about the data. For the regression problem its seeks to minimise the norms of the hyper-planes' gradient subject to constraints relating to data points close to the hyper-plane (the margin). An rbf kernel was used to count for non-linearities in the data. The penalty term was set to 1.

B.4 Neural Net

A Neural net is a network connected by weights which represent a feature's relative importance. Neural Nets make predictions via a forward pass. Using those predictions it minimises a specified loss function through a backward pass. The architecture of the neural net used in this report is given by the python code (keras) below.

```python
model = tf.keras.models.Sequential([
  tf.keras.layers.Dense(1024, activation='relu'),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dropout(0.3),
  tf.keras.layers.Dense(512, activation='relu'),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dropout(0.3),
  tf.keras.layers.Dense(256, activation='relu'),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dropout(0.3),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dropout(0.1),
  tf.keras.layers.Dense(32, activation='relu'),
  tf.keras.layers.BatchNormalization(),
  tf.keras.layers.Dropout(0.3),
  tf.keras.layers.Dense(1, activation='relu')
])
```

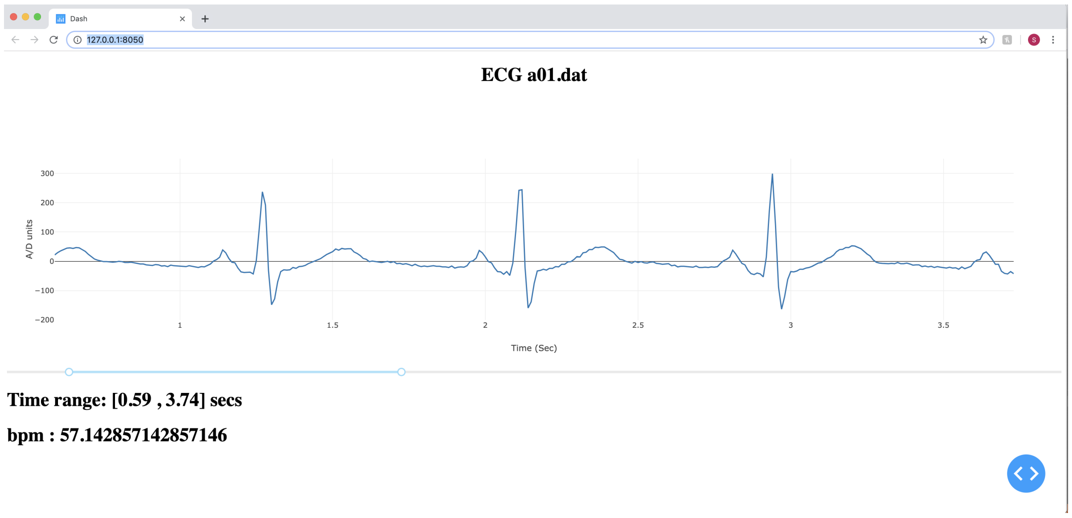
Listing B.1: Neural net architecture

Appendix C

Web Application

Figure C.1a shows a screen shot of the web application of an ECG while C.1b shows a screen shot of the web application of respiratory data. The graph at the top displays the data for a given time window. The window can be adjusted using the slider below the graph. For each specified window the application automatically calculates the beats or breaths per minute.

Further work on this application should include incorporating the learners from Chapter 4 to automatically calculate the breaths per minute in the ECG application. Next one would deploy the application to a light weight device for real time analysis. However, the application in Figure C.1 uses synchronous code making it slow and unsuitable for real time analysis. One solution to this problem is to use multi-threading so processes can run separately via careful memory management of a buffer. A second alternative is to use coroutines. Coroutines are cooperatively multitasked, whereas threads are typically preemptively multitasked. Thus, coroutines allow execution to be suspended and resumed which is the case for our problem. The advantage is that coroutines do not involve any system calls or any blocking calls.

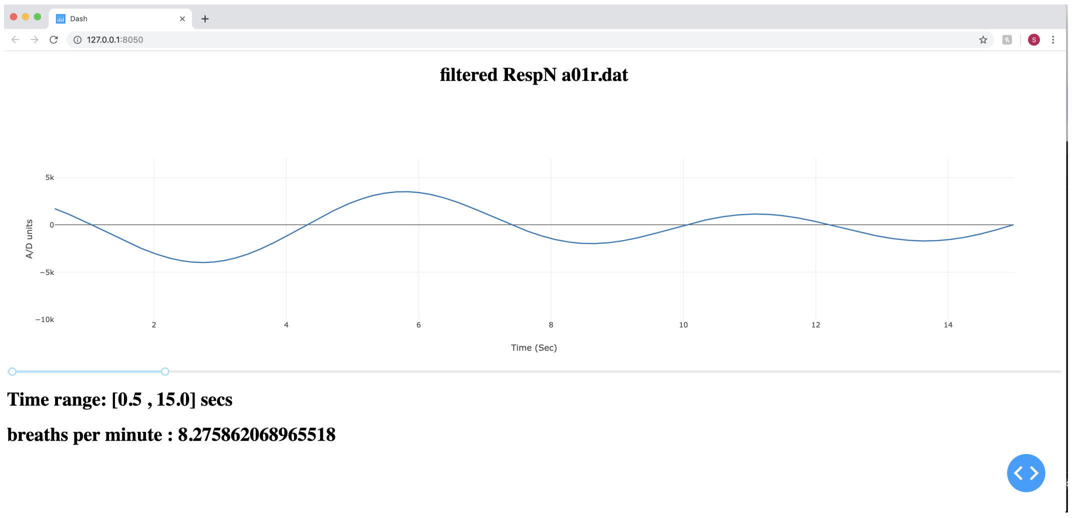


ECG a01.dat

Time range: [0.59 , 3.74] secs

bpm : 57.142857142857146

(a) ECG



filtered RespN a01r.dat

Time range: [0.5 , 15.0] secs

breaths per minute : 8.275862068965518

(b) RespN

Figure C.1: (a) Web application for the ECG data and (b) Web application for RespN.

References

[1] M. Folke, L. Cernerud, M. Ekström, and B. Hök, “Critical review of non-invasive respiratory monitoring in medical care,” *Medical and Biological Engineering and Computing*, vol. 41, no. 4, pp. 377–383, 2003.

[2] P. Leonard, T. Beattie, P. Addison, and J. Watson, “Standard pulse oximeters can be used to monitor respiratory rate,” *Emergency medicine journal*, vol. 20, no. 6, pp. 524–525, 2003.

[3] P. H. Charlton, T. Bonnici, L. Tarassenko, D. A. Clifton, R. Beale, and P. J. Watkinson, “An assessment of algorithms to estimate respiratory rate from the electrocardiogram and photoplethysmogram,” *Physiological measurement*, vol. 37, no. 4, pp. 610–626, 2016.

[4] D. Price, “How to read an electrocardiogram (ecg). part 1: Basic principles of the ecg. the normal ecg,” *South Sudan Medical Journal*, vol. 3, no. 2, pp. 26–31, 2010.

[5] A. S. Ginsburg, J. L. Lenahan, R. Izadnegahdar, and J. M. Ansermino, “A systematic review of tools to measure respiratory rate in order to identify childhood pneumonia,” *American journal of respiratory and critical care medicine*, vol. 197, no. 9, pp. 1116–1127, 2018.

[6] E. Jovanov, D. Raskovic, and R. Hormigo, “Thermistor-based breathing sensor for circadian rhythm evaluation,” *Biomedical sciences instrumentation*, vol. 37, pp. 493–498, 2001.

[7] T. Penzel, G. B. Moody, R. G. Mark, A. L. Goldberger, and J. H. Peter, “The apnea-ecg database,” in *Computers in Cardiology 2000. Vol. 27 (Cat. 00CH37163)*, pp. 255–258, IEEE, 2000.

[8] L. Mirmohamadsadeghi and J.-M. Vesin, “Real-time multi-signal frequency tracking with a bank of notch filters to estimate the respiratory rate from the ecg,” *Physiological measurement*, vol. 37, no. 9, p. 1573, 2016.

[9] R. N. Bracewell and R. N. Bracewell, *The Fourier transform and its applications*, vol. 31999. McGraw-Hill New York, 1986.

[10] C. M. Bishop, *Pattern recognition and machine learning.* springer, 2006.

[11] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* ” O’Reilly Media, Inc.”, 2017.