**Adaptive threshold estimation for QRS-complex detection in a single-lead ECG using an online Expectation Maximization algorithm.**

# Introduction

QRS complex detection is a necessary stage in almost any automatic ECG processing procedure. The positions of R waves found at this stage are used either directly, e.g. for a pulse rate analysis, or as an anchor points for discovery and analysis of other ECG features. In both cases the overall quality of the processing is totally dependent on the quality of the QRS detection. That’s why, despite the decades of previous efforts, QRS detection algorithms still remain the hot subject for many researchers [links].

So far the best accuracy, comparable to trained human, is achieved in methods based on machine learning [links]. But they have essential deficiencies – the high computational complexity and the poor portability between computing platforms. So there still exists high demand for advances in the area of low-complexity detectors, the ones that can be implemented in a low-power microcontroller.

The golden standard among low-complexity detection methods is a Pan-Tompkins (PT) algorithm described in [link]. Its main features are

* Low computational complexity
* Good accuracy
* Independence on absolute value of the signal
* Tolerance to noise

Besides having become ubiquitous in practical applications, the PT algorithm have also defined a general framework for building QRS detectors. Since its introduction in 1985, numerous attempts were made to improve on one or more aspects of the PT, resulting in many spinoff algorithms (e.g. [links]).

The framework defined by the PT consists of the following 3 processing stages:

1. Preprocessing and estimating a QRS indicator function (IF). The latter should take the high values at the positions of QRS waves and be as low as possible elsewhere.
2. Finding the peaks of the IF
3. Classifying those peaks as corresponding to QRS complexes or noise. Then the QRS-positive peaks are fitted to the closest peaks in original ECG and reported as R wave locations.

In the present work we will focus mostly on the analysis and advancement of the 3-rd stage and will also make a few notes on the 2-nd. We will show that the PT’s approach to classifying local peaks is essentially based on a statistical modeling with the online variant of the K-means algorithm with K=2.  
We will show that such approach works due to bimodal distribution of the local peak values in the IF.

Taking the idea of statistical model further, we will introduce a more elaborate statistical model and will suggest a method for its online fitting. We will also introduce a measure of the instant ECG quality based on the parameters of the fitted model. Finally, we will suggest the complete QRS detection algorithm based on our approach, and compare its performance to the other existing algorithms.

# The outline of the following sections

**The background**

1. **Estimating a QRS indicator function**  
   Overview of a few existing approaches.
2. **The properties of the distribution of the IF peaks**   
   Examples of typical distributions. Bimodality in normal ECG. Effect of noise.
3. **Classification via cluster analysis**  
   K-Means algorithm. Online K-Means, definition and formulas. Connection to the PT’s classification method.
4. **Classification via continuous distribution modeling**  
   The EM algorithm, general formulas. Solution for Gaussian Mix Model. Connection to the K-Means.

**The findings**

1. **Online EM algorithm**  
   Derivation of formulas for online fitting of GMM. Experiments with artificial signals.
2. **QRS detection algorithm**Ensuring stability of the OEM. Using peak intervals as independent discriminator. Complete algorithm.
3. **Results and discussion**
4. **Future work**

# Estimating a QRS indicator function

Typically, the QRS detection algorithms start with converting an original ECG signal into a form where the QRS waves are emphasized while another ECG features and noise are suppressed, called hereafter an indicator function (IF). The highly desired features of the IF are low computation complexity and, optionally, causality (dependence only on current and previous samples of the original signal). The latter is important for real-time applications.

The design of the IF is based on typical properties of the QRS waves that distinguish them from the other ECG features. These properties include amplitude range, slope, duration, morphology, intervals and power spectrum.

|  |  |
| --- | --- |
| *Figure 1, QRS and non-QRS spectra* | |
| qrs_spectrum | qrs_power_ratio |
| Left – the power spectrums of QRS-only and non-QRS parts of ECG. Right – the ratio of the QRS/non-QRS power spectrums. Obtained by averaging a PSD for the first 9 records in the MIT-BIH Arrythmia Database. | |

Let’s consider some existing approaches to designing the IF.

**The classic approach by Pan and Tompkins,** as described in [link]. It consists of 3 steps:

1. the ECG signal is bandpass-filtered in the frequency range 5-15 Hz in order to emphasize the QRS waves in the frequency domain, see fig.1;
2. the squared derivative is found, which emphasizes the high slope of the QRS waves;
3. the squared derivative is integrated by a flat averaging filter of length Fs\*0.15 which roughly corresponds to the maximum duration of a QRS wave.

The examples are shown in fig.2.

The method and its variants are of widespread use because of their simplicity and good performance. The main drawback of this approach is a tendency to suppress the ectopic QRS complexes because of their lower slope (see fig.2)

|  |  |  |
| --- | --- | --- |
| *Figure 2, examples of the IF* | | |
| *C:\Users\Windows 10\AppData\Local\Microsoft\Windows\INetCache\Content.Word\IF_Peaks_mitdb-105_500_504.png* | *C:\Users\Windows 10\AppData\Local\Microsoft\Windows\INetCache\Content.Word\IF_mitdb-208_1617_1619.6.png* | |
| C:\Users\Windows 10\AppData\Local\Microsoft\Windows\INetCache\Content.Word\IF_mitdb-102_219_222.png | | C:\Users\Windows 10\AppData\Local\Microsoft\Windows\INetCache\Content.Word\IF_mitdb-105_1204_1209.png |
| (a) top-left – normal ECG, (b) top-right – ectopic complex, (c) bottom-left – paced beat, (d) bottom-right – motion noise | | |

<\*\*\*\*more examples of the IF design\*\*\*\*>

# The properties of the distribution of the IF peaks

The peaks {k} of the IF are defined by the equation:

where is the IF, are the positions of the peaks and is the desired minimum distance between the adjacent peaks.

The peaks may indicate not only the QRS complexes, but also the P or T waves or the bursts of noise, so the detector have to distinguish the first ones from the rest. The design of the IF assumes that this should be possible using the peak heights only, which is illustrated by the fig.3.

|  |
| --- |
| *Figure 3, peaks of the IF* |
| *C:\Users\Windows 10\AppData\Local\Microsoft\Windows\INetCache\Content.Word\IF_Peaks_mitdb-105_500_504.png* |
| The IF peaks are marked by the green asterisks. QRS peaks are much higher. |

From the fig.4 it is evident that the distribution of the IF peaks in the normal ECG is bimodal and consists of the two distinct clusters. The left cluster corresponds to the P, T waves and the noise, while the right one to the QRS peaks. So the extraction of the QRS peaks may be accomplished by some sort of cluster analysis.

|  |  |
| --- | --- |
| *Figure 4, the distribution of the IF peaks in the normal ECG* | |
| *IF_Peaks_mitdb-105_500_530* | *Hist_mitdb-105_500_530* |
| Left – the IF with the peak marks, Right – the histogram of the peak values | |

In the presence of noise the clusters become wider, up to the point where they finally merge together while the distribution becomes unimodal, which is shown in the fig.5. In this case the classification of peaks by their values becomes impossible.

|  |  |
| --- | --- |
| *Figure 5, the distribution of the IF peaks in the noisy ECG* | |
| *IF_Peaks_mitdb-105_1204_1234* | *Hist_mitdb-105_1204_1234* |
| Left – the IF with the peak marks, Right – the histogram of the peak values | |

# Classification via cluster analysis

The idea of cluster analysis is to partition the set of *N* data points into clusters represented by prototype points and assignment matrix:

The quality of cluster mapping is measured by the squared error:

|  |  |
| --- | --- |
|  | (1) |

Given the and the desired number of clusters, the task of the analysis is to find the and that minimize the expression (1).

The condition of local optimality of the solution results in the following properties of and :

|  |  |  |
| --- | --- | --- |
|  |  | (2) |
|  |  | (3) |

That is,

* each cluster’s prototype point is an arithmetic average of the cluster’s data points,
* each data point belong to the cluster with the closest prototype point.

From the equations (2) and (3) directly may be derived the iterative algorithm for finding the local optimum of (1). Considering the and on the left side of the equations (2),(3) as the unknowns, and the ones on the right side as the result of previous iteration, we come to the algorithm known as *k-means*, or *Lloid’s algorithm* [links]:

|  |  |  |  |
| --- | --- | --- | --- |
| 1. initialize the prototype points: |  | |  |
| 1. find the assignment matrix: |  | repeat steps 2 and 3 until convergence | (4) |
| 1. update the prototype points: | (5) |

The random values in the step 1 should be taken from the expected domain of .

The serious deficiency of the *k-means* is that the cluster-separating planes are equidistant from the cluster centers. This may result in misclassification and skewing the central points, as shown in fig.6.

|  |
| --- |
| *Figure 6, wrong classification using k-means* |
|  |
| The right cluster is much wider. Some of its points were wrongly assigned to the left cluster; the centers are skewed. |

This problem may be circumvented if one knows in advance the expected relative cluster widths, by weighting the distances in the formula (4):

|  |  |  |
| --- | --- | --- |
|  |  | (4.1) |

where are the expected relative widths. Fig.7 shows the outcome of such improvement.

|  |
| --- |
| *Figure 7, improved k-means classification using weighted distances* |
|  |
| The improved classification using formula 4.1 and expected relative widths 1:4 |

## The online k-means

By the word “online” we denote here the class of processes that convert a sequence of samples {} to another sequence {}, provided that:

* The process is causal, that is, the depends only on the and
* Each sample is processed in constant time and memory

With small modifications to the expressions for the regular *k-means* we can derive the online version of this algorithm. This is based on the assumption that each sample in the sequence is generated by the same statistical model with constant or slowly changing parameters. The task is to track these parameters using an online process. The algorithm is based on two ideas. The first is to perform the iteration for each input sample. The second is to perform the computation in steps 2 and 3 over a fixed number of the last points of the data sequence. Thus we will obtain a running estimate of the instant cluster parameters.

The input to the algorithm is a sequence of samples, one at a time. For each input sample the algorithm will output an instant estimate of cluster centers and an assignment vector :

The parameters to the algorithm are the number of clusters *K*, the size of processing window L and optionally the expected cluster widths .

First, let’s consider the (4.1) and (5). Noting that iteration now is performed for each input sample, and that each iteration creates a new assignment vector instead of updating the whole assignment matrix, we make the following changes to the equations:

* the indices *i* and *n* are now the same, so becomes ,
* the summation now is performed over the last L points

This results in the following algorithm:

|  |  |  |  |
| --- | --- | --- | --- |
| 1. initialize the prototype points: |  | |  |
| 1. for input sample compute the assignment vector: |  | repeat steps 2 and 3 for each following sample | (6) |
| 1. compute the updated prototype points: | (7) |

The step 3 may be further simplified. Noting that the in the (7) belong to {0, 1}, the expression for may be considered as a running average of varying length over the samples that belong to the cluster . But this average may be approximated by a lowpass IIR filter of less computational complexity. Using the following averaging IIR filter:

we obtain the following expression for the step 3:

|  |  |  |
| --- | --- | --- |
|  |  | (7.1) |

where is a learning rate.

## Connection to the Pan-Tompkins algorithm

Let’s apply the above algorithm to classifying the peaks of the QRS indicator function. As noted in section 4, the expected number of clusters is 2, where corresponds to the lower (non-QRS) peaks and corresponds to the higher (QRS) peaks. Using and we obtain the following algorithm:

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Initialization. Assign to the and some meaningful values derived from the first seconds of the IF, e.g.  10-th percentile of the   95-th percentile of the | | |  |
|  |  | repeat steps 2 and 3 for the next peaks | (8) |
| 1. If PEAK >  else | (9) |

Expressions (8) and (9) exactly match the expressions (12)-(14) and (17)-(18) in [link to PT work], so we conclude that the PT’s peak classification algorithm is based on cluster analysis in the form of online *k-means* algorithm. Fig.8 shows the sample output of this algorithm.

|  |  |
| --- | --- |
| *Figure 8, output of the online k-means* | |
| C:\Users\Windows 10\AppData\Local\Microsoft\Windows\INetCache\Content.Word\okm_mitdb-105_505_530.png |  |
| left – normal ECG, right – noisy one | |

# Classification via continuous distribution modeling

In statistical modeling a set of data samples is considered to be generated by a stochastic process with a certain probability density function (PDF).  
The common task of statistical modeling is, given a PDF with unknown parameters and a set of samples, to find the parameters that maximize the likelihood of to be produced by the . This is a classic optimization task where the measure of optimality is commonly represented by a log-likelihood:

|  |  |  |
| --- | --- | --- |
|  |  | (10) |
|  |  | (11) |

For some distributions the optimal parameters may be found analytically, e.g. for Gaussian distribution:

|  |  |  |
| --- | --- | --- |
|  |  | (12) |
|  |  | (13) |
|  |  | (14) |

Statistical modeling may be applied to cluster analysis. In this approach each cluster is modeled by unimodal distribution which is easy to optimize, e.g. the Gaussian one, and the whole distribution is modeled by a weighted sum of the *K* per-cluster distributions:

|  |  |  |
| --- | --- | --- |
|  |  | (15) |

where are the cluster weights, are the per-cluster distributions, are the per-cluster parameter vectors, and the following constraint is met:

|  |  |  |
| --- | --- | --- |
|  |  | (16) |

The task is to find that maximize the log-likelihood (10) of the PDF (15) over a sample set S under condition (16). This problem cannot be solved analytically, but a simple numerical solution similar to the *k-means* algorithm may be derived. Let’s write the equations for the constrained local maximum of:

|  |  |
| --- | --- |
|  | (17) |
|  | (18) |

where is the Lagrangian multiplier. By summing up the equations in (17) and using (15) and (16) we obtain and therefore

|  |  |  |
| --- | --- | --- |
|  |  | (17.1) |

We need to solve the system of equations (17.1) and (18) for and . The analytical solution is hardly possible, but these equations can be converted into the formulas for the iterative solution. Interpreting the on the left side of (17.1) and the under the logarithm in the (18) as the unknown values, and the rest occurrences of and as a results of previous iteration, we obtain the following algorithm:

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Initialize the parameters: |  | |  |
| 1. Find the updated cluster weights : |  | repeat steps 2 and 3 until convergence | (19) |
| 1. Update cluster parameters by solving for : | (20) |

The equation (20) can be considered an independent optimization of each cluster’s distribution over a weighted sample set. Thus the problem of fitting the composite distribution is reduced to a set of more tractable problems. With the proper choice of the equations in (20) may be solved analytically, giving the final expressions for the computation. The resulting algorithm is known as Expectation Maximization [link].

The matrix in (19) describes the expectancy of a sample to be produced by a cluster and is a complete analog of the assignment matrix in Section 5, the difference being that the assignment here is probabilistic rather than deterministic. The EM and *k-means* also share the general structure: in each iteration we first update the assignment matrix and then use it to update the per-cluster parameters. For this reason the regular EM algorithm is sometimes called a “soft EM” while the *k-means* is called “hard EM”. More on the connection between the two may be found in [links].

In the following sections we will consider the distributions produced by the one-dimensional Gaussian Mix Model (GMM) – a weighted sum of normal distributions:

|  |  |  |
| --- | --- | --- |
|  |  | (21) |

Let’s write down the explicit expressions for the EM for this model. In the expression (19) we just substitute by, and the solution of the equation (20) is given by (13) and (14) with added weights.

So the final algorithm is:

|  |  |  |  |
| --- | --- | --- | --- |
| 1. initialize the parameters: |  | |  |
| 1. update the assignment matrix: |  | repeat steps 2, 3 until convergence | (22) |
| 1. update the parameters: | (23) |
|  | (24) |
|  | (25) |

# Online EM algorithm

Using the same reasoning as in the section 5.1 we can derive the online version of the EM algorithm. We will assume that each sample in the sequence is generated by a Gaussian Mix Model with slowly changing parameters and our task is to track these parameters using an online process. The input to the algorithm will be an infinite sequence of samples, and the output will be a sequence of instant estimations of GMM parameters , and , .

Let’s consider what would happen if, instead of using the same sample set in all iterations, we would perform each iteration over a different sample set produced by the same random process with distribution *G*. Our assumption is that, as all the are produced by the same process, they have the same statistical properties, so the weighted statistical moments of appearing in (23)-(25) will differ only by negligible amounts from the true moments of *G*, so the process will eventually converge to a close vicinity of the true solution.

We are going to perform an iteration for each new sample in the input sequence and to use the last *L* samples of the sequence as an input to the iteration:

|  |  |
| --- | --- |
|  | (26) |

Then, we can significantly reduce the computations with the following change in the treatment of the assignment matrix. Instead of computing the whole matrix in each iteration we are going to compute only one assignment vector for the new sample . In the formulas (23)-(25) we will use a set of the last *L* assignment vectors:

Such change is justified by the following consideration. Suppose that we use the input sequences (26) and a full assignment matrix, and the iterations came close to a convergence point. Then the parameters , and will be approximately the same in each iteration, therefore the columns of the assignment matrix will depend only on the corresponding input sample:

But, due to (26), the *I*-th element of the is the same as the *(i-1)*-th element of the , so the *I*-th column of the assignment matrix in the *n*-th iteration will be approximately the same as the *(i-1)*-th column in the *(n-1)*-th iteration. So in each iteration we actually need to compute only the first column and can take the rest columns from the previous iterations. By construction, the convergence points for the modified algorithm will remain intact. The convergence rate may change, but we don’t expect this change to be essential.

As the index *i* in (23)-(25) is now a function of *n*, we may omit it completely and perform the summations by the upper index. So we come to the following algorithm:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. Initialize the parameters: |  |  | |  |
| 1. For each input sample compute the new assignment vector using distribution parameters found in previous iteration: |  |  | repeat steps 2, 3 for each input sample | (27) |
| 1. Find new parameters using *L* previous input samples and assignment vectors: |  | (28) |
|  |  | (29) |
|  |  | (30) |

The upper bar here means averaging over the last *L* points.

The expressions in (28)-(30) may be optimized by approximating the averages by a simple averaging IIR filter. In this case we should introduce the three internal variables for the weighted moments , and . Using the filter , where is learning rate, we finally obtain the following:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. Initialize the parameters and average moments: |  |  | |  |
| 1. For an input sample compute the new assignment vector: |  |  | repeat steps 2-4 for each input sample | (31) |
| 1. Update the average moments: |  | (32) |
| 1. Update and output the parameters: |  | (33) |
|  |  | (34) |
|  |  | (35) |

Finally, a few notes should be made on the choice of the initial parameters. Though the algorithm above suggests using random values for initialization, this should be avoided in actual implementations, because of the possible adverse effect on convergence. Here are a few heuristic rules that have proved to provide a reliable convergence in our experiments. The best option is when the true parameters of the model may somehow be guessed in advance, in which case they should be used for the initialization. If no such guess can be made, it is recommended to evenly distribute the initial clusters across the expected range of the signal and make the clusters to overlap by approximately 50%. If a part of the input sequence is known in advance and no guess can be made about the clusters, it’s better to evenly distribute the clusters over the percentiles of the known sample values. E.g. for a bimodal model we’d use the 25-th and 75-th percentile, for a trimodal one we’d use the 17-th, 50-th and 83-th percentiles, and so on. And if we totally miss any clues on the forthcoming data samples, we resort to random initialization.

## Testing the Online EM against artificial data

# QRS detection algorithm

# Results and discussion

# References

EMG frequence range, 25-450 Hz,

van Boxtel, A. (2001). Optimal signal bandwidth for the recording of surface EMG activity of facial, jaw, oral, and neck muscles. Psychophysiology 38, 22–34.

Viitasalo, J. H. T., and Komi, P. V. Signal characteristics of EMG during fatigue. European Journal of Applied Physiology 37, 111–121

De Luca, C. J. (1997). The use of surface electromyography in biomechanics. Journal of applied biomechanics

QRS morphology

Ary L. Goldberger, Zachary D. Goldberger, Alexei Shvilkin.

Goldberger’s Clinical Electrocardiography: A Simplified Approach, 8-th Edition, page 9

QRS power spectrum

Estimation of QRS Complex Power Spectra for Design of a QRS Filter  
Nitish V. Thakor, John G. Webster, Willis J. Tompkins

IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, VOL. BME-31, NO.11, NOVEMBER 1984

EM vs k-means

<https://ttic.uchicago.edu/~dmcallester/ttic101-07/lectures/em/em.pdf>  
 <https://cogcomp.seas.upenn.edu/papers/SamdaniChRo12.pdf>