NOVATEUR PUBLICATIONS
INTERNATIONAL JOURNAL OF INNOVATIONS IN ENGINEERING RESEARCH AND TECHNOLOGY
[IJIERT] ISSN: 2394-3696 Website: ijiert.org
VOLUME 5, ISSUE 10, Oct. -2018

DYNAMIC THRESHOLDING CONSTANT COMPUTATION FOR DETECTION OF QRS-COMPLEXES IN ELECTRO-CARDIO GRAM SIGNALS USING GENETIC ALGORITHM AND VARIABLE WAVE APPROXIMATION

Alok Singh Gahlot
[1]Assistant Professor ,MBM Engineering College, J. N. Vyas University,
Jodhpur 342001, Rajasthan, India

ABSTRACT
Detecting QRS-complexes from ECG waveform accurately, can now be treated as a classical problem. Many of the traditional QRS-detection algorithms somehow uses an empirically derived, universal thresholding constant to separate out QRS-complexes from Non-QRS region, after some processing for different patient records. For instance the processing of an ECG signal may comprises of noise-filtering the raw signal, differentiating, squaring, running a moving window integral, normalizing and finally using a constant threshold for discrimination. The method of thresholding works best after normalization. Unfortunately, in cases, where a noise peak is of greater amplitude (than QRS-complexes) the normalization process scales-up the noise signal to unity and suppresses the QRS-complexes (may be, lower than the thresholding constant). This may even result in neglecting the actual QRS-complexes and counting the noise peak as the only QRS-complex in the signal. To avoid this problem if the thresholding constant is set to a lower level some T-waves may accidentally be detected as QRS-complexes.
In this work we have tried to apply principles of Genetic Algorithm (GA) in conjunction with a novel method similar to wavelet transform to compute a Dynamic Thresholding Constant (DTC). The method is named as Variable Wave Approximation (VWA). The performance of this algorithm is evaluated against the standard CSE ECG database. The results indicated that the algorithm achieved 99.xx% of the identification rate. The percentage of false positive and false negative is x.xx% and x.xx%, respectively.
The proposed method (VMA) along with GA can be used for various other signal processing applications also.

Keywords: ECG, QRS complex, Genetic Algorithm, Variable Wave approximation.

INTRODUCTION
Typical QRS-Detection Algorithm
To detect QRS complexes from a noise-filtered ECG signal:
STEP 1: Calculate the slope of individual ECG lead and square each slope
STEP 2: Summation of above 12 signals is taken and normalized
STEP 3: A moving window integral is taken
STEP 4: Using a thresholding constant QRS and Non-QRS regions are separated.
Traditionally, the processing of an ECG signal may comprise of noise-filtering the raw signal, differentiating, squaring, running a moving window integral and normalizing.
At this point, the above signal can be drawn on a paper and a hyper-plane can be imagined, which separates QRS-complex from P-waves, T-waves and rest of the non-QRS region.
Since the base-line-wander is already removed from the signal, this plane can be considered horizontal.
Further, since the individual ECG-lead data is a 2-dimensional waveform the horizontal-hyper-plane can be treated as a straight line.

$y = mx + c$ ---------- (Eq. 1)

or simply

y = c (as it is parallel to x-axis) ---------- (Eq. 2)

If the signal at after Moving Window Integral (MWI) has an amplitude 'a', then the hyper plane will be definitely below 'a', as it has to divide the signal into QRS and Non-QRS regions.

c = a/k ---------- (Eq. 3)

(Where a > **k** > 1, is a constant discussed latter)

Even though the signal after MWI is normalized, 'a' is different for different patient records. Also there may be some noise peaks grater than ECG amplitude which greatly disturbs the normalization process (see Fig 1) and the thresholding misses the QRS –peaks.

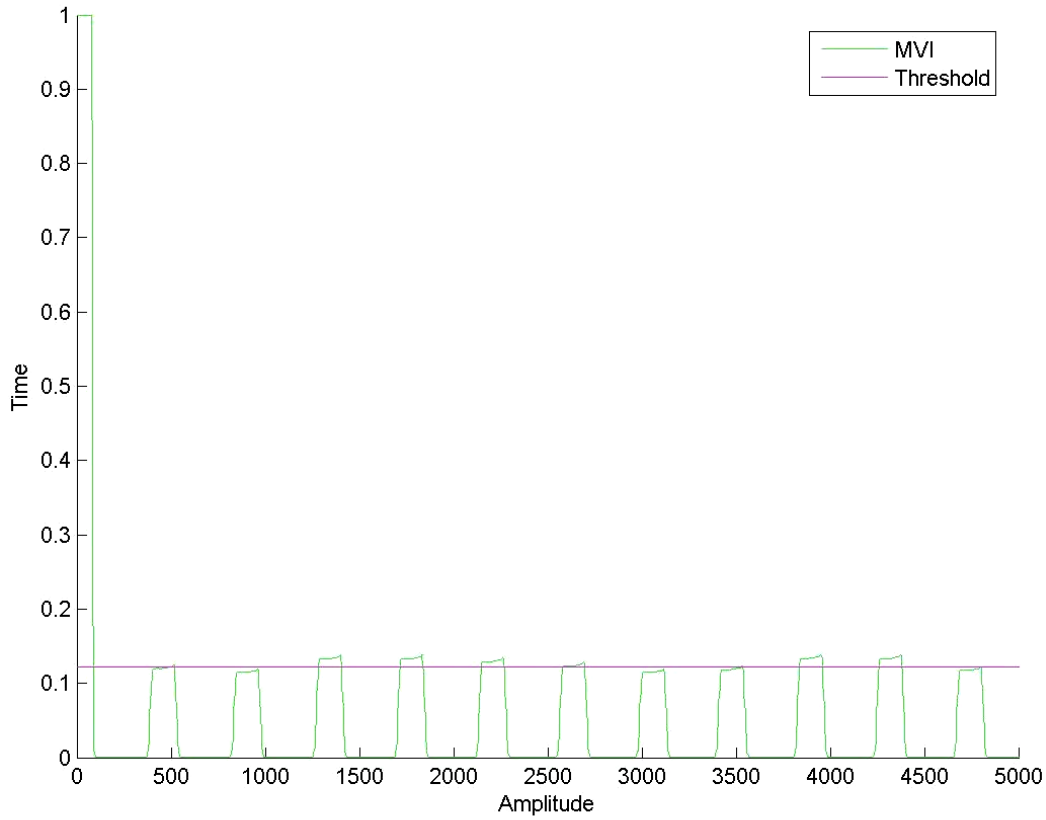

Figure: 1

To avoid this problem, if the thresholding constant is set to a lower level some small noise peaks associated with T-waves may accidentally be detected as QRS –complexes.

SHORTCOMMINGS OF PREVIOUSLY USED ALGORITHMS

If the signal at STEP 3 is plotted (with time on abscissa), the maximum y-value of the signal is always 1 (as it is normalized), but it has been observed that in many cases this is not the amplitude 'a' of the signal. Hence, we cannot assume a constant universal hyper plane, which will always divides the signal into QRS and Non-QRS regions, irrespective of noise. In other words, a universal thresholding constant may not accurately work for varied ECG-records. Finding correct thresholds empirically not always work accurately.

If a method can be found to correctly find the signal amplitude, we can find the height ('c') of the hyper plane (Eq. 3).
The proposed method calculates the correct amplitude of the signal despite of considerable amount of noise. The amplitude so found cannot be directly used as a hyper plane 'c'. As 'c' varies between 0 and 'a'. Hence the constant 'k' in Eq. 3, is used to reduce 'a', by some factor.

PROPOSED ALGORITHM

Terminology: Variable Wave Approximation method: Using this method a family of similar waveforms can be generated and compared with the signal under consideration. In this way the characteristics of the unknown waveform can be determined. For example after MWI, the ECG signal turns into a shape similar to a square wave. Now, for the VWA method the "Variable Wave" is a square wave. GA will return the parameters of the "Approximately" similar waveform.

To generate a square wave the following parameters are required:
1) T_on The period of time when the wave remains high
2) T_off The period of time when the wave remains low
3) Amplitude The amplitude of the square wave.
4) L-shift The amount of phase-shift required to synchronize it, with the signal under comparison (ECG after MWI)

The GA will now use VWA to generate a whole range of square waves and find the difference of the original signal with each square wave. The GA will finally return the parameters required to generate a square wave which is optimally similar to the ECG-MWI signal. Out of the above four parameters returned by GA, we use only one parameter ie amplitude. This is the amplitude of MVI-ECG signal irrespective of any noise peaks. Other parameters such as T_on and T_off can be used to find the periodicity of the ECG signal for determination of degree of arrhythmia. Currently we are only considering amplitude. This amplitude varies for each patient-record. We divide this amplitude by a factor 'k' and use the resultant value as the thresholding value as per Eq. 3.

Proposed Algorithm (How GA uses VWA to determine the optimal match for MVI-ECG)

STEP 1: For the 4-unknown *individuals* a random *population* is generated. The four individuals t_on, t_off, amplitude and l_shift act as genome(s). A random population of 20 genes is generated for each genome.

STEP 2: GA applies the genomes to the fitness function and calculates the score for each. The fitness function (minimization) is:

$$d = f(t\_on, t\_off, amplitude, l\_shift) = \sum | vsw - ecgmvi |$$

Where:
vsw: is a vector containing samples of the "Variable Square Wave" generated, using a set of genomes from the current population.
emvi: is a vector containing samples of the "ECG Moving Window Integral"
d : is the sum of, absolute values of, individual samples of the two vectors.

STEP 3: New generation is evolved using:
Elite Children: the individuals which produced
minimum 'd' are guaranteed to survive to the next generation.
Crossover children: are created by combining the vectors of a pair of parents.
Mutation children: are created by introducing random changes, or mutations, to a single parent.

STEP 4: Termination Condition
The above two steps are repeated until any one condition fails:
a) Number of maximum generations defined exceeds,
b) Number of 'stall generations' exceeds or
c) Maximum time allowed to execute the algorithm exceeds.

STEP 5: At this point the GA has obtained the parameters required to generate a square wave, which best 'approximates' the ECG_MVI. The amplitude so obtained is now divided by a constant 'k' to obtain the Dynamic Thresholding Constant (DTC).

STEP 6: Using the above computed DTC, the QRS and Non-QRS regions are separated.
A hyper plane or a thresholding line (y=c) is used to classify QRS and
Non-QRS regions.

The performance of the algorithm is evaluated against the standard CSE ECG database. The results indicated that the algorithm achieved 99.xx% of the identification rate. The percentage of false positive and false negative is x.xx% and x.xx%, respectively.

Genetic Algorithm parameter list

1. Population Type: It is taken as 'doubleVector', individuals in the population have type 'double'. The other option provided in Matlab is *Bit string* ('bitstring'). This option is suitable if the individuals in the population are bit strings mainly seen in problems which involves sequence matching. In our case the desired output of GA are decimal numbers so the *Population Type* is taken as 'doubleVector',

2. Population Size: Specifies how many individuals are there, in each generation. With a large population size, the genetic algorithm searches the solution space more thoroughly but increases the execution time. So it has to be chosen carefully, in our case we have set it to 20. In our case the final output(s) of the GA are all decimal numbers between 0 and 1. Taking 20 numbers in between 0 and 1 as the initial population was found sufficient.

3. Elite Count: Specifies the number of individuals that are guaranteed to survive to the next generation. The Elite count can be any integer number less than or equal to the population size. We have taken population size as 20 and elite count as 2. Taking too many elite children may not help in getting a varied population in the next generation.

4. Crossover Fraction: Is a field, in the GA's reproduction options. It specifies the fraction of each population, other than elite children, that are made up of crossover children. A crossover fraction of 1 means that all children other than elite individuals are crossover children, while a crossover fraction of 0

means that all children are mutation children. It has been observed that neither of these extremes is an effective strategy for optimizing a function. In our case it is chosen as 0.8. Since the population size, we have taken is 20 and elite children are 2, the remaining 18 are divided using Crossover Fraction. Hence 15 children are determined using *crossover* and 3 are determined using *mutation*.

5. Generations: At each iteration, the genetic algorithm performs a series of computations on the current population to produce a new population. Each successive population is called a new generation. It is a positive integer specifying the maximum number of iterations before the algorithm halts. If this number less, the output of the GA may not be desirable. In our case it was experimentally determined and assigned a value of 1200. This is used as one of the stopping criteria of the algorithm.

6. Time Limit: This also acts as one of the stopping criterion of the algorithm. It is a positive number, the algorithm stops after running for *TimeLimit* seconds. We have experimentally derived the value as 30. Since Generations also act as stopping criterion, GA algorithm terminates on 'which ever is first' basis. For example if number of generations is 896 only and it has already taken 30 second, then the algorithm will terminate. The experiments were conducted on an IBM® PC with Intel® Core™ Duo processor for which this time was sufficient for more than 1500 iterations.

7. StallGenLimit: The algorithm stops if there is no improvement in the best fitness value for the number of generations specified by Stall Generations Limit. We have chosen that the algorithm must run for at least 1200 generations or 30 seconds, which ever is first, irrespective of number of stall Generations, so this limit is taken as Infinity.

8. Initial Population: We have not defined a per-computed initial-population, instead the function is called to randomly calculate it. Initial population can be pre-defined using the *saved-results* of the previous run of the GA. Since the hyper plane for every record will be different, we have not used the previous population. It was observed during the experiments that allowing GA to use random values instead of using the previous values gave better results. To create the initial population the 'CreationFunction' (discussed below) was used.

9. CreationFcn: Matlab's inbuild function 'gacreationuniform' was used, it creates a random initial population with a uniform distribution.

10. FitnessScalingFcn: Fitness scaling converts the raw fitness scores that are returned by the fitness function, to values in a range, that is suitable for the selection function. The fitness scaling function based on 'Rank' (*fitscalingrank*) was used. It scales the raw scores based on the rank of each individual. The rank of an individual is its position in the sorted scores. The rank of the most fit individual is 1, the next most fit is 2, and so on. Rank fitness scaling removes the effect of the spread of the raw scores.

11. SelectionFcn: Selection options specify how the genetic algorithm chooses parents for the next generation. The Selection function used is *Stochastic uniform* ('selectionstochunif').

12. CrossoverFcn: Crossover options specify how the genetic algorithm combines two individuals, or parents, to form a crossover child for the next generation. we have used crossover function named

Scattered 'crossoverscattered'. It creates a random binary vector and selects the genes where the vector is a 1 from the first parent, and the genes where the vector is a 0 from the second parent, and combines the genes to form the child.

13. MutationFcn: Mutation options specify how the genetic algorithm makes small random changes in the individuals in the population to create mutation children. Mutation provides genetic diversity and enables the genetic algorithm to search a broader space. We have used Matlab's default mutation function, Gaussian (*mutationgaussian*), which adds a random number taken from a Gaussian distribution with mean 0 to each entry of the parent vector. The variance of this distribution is determined by the parameters Scale and Shrink and by the Initial range setting in the Population options.

REFERENCES

1. Arnold J.M. , "Time domain filters", MIT Quart Prog. Rep., vol. 106, pp. 184-194, 1972.
2. Arzeno, N. M., Poon, C. S., and Deng, Z. D., "Quantitative analysis of QRS detection algorithms based on first derivative of the ECG", Paper presented at 28th IEEE EMBS annual international conference, New York, USA, 1788-1791 2006
3. Chen, S. W., Chen H. C. and Chan, H. L., "A real time QRS detection method based on moving-averaging incorporating with wavelet denoising", Comp. Methods and Progs. in Biomed., 82, 187-195. 2006
4. Goldberg, David E., Genetic Algorithms in Search, Optimzation & Machine Learning, Addison-Wesley, 1989.
5. Goldschlager N. and Goldman M. J. "Electrocardiography: Essential of Interpretation" Lange Medical Publications, Maruzen Asian Edition, USA 1984.
6. Mehta S.S., Sexana S.C. and Verma H.K, "Computer-aided interpretation of ECG for diagnostics", International Journal of System Science, vol-27, pp.43-58, 1996.
7. Mehta S.S., Dave V., Vyas S.D. and Chouhan V.S., "Detection of QRS-complexes in 12-lead ECG using Error Back Propogation neural network", Int. Cong. on Bio. and Med. Engg., Singapore. Dec 4-5, 2002
8. Mehta S.S., Dave V., Kumar A.and Gupta S., "An entropic method for detection of QRS Complexes in 12 lead ECG using ANN", World Congress in Medical Physics and Biomedical Engineering , Australia. 24-29 Aug 2003
9. The MathWorks, Inc. (MATLAB® Genetic Algorithm Toolbox).