Obdelava biomedicinskih slik in signalov

Aleš Kert

Assignment 1 Report

Ljubljana, 2021

Chapter 1

Report on Assignment 1

1.1 Abstract

In this report we describe our implementation of the QRS complex detection algorithm first described in the paper *An efficient selection, scoring, and variation ratio test algorithm for ECG R-wave peak detection* authored by Ding et al. We describe the algorithm in steps and the parameters we have chosen for our system. The implemented system was tested on the LTST DB, which has given us $Se = 95.17\%$ and $+P = 97.26\%$, and the MIT-BIH Arrythmia DB, which has given us $Se = 98.89\%$ and $+P = 95.08\%$.

1.2 Introduction

Detection of QRS complexes in an ECG signal is an important problem in the field of cardiology. Often, a cardiologist might order a patient to wear a device that records the said signal in order to diagnose them and any potential cardiovascular diseases they might have. Detection of said complexes is also useful in live-feed signals. These recordings create large volumes of data, which would be very time intensive to analyse manually, so an implementation of a detection algorithm would save hours of tedious work for each recording. We chose to implement an algorithm presented in the paper *An efficient selection, scoring, and variation ratio test algorithm for ECG R-wave peak detection* from Ding et al.

1.3 Methods

The described algorithm introduces four novel techniques - Haar-like filtering, R-wave peak sifting, regularity test for adaptive thresholding and variation ratio test, used in that order.

The first step is to extract the baseline from the ECG signal. In our variation we use a simple high-pass recursive filter for drift suppresion, we described at the lectures. We then take the acquired signal and apply a Haar-like matched filter, defined as

$$h(n) = \begin{cases} c & \text{for } -B_1 \leq n \leq B_1 \\ , -1 & \text{for } B_1 < |n| \leq B_2 \\ , 0 & \text{otherwise} \end{cases} \tag{1.1}$$

where $B_1 = 0.025 f_s$, $B_2 = 0.06 f_s$ and $c = \dfrac{2(B_2 - B_1)}{(2B_1 + 1)}$.

In the second step we take the filtered signal, marked as $y(n) = x(n) * h(n)$, and use it to calculate the following scoring function

$$s(n) = y(n)[x(n) + c_1 x_2(n)], \tag{1.2}$$

where c_1 is a constant, set at 0.55, and x_2 is the second derivative of the signal. We then take the calculated scoring function and use it to sift through the signals and find the possible R-peak candidates by checking if the score at a given point is a local maximum(by absolute value).

In the third step, we use our new list of peak candidates to calculate an adaptive threshold. For that we use the formula

$$threshold = W_1 W_2, \tag{1.3}$$

$$W_1 = T + S_5, \tag{1.4}$$

$$W_2 = \beta_1 + \beta_2 \left| \frac{n - m_1}{I_e} - round\left(\frac{n - m_1}{I_e}\right) \right|, \tag{1.5}$$

$$I_e = \sum_{h=1}^{5} \tau_h (m_h - m_{h+1}), \sum_{h=1}^{5} \tau_h = 1, \tag{1.6}$$

where T is a constant and S_5 is the 5th largest absolute scoring function value in the last 10 seconds of the signal. β_1 and β_2 are positive constants and $m_1 > m_2 > m_3 > m_4 > m_5$ are five most recently detected peaks. $\tau_1 \geq \tau_2 \geq \tau_3 \geq \tau_4 \geq \tau_5$ are constants. A peak is detected when $|s(n)| > threshold$.

In our system the constants are $\beta_1 = 0.4$, $\beta_2 = 2.9$, $T = 0.75$ and $(\tau_1, \tau_2, \tau_3, \tau_4, \tau_5) = (0.45, 0.25, 0.15, 0.1, 0.05)$.

In the final step we use a variation ratio test to remove any peaks detected due to noise. We do this using the following formula

$$\Omega = \frac{u_1}{u_2}, \tag{1.7}$$

$$u_1 = max\{x[n] | m - 0.1 f_s \leq n \leq m + 0.1 f_s\} - min\{x[n] | m - 0.1 f_s \leq n \leq m + 0.1 f_s\}, \tag{1.8}$$

$$u_2 = \sum_{n=m-B+1}^{m+B} |x[n] - x[n-1]|, \tag{1.9}$$

and use Ω to determine whether the detection was caused by noise or not. If the detection is genuine, Ω should be close to 0.5. In our system we accept peaks that produce an Ω that is within 0.275 of 0.5.

1.4 Results

We ran the implemented algorithm on both the LTST Database and the MIT-BIH Arrythmia Database and got the following results

Database	Sensitivity[%]	Positive prediction[%]
MIT-BIH DB	98.89	95.08
LTST DB	95.17	97.26

Table 1.1: Comparison of the performance on the given databases.

With our chosen parameters, we weren't able to replicate the results presented in the paper, although when we analyse the results record by record, we notice something interesting - the results of the detection are either really good or they are far worse in only certain records, which drags the performance from an average of >99% to a less impressive 95% or lower. This can be seen in the first 15 records of MIT-BIH, found in the Table 1.2

Here, we can see the records 106, 107, 113 and 114 deviate heavily from the rest and throwing off the final measurements. The reason behind this isn't entirely clear(to me at least), but we(I) suspect it might have to do something with the chosen baseline extraction method. In the first version of the algorithm, the baseline extraction was accidently left out of the algorithm, which produced similiar results, with a few records having atrocious performance, while others nearly perfect. Maybe a better choice of extraction would yield better results.

But on the other hand, the free choice of the parameters could be the crux of the problem and we just weren't able to find the optimal combination of parameters to get the same results as the authors of the paper. Many combinations were tried and the chosen ones were simply the least suboptimal parameters we found.

Database	Sensitivity[%]	Positive prediction[%]
100	99.95	99.95
101	99.93	99.87
102	99.84	100.00
103	99.94	98.01
104	99.41	100.00
105	98.70	95.25
106	98.64	76.78
107	97.36	69.03
108	97.36	96.52
109	99.67	99.86
111	99.27	98.44
112	99.81	99.91
113	99.40	53.16
114	89.59	99.86
115	99.88	100.00

Table 1.2: MIT-BIH record comparison. Full list of records available in mit-bih/results.txt.

1.5 Discussion

The implemented method has a few clever solutions, which could also negatively affect how the algorithm performs, namely the system where the distances between the last 5 predicted peaks are used to determine whether the next peak candidate is appropriate or not. This could affect how the first few peaks in a record are determined, since we are only relying on the first R-peak sifting to determine the first 5 peaks. If that turned out to be a problem in some cases, an alternate system could be implemented for the first few predictions. But besides that, the algorithm still achieves a great performance(in the paper).

With regards to our implemented system, I am sure it could be improved with a better(and maybe more systematic) parameter combination choice. It would also be interesting to see if different baseline extraction techniques affect the effectivness of the system.

A last minute thought we had, was that the reason for such different performances on the two databases was the different sampling rates, which are 360 for MIT-BIH and 250 for LTST. We think this might be significant with the β_1 and β_2 parameters, which determine the weight of the distances between the last 5 detected peaks and our current candidate, since in LTST, the distances are smaller due to a lower sampling rate. For this reason we again ran the detection algorithm on the LTST DB with the parameters $\beta_1 = 0.3$, $\beta_2 = 2.45$, $T = 0.5$, and got the following results.

Database	Sensitivity[%]	Positive prediction[%]
LTST DB	97.00	93.28

Table 1.3: Performance of the modified model on the LTST DB.

There is some room for improvement with regards to the choice of the parameters based on the sampling rate of the given record.