HHS Public Access

Author manuscript

Comput Cardiol (2010). Author manuscript; available in PMC 2016 June 16.

Published in final edited form as:

Comput Cardiol (2010). 2015 September ; 2015: 273–276. doi:10.1109/CIC.2015.7408639.

The PhysioNet/Computing in Cardiology Challenge 2015: Reducing False Arrhythmia Alarms in the ICU

Gari D Clifford[1,2], **Ikaro Silva**[3], **Benjamin Moody**[3], **Qiao Li**[1], **Danesh Kella**[4], **Abdullah Shahin**[5], **Tristan Kooistra**[5], **Diane Perry**[5], and **Roger G. Mark**[3]

[1]Department of Biomedical Informatics, Emory University, Atlanta, GA USA

[2]Department of Biomedical Engineering, Georgia Institute of Technology, Atlanta, GA, USA

[3]Institute for Medical Engineering & Science, Massachusetts Institute of Technology, USA

[4]Emory School of Medicine, Emory Univesrity, Atlanta, GA

[5]Beth Israel Medical Center, Harvard University, Boston MA, USA

Abstract

High false alarm rates in the ICU decrease quality of care by slowing staff response times while increasing patient delirium through noise pollution. The 2015 Physio-Net/Computing in Cardiology Challenge provides a set of 1,250 multi-parameter ICU data segments associated with critical arrhythmia alarms, and challenges the general research community to address the issue of false alarm suppression using all available signals. Each data segment was 5 minutes long (for real time analysis), ending at the time of the alarm. For retrospective analysis, we provided a further 30 seconds of data after the alarm was triggered.

A collection of 750 data segments was made available for training and a set of 500 was held back for testing. Each alarm was reviewed by expert annotators, at least two of whom agreed that the alarm was either true or false. Challenge participants were invited to submit a complete, working algorithm to distinguish true from false alarms, and received a score based on their program's performance on the hidden test set. This score was based on the percentage of alarms correct, but with a penalty that weights the suppression of true alarms five times more heavily than acceptance of false alarms.

We provided three example entries based on well-known, open source signal processing algorithms, to serve as a basis for comparison and as a starting point for participants to develop their own code. A total of 38 teams submitted a total of 215 entries in this year's Challenge.

1. Introduction

In the 2015 PhysioNet/Computing in Cardiology Challenge, we aim to address the problem of high false rrhythmia alarm rates by encouraging the development of new algorithms to improve the specificity of ICU alarms. In this Challenge, we have focused on five types of life-threatening arrhythmia events, which we have defined as follows:

Asystole (ASY): There are no heartbeats at all for a period of four seconds or more.

Extreme bradycardia (EBR): The patient's heart rate is lower than 40 beats per minute; fewer than five beats occur within a period of six seconds.

Extreme tachycardia (ETC): The heart rate is higher than 140 beats per minute; more than 17 beats occur within a period of 6.85 seconds.

Ventricular tachycardia (VTA): There are five or more consecutive ventricular beats within a period of 2.4 seconds (a rate of 100 per minute.)

Ventricular fibrillation or flutter (VFB): The heart exhibits a rapid fibrillatory, flutter, or oscillatory waveform for at least four seconds.

Participants in the Challenge were given samples of ICU patient waveforms that were identified by the bedside monitor as falling into one of the above categories, and were tasked with devising an algorithm to determine which of these alarms represented true arrhythmias, and which were caused by other factors (such as noise, patient movement, leads falling off, or mis-identification of ECG features on the part of the monitor.)

The Challenge was divided into two events. Event 1 was a simulation of the *real-time* alarm suppression problem: the algorithm needed to determine whether the alarm was true or false based solely on the information available before the alarm was first triggered. In Event 2, algorithms were also able to see 30 seconds' worth of waveform data following the time of the alarm, and could use this information to *retrospectively* classify the alarm as true or false. The development of an algorithm that could reliably solve either of these problems would be a major step forward in patient care.

2. Example algorithms

Key to rhythm detection is accurate heart rate estimation. Several ECG R-peak detection algorithms are freely available, several of which were used in the Challenge example entries.

eplimited (available at www.eplimited.com) [1], which used digital filtering and a group of decision rules.

sqrs (available on PhysioNet [2]) [3], which uses a single scan of the sampled data and combines digital filter preprocessing with a detector and feature extractor based on dynamically adjusted slope and timing criteria.

wqrs (available on PhysioNet) [4], which is based on the length transform.

gqrs (available on PhysioNet), which consists of a QRS matched filter with a custom built set of heuristics (such as search back).

coqrs [5–7] based on the peak energy (no search back).

jqrs [8] consists of a window-based peak energy detector but with replacement of the original band-pass filter with a QRS matched filter (Mexican hat) and an additional heuristic ensuring no detections were made during flat lines.

Detection of the onset of the pulses in the ABP and PPG signals can provide further information on rhythm and rate. An open-source algorithm, *wabp* [9], is available from PhysioNet. The algorithm consists of three components: 1) a low-pass filter which is to

suppress high frequency noise that might affect the onset detection; 2) a windowed and weighted slope sum function (SSF) which is to enhance the upslope of the pulse and to suppress the remainder of the pressure wave; 3) a decision rule which allows for detection of each SSF pulse onset.

We provided three example Challenge entries, based on these and other open-source algorithms, and implemented in various programming languages, to serve as a basis on which participants could develop their own code.

The simplest example entry (#1) used *wabp* and *gqrs*, along with the *gqfuse* tool (available on PhysioNet), to analyze all available signals and select the most stable sequence of RR intervals, in order to detect asystole, bradycardia, and tachycardia. To detect the onset of VF, this entry analyzed the ECG and pulsatile signals separately (using *gqfuse* for each), and searched for a 10-second interval where the QRS rate and pulse rate were equal, followed by a 3-second interval in which the QRS rate increased by at least 25% and the pulse rate decreased by at least 75%. This entry did not attempt to detect VT.

An example entry (#2) written in Matlab used *wabp* to detect the beats and used *jSQI* [10] and a template matching SQI [11] to estimate the signal quality from ABP and PPG channels. jSQI flags a signal as bad quality if derived parameters from a blood pressure wave are not in reasonable physiological ranges. The PPG signal quality [11] matches a running PPG template with the pulsatile beat by dynamic time warping, simple matching, linear resampling matching and a clipping detection. When the signal quality was equal or greater than 0.9 and the corresponding HR or beat-to-beat interval derived from either the ABP or PPG did not surpass a predefined HR threshold (4s for ASY, 40 bpm for EBR, 140 bpm for ETC, 100 bpm for VTA and 250 bpm for VFB), the alarm was suppressed (i.e. false).

Finally, the last sample entry (#3) was provided for Octave users, with functions from by WFDB Toolbox for Octave/MATLAB [12]. This sample entry ran three QRS detectors from the WFDB Toolbox: *wqrs* on signal 1, *sqrs* on signals 1 and 2, and *gqrs* on signals 1 and 2. The results of the QRS detectors where then used to compute three tachograms. A decision was made on the truthfulness of the alarm based on the average pair-wise correlation between the tachograms 30 seconds prior to the alarm (a threshold was set arbitrarily based on the training data).

It should be noted that no ECG signal quality was used, although previous studies using the agreement of beat detectors have shown great promise in this area [13]. We also note that no ECG-based rhythm detection was used, although various open source algorithms have been made available to the Challenge participants [14].

We also implemented two voting algorithms. One used a simple unweighted voting of all the competition entrants' final scores. The second used the N best performing final entries ranked by their score on the training data. A tied, absent or no vote was treated as 'true'.

3. Challenge data

Data for the Challenge consisted of waveform recordings from ICU patients in four hospitals in the USA and Europe, representing three major manufacturers of ICU monitoring equipment. For each arrhythmia alarm matching our selection criteria, we collected all available multi-parameter waveforms (including at least five minutes of data before and after each alarm), as well as the alarm messages themselves, and any other status messages reported by the monitor. If possible, we also collected a list of the fiducial points and types of beats that were detected by the monitor; in some cases, the monitor did not provide this information. All of the signals were filtered in order to remove spectral characteristics that might identify the manufacturer or the country of origin. They were then resampled to 250 Hz and scaled to a 16-bit range. The specific names of the various alarm annotations were also normalized to anonymize the data.

3.1. Expert labeling

To build the “gold standard” list of true and false alarms, a team of experts visually inspected the waveform record at the time of each alarm. Each annotator worked independently and was assigned a randomized list of patients to review. For each alarm, the annotator was initially shown 15 seconds of waveforms prior to the alarm and 5 seconds after it, but could resize and scroll the window in order to examine earlier and later portions of the record. If possible, the monitor-computed beat labels were also displayed.

After examining the alarm label and surrounding waveforms, the annotator was asked to press one of four buttons: *True*, *False*, *Reject*, or *Uncertain*. The *Reject* label was used for records that were clearly fallacious (usually due to bugs in the monitor’s data-exporting interface.) In order for an alarm to be included in the Challenge data set, it had to be independently reviewed by at least two annotators of whom a two-thirds majority had to agree that the alarm was either *True* or *False*.

3.2. Training and test data

From the set of 1,564 alarms meeting all of the above criteria, we randomly picked 1,250 to serve as training and test data for the Challenge (see table 1). The distribution of alarms was chosen to reflect the distribution of alarm types in the original data set (17% ASY, 11% EBR, 17% ETC, 47% VTA, 7% VFB) as well as to maintain the approximate true-to-false ratio for each alarm type. No single patient appeared in both the training and test sets, and no single manufacturer or hospital made up more than half of the records in either set.

Up to four signals were selected from each record: two ECG leads (preferably one limb lead and one precordial lead), and up to two other signals, including ABP, PPG, or respiration. The public training set consisted of 375 “short” records, containing only the five minutes leading up to the alarm, and 375 “long” records, containing a further 30 seconds after the alarm. The hidden test set consisted of 250 “short” records (used only for Event 1) and 250 “long” records (used for both events.) Each record was labeled with the alarm type, and in the case of the training set, veracity (true or false). The records did not include the monitor-computed beat locations or heart rate.

4. Scoring

Participants were asked to submit their entries in the form of a ‘zip’ or ‘tar’ archive that included everything needed to compile and run their program on a GNU/Linux system, together with the results that they expected their program to produce for the records in the public training set. When an entry was uploaded, the scoring system would first attempt to compile the program and run it over a randomly selected subset of the training set; if this did not produce the expected results, evaluation stopped and the error messages were sent back to the submitter.

Once the program was successfully compiled and validated, it was then invoked for each record in the test set. (For the 250 “long” records, the program was invoked twice: once with the full record as input, and once with a truncated version.) If the program failed to produce output for a given record, it was treated as if it had classified that alarm as true.

For each category, the entry’s score was computed based on the number of *true positives* (true alarms classified as true), *false positives* (false alarms classified as true), *true negatives*, and *false negatives*. The scoring function was designed to treat false negatives – genuinely life-threatening events that the program considered unimportant – especially harshly, and was defined as:

$$score = \frac{100 \cdot (TP + TN)}{TP + TN + FP + 5 \cdot FN}$$

5. Results, Discussions & Conclusions

A totla of 29 closed-source entries and 215 open-source entries were submitted in the Challenge. Table 2 provides a breakdown of the top scoring entries. A different contestant ranked highest in each separate alarm category, indicating that there was no best general algorithm. Interestingly, a simple majority vote of all the 38 competitors’ final entries gave scores of 60.15 in the real-time event and 62.41 in the retrospective event. These moderate performances, well below the top 10 algorithms, indicating that simple voting schemes do no yield an improved performance in this context, since the performance tail is long. A voting algorithm using the N=13 best performing final entries ranked by their score on the training data, provided the highest scores in both event 1 (84.26) and event 2 (87.04), although N=11 was sufficient to beat the best performance in either event. A weighted voting scheme may well improve these scores. We note, however, that N was selected on the test data, so these results should not be considered truly out of sample.

For the top performing entrants, it was the VTA alarm that proved the hardest to classify accurately. We note that retrospective scores were generally higher than ‘real-time’ scores, with the highest performing retrospective approach only suppressing 1% of the true alarms, while 80% of the false alarms were suppressed. Although debatable, this may be acceptable as a clinical algorithm if a 30 second window were acceptable. We suggest that this may spur a re-consideration of the AAMI guidelines for maximum alarm latency. A special issue in the journal *Physiological Measurement* will follow this competition and provide a forum

for an extended editorial which will discuss the methods and results in more detail, and provide the opportunity for entrants to revise their algorithms.

Acknowledgments

This work was funded in part by the National Institutes of Health, grant R01-GM104987. We also would like to thank Arvind Ananthan, Jaka Cikac, Barbara Drew, Quan Ding, Scott Eaton, Xiao Hu, Franc Jager, Cadathur 'Raj' Rajagopalan and Anzë Rezelj, GE Healthcare, Math-Works, Mindray North America and Philips Healthcare.

References

1. Hamilton PS, Tompkins WJ. A real-time QRS detection algorithm. IEEE Trans Biomed Eng. 1986; 33(12):1157–1165. [PubMed: 3817849]
2. Goldberger AL, Amaral LA, Glass L, Hausdorff JM, Ivanov PC, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE. Physiobank, Physiotoolkit, and Physionet Components of a New Research Resource for Complex Physiologic Signals. Circ. 2000; 101(23):e215–e220.
3. Engelse WAH, Zeelenberg C. A single scan algorithm for QRS-detection and feature extraction. Comput in Cardiol. 1979; 6:37–42.
4. Zong W, Moody GB, Jiang D. A robust open-source algorithm to detect onset and duration of QRS complexes. Comput in Cardiol. 2003; 30:737–740.
5. Clifford, GD. PhD thesis. University of Oxford; 2002. Signal Processing Methods for Heart Rate Variability.
6. Nygårds ME, Sörnmo L. Delineation of the QRS Complex using the Envelope of the ECG. Med and Biol Engin and Comput. 1983; 21(5):538–547.
7. Oster J, Behar J, Colloca R, Li Q, Li Q, Clifford GD. Open source Java-based ECG Analysis Software and Android App for Atrial Fibrillation Screening. In Comput in Cardiol. 2013:731–734.
8. Behar J, Oster J, Clifford GD. Combining and Benchmarking Methods of Foetal ECG Extraction without Maternal or Scalp Electrode Data. Phys Measur. 2014; 35(8):1569.
9. Zong W, Heldt T, Moody G, Mark R. An open-source algorithm to detect onset of arterial blood pressure pulses. In Comput in Cardiol. 2003:259–262.
10. Sun J, Reisner A, Mark R. A Signal Abnormality Index for Arterial Blood Pressure Waveforms. In Comput in Cardiol. 2006:13–16.
11. Li Q, Clifford GD. Dynamic time warping and machine learning for signal quality assessment of pulsatile signals. Phys Meas. 2012; 33(9):1491–1501.
12. Silva I, Moody G. An open-source toolbox for analysing and processing physionet databases in matlab and octave. Journal of Open Research Software. 2014; 2(1):e27. [PubMed: 26525081]
13. Behar J, Oster J, Li Q, Clifford GD. ECG Signal Quality during Arrhythmia and its Application to False Alarm Reduction. IEEE Trans on Biomed Engin. 2013; 60:1660–1666.
14. Li Q, Rajagopalan C, Clifford G. Ventricular fibrillation and tachycardia classification using a machine learning approach. IEEE Transactions on Biomedical Engineering. 2014; 61(6):1607–1613. [PubMed: 23899591]

Table 1

Types of alarms and signals used in the Challenge. Each of the N records included two ECG channels.

	Training (N=750)		Test (N=500)	
	False	**True**	**False**	**True**
ASY	100	20	90	12
EBR	45	45	38	26
ETC	8	131	5	68
VTA	253	90	176	45
VFB	52	6	34	6
PPG	227	178	158	83
ABP	59	63	58	39
Both	172	51	127	35
Total	458	292	343	157

Table 2

Final scores for the top 9 entrants in both events (real-time and retrospective) ranked by overall real-time score, the three example algorithms provided and a voting approach.

	Event 1 (Real-time)			Event 2 (Retrospective)		
Entrant	**TPR**	**TNR**	**Score**	**TPR**	**TNR**	**Score**
Plešsinger *et al.*	92%	88%	**81.39**	95%	88%	84.96
Kalidas & Tamil	**94%**	82%	79.44	94%	86%	81.85
Krasteva *et al.* *	93%	83%	79.41 *	93%	84%	79.56
Couto *et al.*	89%	**91**%	79.02	88%	**92%**	78.28
Fallet *et al.*	**94%**	77%	76.11	**99%**	80%	**85.04**
Hoog Antink & Leonhardt	93%	77%	75.55	90%	82%	75.18
Eerikäinen *et al.*	90%	82%	75.54	89%	85%	75.52
Ansari *et al.*	89%	84%	74.48	89%	87%	76.57
Liu *et al.*	89%	79%	71.68	93%	78%	75.91
Example Algorithm 1	76%	44%	41.41	73%	46%	40.83
Example Algorithm 2	86%	38%	45.07	84%	38%	44.37
Example Algorithm 3	64%	76%	45.59	61%	77%	47.35
Voting Algorithm (N=13)	94%	90%	84.26	94%	94 %	87.04

Best performances of competition entrants are in bold. TPR = fraction of true alarms correctly classified; TNR = fraction of false alarms correctly classified.

* denotes an unofficial (“closed-source”) entry.

Underlined scores are the highest unnoficial scores in the competition.