Moduł HeartClass

Michał Ciszewski, Łukasz Dudek, Krystian Mucha

Akademia Górniczo - Hutnicza w Krakowie

Na przedmiot: Elektroniczne Systemy Diagnostyki Medycznej i Terapii

Spis treści

I Wstęp **1**

II Koncepcja proponowanego rozwiązania **3**
I Ekstrakcja cech 3
II Klasteryzacja 3
III Klasyfikacja 6

III Rezultaty i wnioski **7**

IV Podsumowanie **7**

V Bibliografia **11**

Appendices **12**

A zagadnienia implementacyjne **12**

Streszczenie

Niniejszy artykuł dotyczy części programu do analizy sygnału elektrokardiograficznego, aby wykrywać w nim wszelkie niezgodności z normami. Ten moduł, nazwany "HeartClass", służy do analizy załamków QRS i grupowania ich według odpowiednio zdefiniowanego podobieństwa.

Słowa kluczowe: elektrokardiografia, klasyfikacja załamków QRS, algorytm k-średnich, algorytm G-średnich, metoda wektorów nośnych.

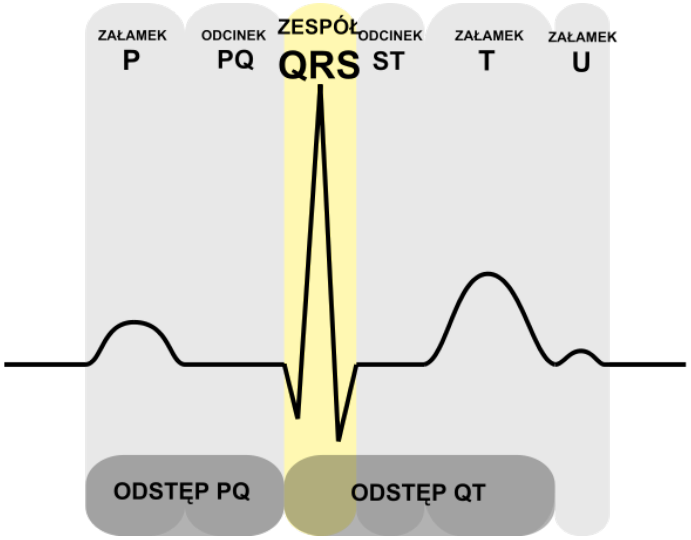
I. Wstęp

Elektrokardiogram jest jednym z najefektywniejszych narzędzi diagnostycznych do wykrywania chorób serca. EKG dostarcza prawie wszystkich informacji o aktywności elektrycznej serca. Typowy sygnał EKG składa się z załamka P, zespołu QRS oraz załamków T i U. Spośród wszystkich tych elementów sygału elektrokardiograficznego najbardziej charakterystycznym i zarazem najbardziej znaczącym jest zespół QRS. Na podstawie jego kształtu można zdiagnozować różne dysfunkcje serca, dlatego jego automatyczna klasyfikacja jest ważnym zagadnieniem.

Zespół QRS opisuje pobudzenie mięśni serca i składa się z jednego lub kilku załamków określanych jako Q, R i S.

1. Załamek R – każdy załamek dodatni w obrębie zespołu QRS.
2. Załamek Q – pierwszy ujemny załamek widoczny przed załamkiem R.
3. Załamek S – pierwszy ujemny załamek widoczny po załamku R.

Przykładowy (wyidealizowany) zespół QRS widoczny jest na rys. 1, przedstawiającym schematyczny fragment zapisu elektrokardiograficznego.



Rysunek 1: *Wyidealizowany schemat zapisu EKG z zaznaczonym zespołem QRS. Źródło [1]*

Klasyfikacja zespołu QRS ma na celu wyodrębnienie grup zespołów podobnych (w zadanym zakresie tolerancji). Odmienny kształt zespołu jest konsekwencją odmiennie przebiegającego pobudzenia. Klasyfikacja polega na stwierdzeniu przynależności klasyfikowanego zespołu do jednej z istniejących klas albo tworzenie nowych klas, jeżeli przynależności nie stwierdzono [2].

Celem opisywanego modułu jest wyliczenie liczby klas zespołów QRS, określenie reprezentantów każdej z nich oraz oznaczenie klas zespołów QRS na wykresie EKG. Wyodrębnienie klas QRS występujących w sygnale EKG pozwala na określenie prawidłowości rytmu pracy serca. Z reguły nieregularności mają charakter przejściowy, dlatego ich poprawne wyznaczenie wymaga przeprowadzenia 24-godzinnego badania pracy serca, czyli testu Holtera [3].

Rozwiązanie przyjęte w niniejszej pracy opiera się o ekstrakcję cech z sygnału EKG, klasteryzacji otrzymanych danych przy pomocy algorytmu G-średnich oraz klasyfikacji z wykorzystaniem maszyny wektorów nośnych. Dobór tych metod jest zgodny z wyborem poprzedniego zespołu projektowego, jako że celem niniejszej pracy jest porównanie działania jednego algorytmu realizowanego w różnych językach programowania. Założono, że dane wejściowe dostarczone przez poprzednie moduły są poprawne.

W literaturze można spotkać się z różnymi podejściami do klasyfikacji zespołów QRS. W [4] autorzy zaproponowali algorytm polegający na wykorzystaniu liniowej analizy dyskryminacyjnej (LDA) w celu zredukowania wymiaru przestrzeni cech. Do klasyfikacji zastosowano maszynę wektorów nośnych (SVM). Ponadto w pracy tej podjęto próbę klasyfiacji przy pomocy MLP (ang. Multilayer Perceptrons) oraz klasyfikatora FIS (ang. Fuzzy Inference System). Najlepsze rezultaty autorzy otrzymali wykorzystując SVM. Podobne podejście zastosowano w [5], gdzie dodatkowo w celu ograniczenia zakłóceń oraz ekstrakcji cech wykorzystano transformację falkową. W [6]

autorzy stworzyli adaptacyjny algorytm, działający w czasie rzeczywistym, klasyfikujący zespoły QRS. Zaproponowane rozwiązanie bazuje na modelu funkcji Hermite'a. W innej pracy autorzy wyekstrahowali cztery konkretne cechy z sygnału EKG, aby następnie wykorzystać odległość Mahalanobisa jako kryterium klasyfikacji [7].

II. Koncepcja proponowanego rozwiązania

Algorytm klasyfikacji załamków QRS został podzielony na trzy części. Najpierw dane wejściowe zostają znormalizowane i skwantyzowane, następnie przeprowadzana jest procedura ekstrakcji cech. W drugiej części następuje klasteryzacja wektorów cech zespołów QRS. Polega to na grupowaniu tych danych w klasy, które mają najwięcej wspólnego - leżą najbliżej siebie w przestrzeni o wymiarze równym liczbie porównywanych cech (stosowana jest tutaj metryka Euklidesowa). Warto zaznaczyć, iż każdy współczynnik reprezentuje inną wielkość i z tego powodu wartość tolerancji jest dobierana dla każdego z nich indywidualnie. Do klasteryzacji wykorzystywany jest algorytm G-średnich (ang. "G-means"). W ostatnim kroku następuje klasyfikacja, czyli przyporządkowanie każdego zespołu QRS do jednej z klas. W tym celu wykorzystywana jest metoda wektorów nośnych (ang. Śupport Vector Machine", w skrócie SVM).

Dobór tych metod - w tym również wybór klasyfikacji na podstawie wektorów cech, a nie sygnałów - został dokonany przez poprzedni zespół projektowy, a zadaniem autorów niniejszego raportu było przeprowadzenie porównania funkcjonowania tych metod w trzech innych językach programowania niż oryginalny język implementacji.

I. Ekstrakcja cech

Zadaniem tej części zastosowanego algorytmu jest wyliczenie pewnych istotnych wskaźników charakteryzujących zespół QRS na podstawie znormalizowanych danych wejściowych. Bazując na implementacji poprzedniego zespołu projektowego, stosowane są współczynniki wymienione poniżej [3].

1. Początek i koniec całego zespołu QRS,
2. Wartość szczytowa załamka R,
3. Interwał między poprzednim a rozważanym załamkiem R,
4. Interwał między rozważanym a kolejnym załamkiem R,
5. Wartość szczytowa oraz koniec załamka T,
6. Początek, wartość szczytowa i koniec załamka P.

Wszystkie te współczynniki powinny być zapisywane po synchronizacji wszystkich wykrytych zespołów, na przykład względem pozycji załamka R [2].

II. Klasteryzacja

Jak już zostało wspomniane, do grupowania danych w klastry (klasy) użyto algorytmu G-średnich. Jest rozszerzeniem popularnej metody k-średnich, która polega na dobraniu k klas w zbiorze danych tak, aby każdy punkt należał do klasy, do której środka ciężkości ma najbliżej [8].

Przyjęto, że dane, które należy pogrupować to d-wymiarowe wektory należące do zbioru X o liczności n. S to zbiór klas, a więc $S_j = \{x_i \in X | klasa(x_i) = j\}$ dla $i = 1, 2, ..., n$. Zbiór środków ciężkości klas oznaczony został literą C i zdefiniowany jako: $C = \{c_j = \frac{\sum_{x \in S_j} x}{|S_j|}\}$ dla $j = 1, 2, ..., k$. Cel algorytmu k-średnich to minimalizacja wyrażenia przedstawionego wzorem 1 .

$$\sum_{j=1}^{k} \sum_{x \in S_j} \|x - c_j\| \tag{1}$$

Poważnym problemem tego algorytmu jest fakt, iż liczba klas musi być znana bądź przyjęta z góry, co oznacza posiadanie pewnej wcześniejszej wiedzy na temat klasteryzowanego zbioru danych [9, 10]. W przypadku braku takich informacji, należy zastosować uogólnienie algorytmu k-średnich, które pozwoli dobrać optymalne k względem pewnego wskaźnika jakości. Algorytm, który został zastosowany w opisywanym module dobiera k tak, aby w każdej klasie rozkład punktów był możliwie bliski rozkładu normalnego. Stąd też wzięła się litera "G"w nazwie - od rozkładu Gaussa [9].

Metoda G-średnich zaczyna od niewielkiej liczby klas, by później odpowiednio zwiększać k - nie jest przewidziana procedura zmniejszania tego parametru. W pierwszym kroku zwykle przyjmowane jest $k = 1$, z czego wynika, że C jest zbiorem jednoelementowym, zawierającym środek ciężkości całego zbioru X [9]. W każdym kroku algorytm sprawdza, czy dana klasa ma rozkład normalny, a jeśli nie, to dodaje jej dodatkowy środek. Między każdym takim dodawaniem środków jest używana procedura k-średnich, aby poprawić jakość rozwiązania.

Sprawdzenie normalności rozkładu wewnątrz klasy odbywa się za pomocą testu Andersona - Darlinga. Pozwala on rozstrzygnąć, czy znormalizowane dane są rozłożone zgodnie z pewnym rozkładem prawdopodobieństwa. Elementy zbioru danych oznaczono przez $y_i, i = 1, 2, ..., n$. Wartość statystyki Andersona - Darlinga oblicza się na podstawie wzoru 2.

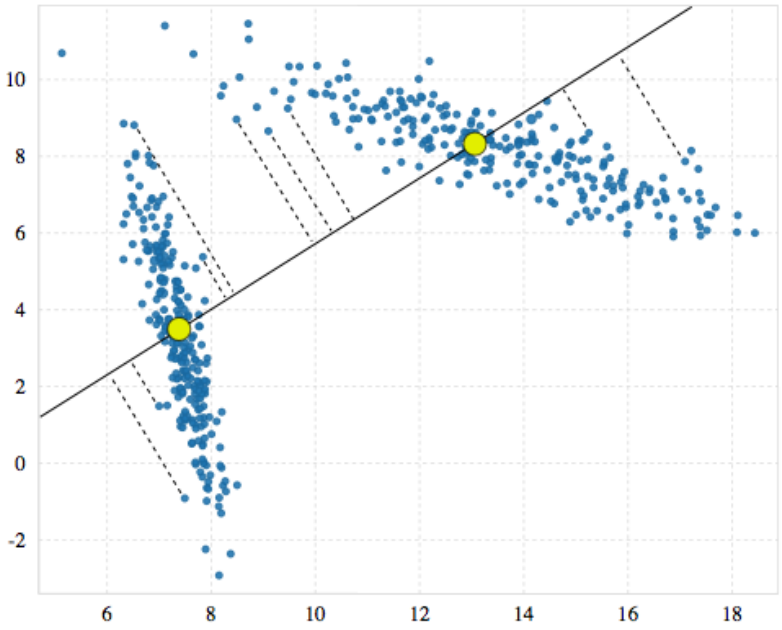
$$A^2 = -n - \frac{1}{n} \sum_{i=1}^{n} (2i - 1)(log(\Phi(y_i)) + log(1 - \Phi(y_{n-i+1}))) \tag{2}$$

Gdy wartość średnia i odchylenie standardowe w testowanym zbiorze danych są obliczane na jego podstawie (a nie znane), wartość statystyki należy poprawić według wzoru 3 [9].

$$A^{2*} = A^2(1 + \frac{4}{n} + \frac{25}{n^2}) \tag{3}$$

Parametrem wejściowym dla tego testu jest poziom ufności α. Jest on zwykle wyrażony w procentach, bądź w ułamku dziesiętnym. Zależy od niego tzw. wartość krytyczna, czyli próg wartości statystyki, ponad którym odrzucana jest hipoteza o rozkładzie normalnym danych. Wzorem autorów algorytmu G-średnich, użyto $\alpha = 0.0001$ [9].

Test Andersona - Darlinga można stosować tylko do jednowymiarowych danych, a więc należy sprowadzić wyjściowy zbiór (o wymiarze d) do przestrzeni liczb rzeczywistych. W tym celu algorytm G-średnich dla każdej klasy używa metody k-średnich z $k = 2$ oraz dwoma środkami $c_j^{1,2} = c_j \pm m$, gdzie m jest wektorem o normie niewielkiej w porównaniu z odległościami między punktami w klasie. Niech otrzymane w wyniku tej operacji środki to c^1 oraz c^2, a wektor u opisuje odległość między nimi: $u = c^1 - c^2$. Cała klasa S_j jest rzutowana prostopadle na wektor u, w wyniku czego otrzymuje się jednowymiarową przestrzeń S'_j, dla której po normalizacji stosuje się test Andersona - Darlinga. W przypadku gdy wartość statystki jest mniejsza niż wartość krytyczna dla danej ufności, nowe środki są odrzucane. W przeciwnym razie zachowuje się je, dzieląc klasę S_j na dwie.



Rysunek 2: *Przykład rzutowania zbioru danych na wektor łączący znalezione środki ciężkości. Źródło: [10]*

Rysunek 2 przedstawia przykład takiego rzutowania. Widać na nim dwa nowe środki znalezione przez metodę k-średnich dla danej klasy oraz rzutowanie prostopadłe punktów tej klasy na prostą wyznaczaną przez wektor między tymi dwoma środkami. Jak można się spodziewać, rozkład takiego zbioru danych jest bimodalny, a nie normalny, tak więc te dwa widoczne na rysunku środki zostaną przyjęte.

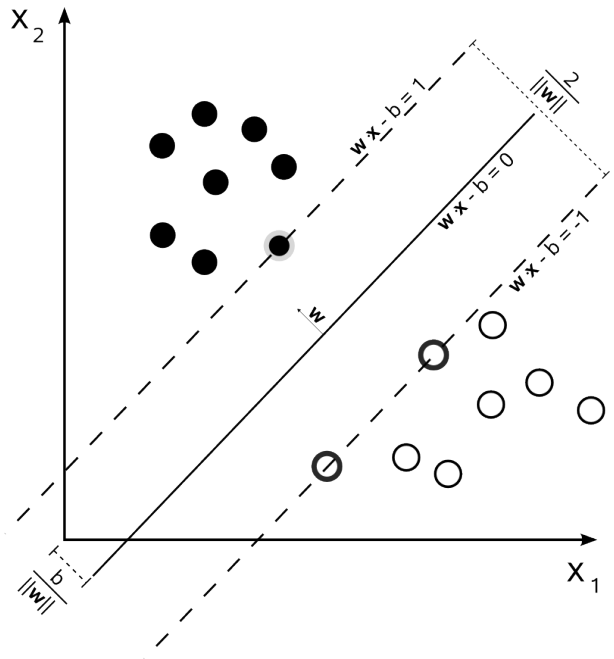
Cały zastosowany algorytm klasteryzacji może być przedstawiony w następujących krokach:

1. Jako parametry wejściowe przyjmij zbiór danych oraz ufność testu Andersona-Darlinga.
2. Wylicz początkowy środek ciężkości: $C = \{\bar{x}\}$.
3. Wykonaj klasteryzację: C = k-średnich(X, C).
4. Dla każdej klasy S_j sprawdź jej rozkład testem Andersona-Darlinga:
   (a) Wylicz dwa środki pochodne c_j^1, c_j^2.
   (b) Wykonaj ponowną klasteryzację: $\{c^1, c^2\} = k - średnich(S_j, \{c_j^1, c_j^2\})$.
   (c) Wyznacz wektor $u = c^1 - c^2$.
   (d) Wyznacz jednowymiarową przestrzeń S'_j.
   (e) Wylicz wartość statystyki Andersona - Darlinga dla S'_j.
   (f) Jeśli jest ona większa od wartości krytycznej, podziel klasę S_j na dwie ze środkami c^1 oraz c^2. Jeśli jest mniejsza, zachowaj poprzedni środek.
5. Powtarzaj od kroku 3 dopóki żadne nowe środki nie zostaną dodane.

III. Klasyfikacja

Aby sklasyfikować powstałe w poprzednim kroku klastry wykorzystano klasyfikator SVM (Support Vector Machine). Problem klasyfikacji wymaga podziału zbioru danych wejściowych na zbiór uczący oraz zbiór testowy. Każdy z elementów zbioru uczącego zawiera wartość oczekiwaną (np. etykieta klasy) oraz jakąś ilość atrybutów (np. wektor cech). Celem SVM jest stworzenie modelu (bazujcego na danych uczących), który przewiduje nieznaną wartość oczekiwaną (etykietę) dla podanego na wejściu elementu zbioru testowego.

W najprostszej postaci klasyfikator SVM służy do wyznaczenia hiperpłaszczyzny rozdzielającej dwa liniowo separowalne zbiory. Hiperpłaszczyzna ta wyznaczana jest z maksymalnym marginesem, tzn. tak, aby suma jej odległości od najbliższych próbek z obu klas była jak największa (patrz rys. 3).



Rysunek 3: *Dwuwymiarowy przypadek hiperpłaszczyzny rozdzielającej dwie klasy z zaznaczonym marginesem. Źródło [11]*

Mając dany zbiór uczący, będący zbirem par składających się z wektora cech oraz etykiety (x_i, y_i), $i = 1, ..., l$, gdzie $x_i \in \mathbb{R}^n$ i $y_i \in \{1, -1\}$, maszyna wektorów nośnych (SVM) wymaga rozwiązania następującego problemu optymalizacji [12]:

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \tag{4}$$

z ograniczeniem:

$$\begin{aligned} &y_i \left(w^T \phi \left(x_i \right) + b \right) \geqslant 1 - \xi_i, \\ &\xi_i > 0 \end{aligned} \tag{5}$$

W wielu przypadkach nie można zagwarantować liniowej separowalności zbiorów. W takich sytuacjach stosuje się tzw. Kernel Trick. Polega to na zwiększeniu wymiaru przestrzeni danych wejściowych, aby w nowej przestrzeni istniała własność liniowej separowalności zbiorów.

W zdefiniowanym wzorami (4) oraz (5) problemie optymalizacji wektor uczacy x_i przekształcany jest do przestrzeni o większym wymiarze dzięki funkcji ϕ. Parametr $C > 0$ jest karą za niespełnienie warunków zadania. Ponadto, funkcja $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ jest nazywana funkcją jądra (ang. kernel function). Poniżej wypisano cztery przykładowe funkcje jądra, jakie można spotkać w literaturze poświęconej SVM.

1. Liniowa: $K\left(x_i, y_j\right) = x_i^T x_j$
2. Wielomianowa: $K\left(x_i, y_j\right) = \left(\gamma x_i^T x^j + r\right)^d, \gamma > 0$
3. Radialna funkcja bazwa (RBF): $K\left(x_i, y_j\right) = \exp\left(-\gamma \|x_i - x_j\|^2\right), \gamma > 0$
4. Sigmoida: $K\left(x_i, y_j\right) = \text{tgh}\left(\gamma x_i^T x_j + r\right)$

W opisywanym module wykorzystana została funkcja RBF (ang. Radial Basis Function).

Aby klasyfikator mógł działać wcześniej należy go wytrenować. Polega to na podaniu mu ciągu wektorów uczących. Opisywany klasyfikator został wytrenowany za pomocą bazy danych MIT-BIH Arrhythmia Database [13]. Gotowy model klasyfikatora wczytywany jest z pliku, w którym zapisane są różne parametry oraz zestaw wektorów nośnych, na których opiera się działanie metody SVM.

III. Rezultaty i wnioski

<Podrozdziały: "Funkcjonowanie algorytmu G-średnich", "Funkcjonowanie algorytmu SVM»

Pomiary czasu wykonywania programów napisanych w poszczególnych językach dla różnych paczek danych z bazy MIT-BIH zostały przedstawione w Tab.1.

Krótki czas wykonywania programu napisanego w Matlabie wynika głównie z wykorzystania biblioteki libsvm [12], która jest napisana w C. Dla pozostałych języków zaimplementowano maszynę wektorów nośnych w oparciu o pliki źródłowe wspomnianej biblioteki.

Wykorzystanie metody SVM z szesnastoelementowym wektorem cech wydaje się złym rozwiązaniem, gdyż niemal zawsze wektor klasyfikowany jest do jednej klasy - patrz Rys.4. Poza zbyt licznym wektorem cech wpływ na to ma również fakt, że około 90% zespołów QRS z bazy MIT-BIH reprezentują pobudzenia nadkomorowe. Rozwiązaniem problemu klasyfikacji może okazać się odpowiednie zmniejszenie liczności wykorzystywanego wektora cech.

Przeprowadzono również testy, w których SVM korzystał z różnych modeli (wygenerowanych z różnych paczek danych). Zaobserwowano, że normalizacja wektorów uczących znacznie przyspiesza proces uczenia, czego potwierdzeniem jest Rys.3.

<Coś o gmeans - że w C++ nie było, a tutaj jest>

IV. Podsumowanie

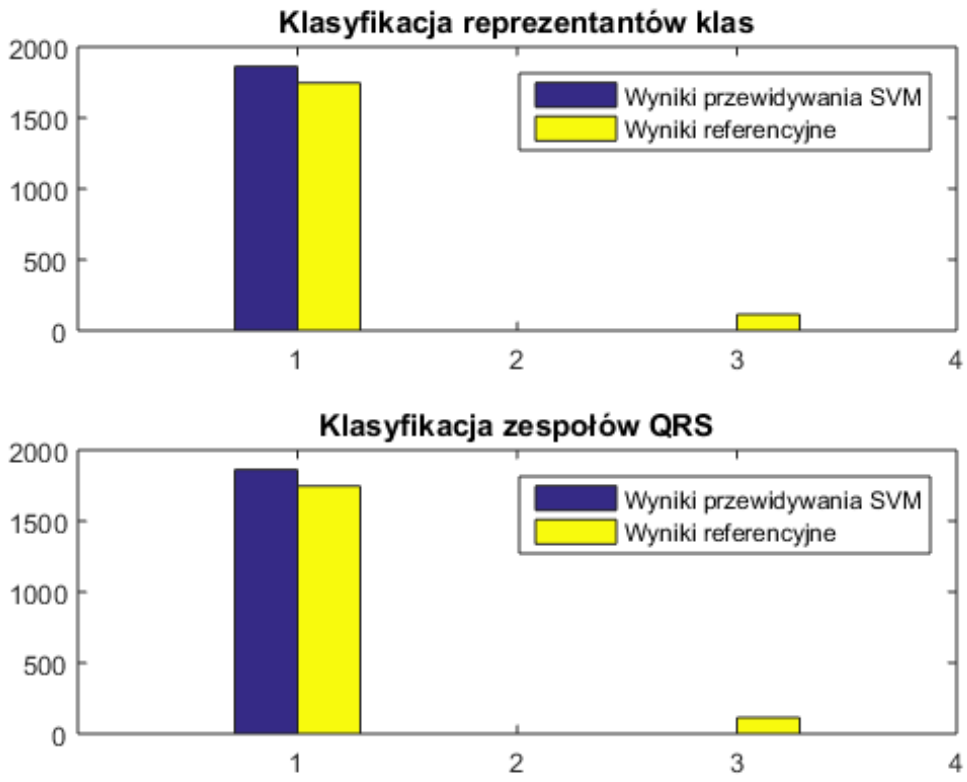
W ramach projektu zaimplementowano moduł HeartClass w trzech różnych językach programowania: Matlabie, Pythonie (3.4.x) oraz Julii. Moduł ten ma celu klasyfikowanie zespołów QRS sygnału EKG. W pracy udało się zaimplementować moduł odpowiedzialny za klasteryzację oraz klasyfikację. Z wykonanych testów wynika, że moduł najszybciej wykonuje się w Matlabie. Powodem tego jest głównie fakt, iż w tym module wykorzystano gotową bibliotekę, a nie, jak w przypadku pozostałych dwóch, gdzie zaimplementowano swoją maszynę SVM.

Tablica 1: *Czasy wykonywania programu w poszczególnych językach dla różnych paczek danych. Wyniki wyrażone w sekundach*

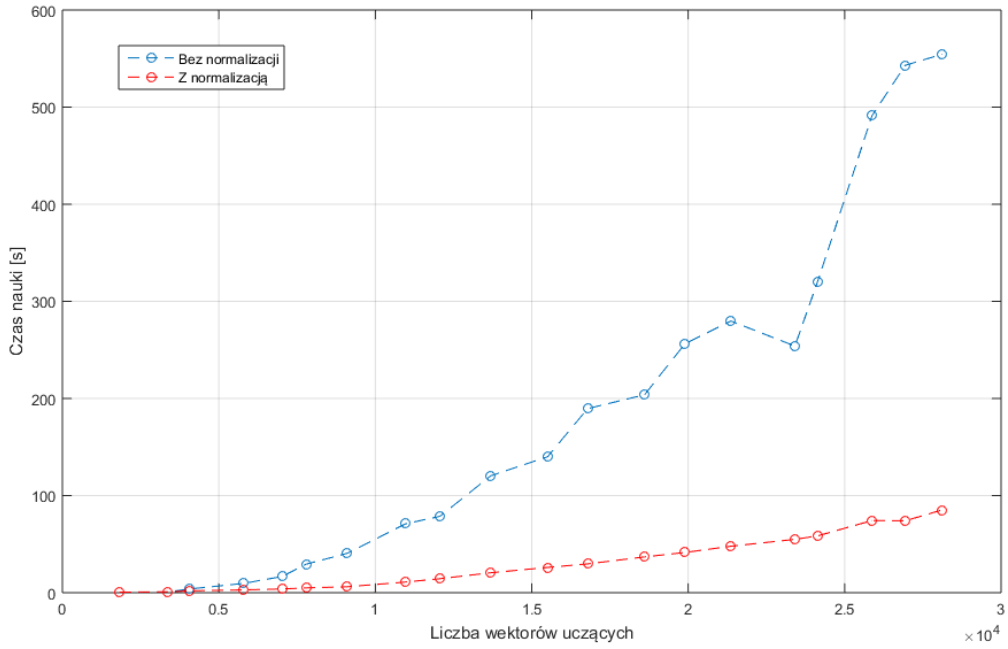
Nr paczki z MIT-BIH	C++	Matlab	Python	Julia
100		1.9523	12.815439	5.901852174
101		1.4195	10.492137	1.899932399
102		1.2609	12.242185	2.215825099
103		2.3951	11.708208	2.179983401
104		1.6457	11.895952	2.133522994
105		1.3186	14.702323	3.123101161
106		1.8217	10.365989	2.114899549
107		1.5331	9.988119	2.069017174
108		1.7391	8.503747	2.10760114
109		1.2506	13.910034	3.42160439
111		1.8369	11.575133	2.874097787
112		2.2931	14.707609	3.345013555
113		1.2857	10.2796	1.938375007
114		0.5056	5.547578	1.534204408
115		1.2325	11.059755	2.207411682
116		1.242	13.344349	2.587637119
117		1.506	8.462129	1.643660476
119		1.5771	10.99204	2.071278448
121		2.0902	10.279925	1.976437859
122		2.0928	14.211312	3.117896543
123		0.7942	8.39699	1.721477354
124		1.6778	8.137572	1.714434604
200		1.8849	11.517102	2.827846386
201		1.8963	9.249272	1.864559098
202		2.132	13.789148	3.778367525
203		1.2565	14.579523	3.147122344
205		1.3475	15.066897	3.260278958
208		2.4411	15.573072	3.529202731
209		2.139	17.397029	4.002761863
210		1.1721	26.913411	6.433184431
212		2.9475	26.890231	3.390088991
213		2.3988	31.479429	4.146098275
214		1.8288	18.571949	2.420839058
215		1.6584	30.439442	3.954894738
217		2.0103	18.701152	2.587683551
219		1.8043	19.748443	3.281731301
220		1.2670	18.710136	2.665970912
221		1.7631	13.708571	3.006166138
222		1.2655	14.3837	3.223767787
223		1.2356	13.731657	3.006029445
230		1.8286	14.19151	2.875082168
231		1.4458	8.698872	3.391200109
232		1.3707	10.32135	2.354562975
233		2.5759	15.583746	3.477575961
234		3.6223	16.492193	3.583639883

Tablica 2: *Liczby klas wyliczonych przez G-means*

Nr paczki z MIT-BIH	Matlab	Python	Julia
100	16	16	16
101	14	16	16
102	14	16	16
103	27	16	16
104	29	16	16
105	1	20	20
106	35	16	16
107	31	16	16
108	32	15	15
109	1	21	21
111	22	17	17
112	15	19	19
113	13	16	16
114	1	15	15
115	15	16	16
116	1	16	16
117	23	16	16
119	33	16	16
121	27	16	16
122	29	21	21
123	1	16	16
124	30	15	15
200	32	20	20
201	56	13	13
202	29	21	21
203	1	26	26
205	1	21	21
208	36	24	24
209	21	31	31
210	1	20	20
212	18	25	25
213	28	32	32
214	33	18	18
215	18	32	32
217	65	17	17
219	36	15	15
220	22	16	16
221	33	19	19
222	1	22	22
223	1	18	18
230	15	20	20
231	39	12	12
232	28	17	17
233	42	24	24
234	27	29	29



Rysunek 4: *Histogramy klasyfikacji zespołów dla paczki danych o numerze 101*



Rysunek 5: *Wpływ normalizacji na czas nauki SVM*

V. Bibliografia

[1] QRS Complex, `https://en.wikipedia.org/wiki/QRS_complex` . Stan na: 05.12.2015 r.

[2] Augustyniak, P. *Przetwarzanie sygnałów elektrodiagnostycznych*. Uczelniane Wydawnictwo Naukowo - Dydaktyczne, AGH, Kraków, 2001 r.

[3] Studenci Automatyki i Robotyki. *ESDMiT - Raport końcowy*. 04.02.2015 r.

[4] Song, M. H., i inni. *Support Vector Machine Based Arrhythmia Classification Using Reduced Features*. International Journal of Control, Automation, and Systems, vol. 3, nr 4, strony 571-579, grudzień 2005.

[5] Thakare, A. S. *QRS Complex Detection and Arrhythmia Classification*. Uniwersytet Pune, Indie.

[6] Laguna, P., Jane, R., Caminal, P. *Adaptive Feature Extraction for QRS Classification and Ectopic Beat Detection*. Computers in Cardiology, IEEE, 1991.

[7] Moraes, J.C.T.B., Seixas, M.O., Vilani, F.N., Costa, E.V. *A Real Time QRS Complex Classification Method using Mahalanobis Distance*. Escola Politécnica da Universidade de São Paulo, São Paulo, Brazylia.

[8] MacQueen, J. *Some methods for classification and analysis of multivariate observations*. W materiałach: "Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics", strony 281-297, University of California Press, Berkeley, Kalifornia, USA, 1967.

[9] Hamerly, G., Elkan, Ch. *Learning the k in k-means*. W: Neural Information Processing Systems, MIT Press, 2003.

[10] Użytkownik 'ashenfad'. *Divining the 'K' in K-means Clustering*. http://blog.bigml.com/2015/02/24/divining-the-k-in-k-means-clustering/. Dodane: 24.02.2015 r. Stan na: 05.12.2015 r.

[11] Support vector machine, https://en.wikipedia.org/wiki/Support_vector_machine . Stan na: 05.12.2015 r.

[12] Chih-Wei, H., Chih-Chung C., Chih-Jen L. A Practical Guide to Support Vector Classification Department of Computer Science, National Taiwan University, Tajpej, Tajwan, 2003 https://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf. Stan na 05.12.2015 r.

[13] MIT-BIH Arrhythmia Database. https://www.physionet.org/physiobank/database/mitdb/ Stan na: 05.12.2015 r.

Dodatek

A. zagadnienia implementacyjne

Zasadę działania zaimplementowanego modułu można opisac w trzech następujących krokach:

- Ekstrakcja cech
- Klasteryzacja
- Klasyfikacja

Wejściem modułu są wektory danych otrzymywanych z poprzednich modułów. Na etapie ekstrakcji cech wybierane są cechy, z których składać się będą wektory podlegajace klasyfikacji. !!!!!!!!DOPISAĆ ALBO OLAĆ!!!!!!!!!!!!

<Podrozdziały: Ćiekawsze aspekty wykonanych implementacji", "Instrukcja uruchomienia", Żóżnice względem kodu w C++>