Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Instytut Systemów Elektronicznych

Praca dyplomowa inżynierska

na kierunku Elektronika
w specjalności Elektronika i Informatyka w Medycynie

Rozmyty system klasyfikacji optymalizowany algorytmem strategii ewolucyjnej

Igor Dawid Markiewicz

Numer albumu 269030

promotor
dr inż. Grzegorz Nieradka

WARSZAWA 2018

Streszczenie pracy

Tytuł: ROZMYTY SYSTEM KLASYFIKACJI OPTYMALIZOWANY ALGORYTMEM STRATEGII EWOLUCYJNEJ

Niniejsza praca inżynierska skupia się na zagadnieniach związanych z zastosowaniem systemów rozmytych w klasyfikacji danych. W ramach realizacji pracy zaimplementowany został algorytm klasyfikacji nazwany algorytmem Sorena. Zaproponowano i wprowadzono autorskie modyfikacje, w tym zastosowanie algorytmu strategii ewolucyjnej do optymalizacji systemu. W pracy zostały przeprowadzone testy wybranych parametrów systemu oraz porównano osiągane rezultaty klasyfikacji z algorytmami znanymi w literaturze jako Chi oraz Ishibuchi & Nakashima. Testy przeprowadzone zostały na biologicznych i medycznych zbiorach danych. Pierwsza część pracy skupia się na wprowadzeniu teoretycznym to teorii systemów rozmytych oraz algorytmów genetycznych i strategii ewolucyjnych. Druga część zawiera opis teoretyczny i założenia implementowanego systemu. Kolejna część pracy skupia się na wynikach badań oraz wnioskach. Ostatnie dwie części stanowią podsumowanie z wnioskami końcowymi oraz przedstawia możliwości dalszego rozwoju.
Słowa kluczowe: zbiory rozmyte, logika rozmyta, systemy rozmyte, strategie ewolucyjne, klasyfikacja.

Abstract of thesis

Title: A FUZZY CLASSIFICATION SYSTEM OPTIMIZED BY AN ALGORITHM OF EVOLUTIONARY STRATEGY

The main purpose of this BSc thesis was to develop and implement a fuzzy system for classification task. For some system known from literature, named in the thesis as Soren algorithm author propsed and introduced some improvments. The main improvement of the Soren algorithm was the use of an evolutionary strategy algorithm to optimize the system. The thesis examines selected system parameters and compares the results of classification tasks with other fuzzy systems known from the literature as Chi and Ishibuchi & Nakashima algorithms. The experiments were conducted on biological and medical data sets. The first part of the paper focuses on theoretical introduction to the theory of fuzzy systems as well as genetic algorithms and evolutionary strategies. The second part contains a theoretical description and assumptions of the implemented system. The next part focuses on research results and conclusions.The last two parts are a summary of the final conclusions and the presentation of further development opportunities.
Keywords: fuzzy stes, fuzzy logic, fuzzy systems, evolutionary strategies, classification

załącznik do zarządzenia nr 28/2016 r.
Rektora PW

Politechnika Warszawska

„załącznik nr 3 do zarządzenia nr 24/2016 Rektora PW

.....
miejscowość i data

.....
imię i nazwisko studenta

.....
numer albumu

.....
kierunek studiów

OŚWIADCZENIE

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

.....
czytelny podpis studenta"

Spis treści

1 Wstęp

Jednym z ważnych problemów współczesnej analizy danych są zagadnienia związane z klasyfikacją danych. Problematyka ta istnieje dużo dłużej niż obecnie rozumiana informatyka, uczenie maszynowe (ang. *machine learning*) czy analiza danych (ang. *data science*) i obejmowała niegdyś przewidywanie sytuacji na podstawie statystycznej analizy dostępnych przypadków. Wraz z rozwojem technologii i wzrostem mocy obliczeniowej komputerów, zaczęły powstawać rozwiązania mające na celu przeprowadzanie kategoryzacji otrzymywanych danych. Stanowią one obecnie podstawę do wykorzystania ich w wielu dziedzinach życia takich jak finanse, medycyna, fizyka, automatyka, czy informatyka. Przykładem takiego systemu jest regułowy system ekspertowy *Mycin*, służący do diagnozowania chorób krwi oraz przedstawiania propozycji leczenia [1].
Jako przykładowe systemy służące do klasyfikacji możemy wyróżnić [2, 3]:

- systemy oparte o regresję logistyczną – analiza oparta o statystykę korzystająca z procesu Bernoulliego
- systemy regułowe – oparte o wiedzę ekspercką lub miarę statystycznego przyrostu informacji np: entropię
- drzewa decyzyjne – oparte o wiedzę ekspercką lub miary przyrostu informacji (lub wielkości popełnianego błędu)
- lasy losowe – zbiór drzew decyzyjnych w których każde drzewo zwraca swoją odpowiedź po czym system wybiera tę kategorię za którą zagłosowało najwięcej drzew
- sieci neuronowe (oraz głębokie sieci neuronowe) – oparte o uczenie na podstawie biologicznego wzorca neuronu
- inne systemy w tym np: systemy oparte o logikę rozmytą których dotyczy niniejsza praca

Przedstawiona praca zawiera analizę działania systemów w przypadku klasyfikacji danych medycznych, jednak zaprojektowany system działa w sposób autonomiczny i może być użyty do dowolnego typu danych.

1.1 Cel pracy

Celem pracy było opracowanie i implementacja systemu rozmytego do zadań klasyfikacyjnych oraz przeprowadzenie na nim testów. Założono możliwość wykorzystania systemu już istniejącego, ale przeznaczenia ogólnego który zostałby dostosowany do potrzeb pracy. Założono również możliwość wprowadzenia modyfikacji, a także przeprowadzenie badań na różnych zbiorach danych medycznych i porównanie otrzymanych wyników z innymi algorytmami klasyfikującymi.

1.2 Motywacja do powstania pracy

Motywacją do powstania pracy było zainteresowanie dziedziną sztucznej inteligencji oraz przetwarzania danych. Systemy rozmyte, mimo iż stosowane do analizy danych, największe znaczenie mają

obecnie w automatyce. Nie jest to więc dział bardzo silnie eksploatowany w zakresie metod klasyfikacji, przez co wydał się on dostatecznie ciekawy jako temat pracy inżynierskiej.

2 Podstawy teoretyczne systemów rozmytych

W tym rozdziale przedstawione zostały podstawy teoretyczne teorii zbiorów, logiki oraz systemów rozmytych. Przedstawione definicje stanowią podstawę umożliwiającą zrozumienie działania opracowanego systemu rozmytego.

2.1 Podstawy teorii mnogości

Pierwsze wprowadzone pojęcia są związane z rozmytą teorią zbiorów i krótkim porównaniem jej z klasyczną teorią zbiorów.

2.1.1 Zbiór w ujęciu klasycznym

W ujęciu klasycznym zbiorem nazywamy pojęcie pierwotne, reprezentujące pewną kolekcję elementów, w którym nie ma znaczenia kolejność oraz wszystkie duplikaty elementów są równoważne z nimi samymi. Ważną cechą zbiorów klasycznych (zwanych także ostrymi) jest możliwość jednoznacznego stwierdzenia czy jakiś podmiot jest elementem danego zbioru czy też nie. Prowadzi to do definicji funkcji charakterystycznej zbioru (indykatora) [4, 5, 6].

Definicja 1 *Niech X będzie pewnym uniwersum, a zbiór A jego podzbiorem. Wtedy funkcję $\chi_A : X \longrightarrow \{0, 1\}$ określoną jako*

$$\chi_A(x) = \begin{cases} 1, & gdy\ x \in A \\ 0, & gdy\ x \notin A \end{cases}$$

nazywamy funkcją charakterystyczną (indykatorem) zbioru A.

Częstą formą zapisywania takich zbiorów jest reprezentacja stwierdzająca że dany zbiór jest kolekcją tych elementów pewnego zbioru A, które spełniają określoną formułę logiczną $W(x)$ [4, 5].

Definicja 2 *Zbiór A definiujemy na podstawie formuły logicznej $W(x)$ jako*

$$A = \{x \in X \mid W(x)\}\ gdzie\ A \subseteq X$$

2.1.2 Zbiór w ujęciu rozmytym

Zagadnienia dotyczące zbiorów rozmytych przedstawione zostały szerzej w literaturze [6, 7].
W odróżnieniu do klasycznej teorii zbiorów, w przypadku zbiorów rozmytych mamy możliwość stwierdzenia w jakim stopniu pewien element należy do danego zbioru. Prowadzi to do wielu ważnych następstw. Jednym z nich jest konieczność zmiany funkcji charakterystycznej zbioru na funkcję mającą możliwość odpowiedniego reprezentowania przynależności elementów. Funkcję taką nazywamy funkcją przynależności.

Definicja 3 *Funkcją przynależności dla zbioru A na uniwersum X nazywamy funkcję*

$$\mu_A : X \longrightarrow [0, 1]$$

Zbiory mające tak określoną funkcję przynależności nazywamy zbiorami rozmytymi typu pierwszego. Niniejsza praca będzie w całości dotyczyć tego typu zbiorów, dlatego w dalszej części będziemy używali uproszczonego zwrotu - zbiór rozmyty.

Definicja 4 *Zbiorem rozmytym nazywamy zbiór par postaci*

$$A = \{(x, \mu_A(x)) \mid x \in X \wedge \mu_A(x) \in [0,1]\}$$

Przedstawiona powyżej definicja formalna zbioru rozmytego często jest niewygodna w stosowaniu, dlatego zazwyczaj utożsamia się go z jego funkcją przynależności. Nie jest to zabieg w pełni poprawny, jednak możliwy do zaakceptowania uwzględniając intuicję.
Zbiór rozmyty o skończonej lub nieskończonej ale przeliczalnej ilości elementów, możemy zapisać jako

$$\sum_{x \in X} \mu_A(x)|x \tag{1}$$

A zbiór rozmyty o nieskończonej i nieprzeliczalnej ilości elementów jako

$$\int_X \mu_A(x)|x \tag{2}$$

Istotną czynnością jest zdefiniowanie odpowiednich, głównych terminów pozwalających na pracę ze zbiorami rozmytymi.

Definicja 5 *Rozmytym zbiorem pustym (zapisywanym jako $\varnothing$) nazywamy zbiór taki że*

$$\mu_A(x) = 0 \ \forall x \in X$$

Definicja 6 *Nośnikiem zbioru rozmytego A nazywamy zbiór ostry $supp(A)$ określony jako*

$$supp(A) = \{x \mid \mu_A(x) > 0\}$$

Definicja 7 *Jądrem zbioru rozmytego A nazywamy zbiór ostry $ker(A)$ określony jako*

$$ker(A) = \{x \mid \mu_A(x) = 1\}$$

Jeśli jądro posiada tylko jeden element, to nazywamy go wartością szczytową.

Definicja 8 *Wysokością zbioru rozmytego A nazywamy zbiór ostry $hgt(A)$ określony jako*

$$hgt(A) = \sup_{x \in A} \mu_A(x)$$

Definicja 9 *Zbiór rozmyty A nazywamy normalnym jeśli spełniony jest warunek*

$$hgt(A) = 1$$

Każdy, niepusty zbiór rozmyty możemy przekształcić do zbioru normalnego z zastosowaniem operacji

$$\mu_{A_{zn}} = \frac{\mu_A(x)}{hgt(A)} \tag{3}$$

Definicja 10 *Zbiór rozmyty A zawiera się w zbiorze rozmytym B (co oznaczamy jako $A \subset B$) wtedy i tylko wtedy, gdy*

$$\mu_A(x) \leq \mu_B(x) \ \forall x \in X$$

Definicja 11 *Zbiór rozmyty A jest równy zbiorowi rozmytemu B (co oznaczamy jako $A = B$) wtedy i tylko wtedy, gdy*

$$\mu_A(x) = \mu_B(x) \ \forall x \in X$$

Definicja 12 *α przekrojem zbioru $A \subseteq X$, oznaczanym jako A_α nazywamy zbiór ostry opisany wzorem*

$$A_\alpha = \{x \in X \mid \mu_A(x) \geq \alpha\}$$

Jest to zbiór określony funkcją przynależności $\chi_{A_\alpha}(x)$

$$\chi_{A_\alpha}(x) = \begin{cases} 1, & gdy\ \mu_A(x) \geq \alpha \\ 0, & gdy\ \mu_A(x) < \alpha \end{cases}$$

Definicja 13 *Zasada dekompozycji - możemy przy pomocy przekrojów podzielić zbiór rozmyty A na zbiory rozmyte αA_α :*

$$A = \bigcup_{\alpha \in [0,1]} \alpha A_\alpha$$

gdzie

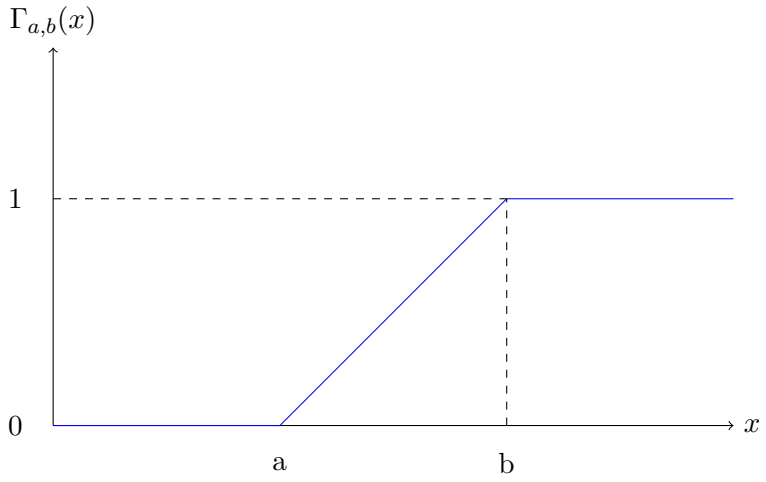
$$\mu_{\alpha A_\alpha}(x) = \begin{cases} \alpha, & gdy\ x \in A_\alpha \\ 0, & gdy\ x \notin A_\alpha \end{cases}$$

2.2 Główne typy funkcji przynależności jednej zmiennej

Poniżej zostają przedstawione główne typy funkcji przynależności używanych w systemach rozmytych. Zostały one przedstawione na podstawie źródeł [7, 8]

- funkcje klasy Γ

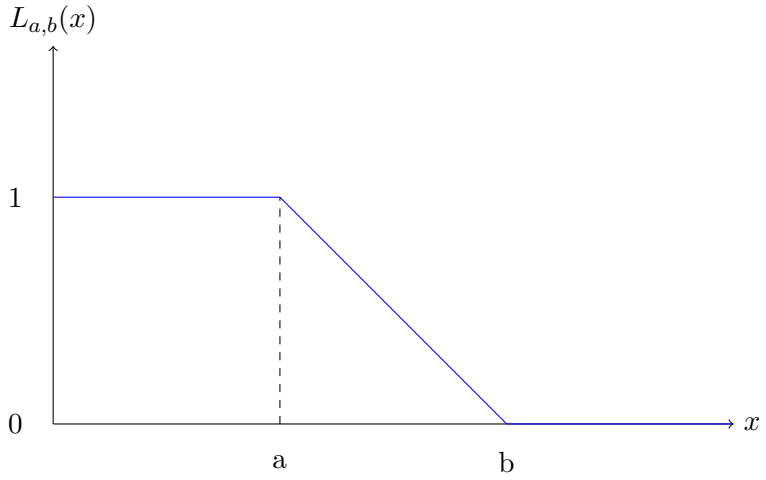
$$\Gamma_{a,b}(x) = \begin{cases} 0, & \text{gdy } x \leq a \\ \frac{x-a}{b-a}, & \text{gdy } a < x \leq b \\ 1, & \text{gdy } x > b \end{cases}$$



Rys. 1: Wykres funkcji przynależności klasy Γ

- funkcje klasy L

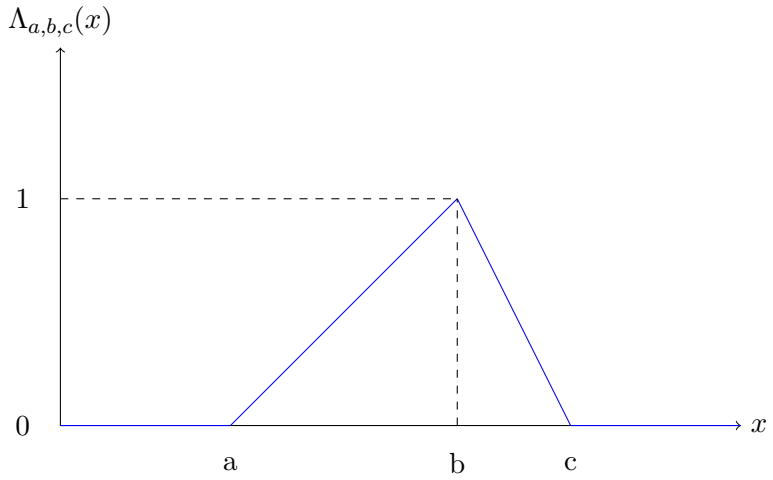
$$L_{a,b}(x) = \begin{cases} 0, & \text{gdy } x \leq a \\ \frac{b-x}{b-a}, & \text{gdy } a < x \leq b \\ 1, & \text{gdy } x > b \end{cases}$$



Rys. 2: Wykres funkcji przynależności klasy L

- funkcje klasy Λ

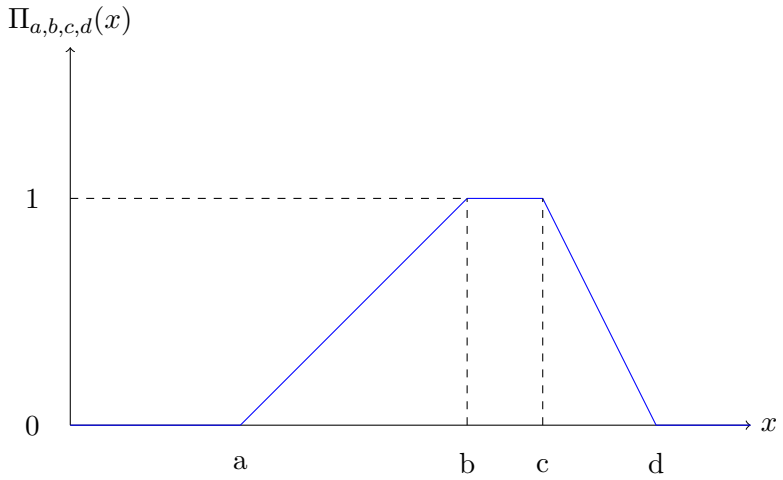
$$\Lambda_{a,b,c}(x) = \begin{cases} 0, & \text{gdy } x \leq a \vee x \geq c \\ \frac{x-a}{b-a}, & \text{gdy } a < x \leq b \\ \frac{c-x}{c-b}, & \text{gdy } b < x < c \end{cases}$$



Rys. 3: Wykres funkcji przynależności klasy Λ

- funkcje klasy Π

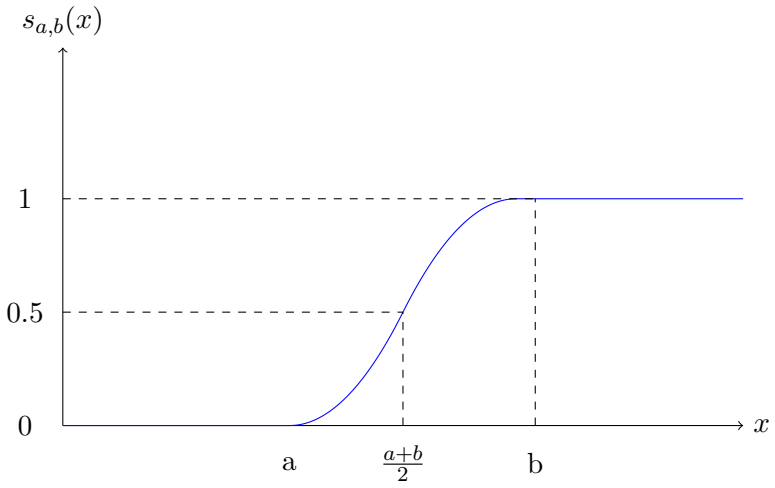
$$\Pi_{a,b,c,d}(x) = \begin{cases} 0, & \text{gdy } x \leq a \vee x \geq d \\ \frac{x-a}{b-a}, & \text{gdy } a < x \leq b \\ 1, & \text{gdy } b < x \leq c \\ \frac{d-x}{d-c}, & \text{gdy } c < x < d \end{cases}$$



Rys. 4: Wykres funkcji przynależności klasy Π

- funkcje klasy s

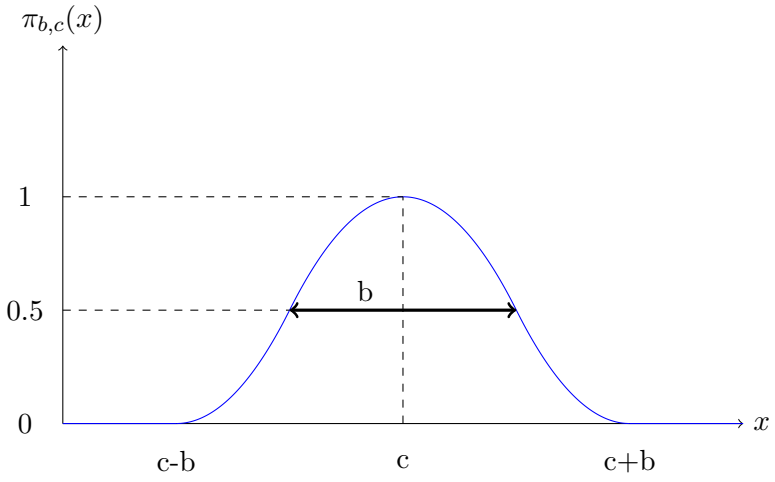
$$s_{a,b}(x) = \begin{cases} 0, & \text{gdy } x \leq a \\ 2\left(\frac{x-a}{b-a}\right)^2, & \text{gdy } a < x \leq \frac{a+b}{2} \\ 1 - 2\left(\frac{x-b}{b-a}\right)^2, & \text{gdy } \frac{a+b}{2} < x < b \\ 1, & \text{gdy } x \geq b \end{cases}$$



Rys. 5: Wykres funkcji przynależności klasy s

- funkcje klasy π

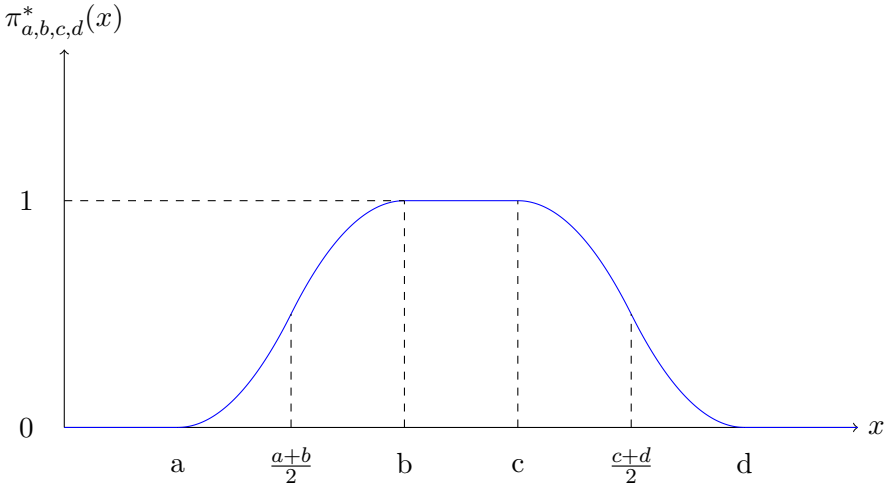
$$\pi_{b,c}(x) = \begin{cases} s_{c-b,c}(x), & \text{gdy } x < c \\ 1 - s_{c,c+b}(x), & \text{gdy } x \geq c \end{cases}$$



Rys. 6: Wykres funkcji przynależności klasy π

- funkcje klasy π^*

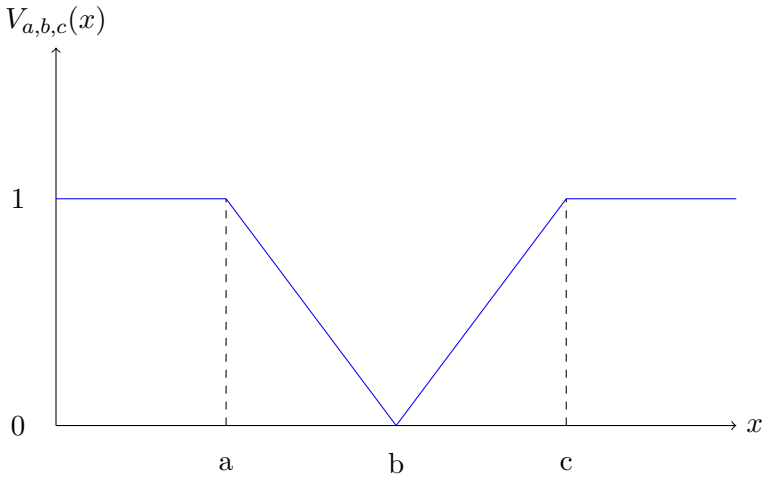
$$\pi^*_{a,b,c,d}(x) = \begin{cases} 0, & \text{gdy } x \leq a \\ 2\left(\frac{x-a}{x-b}\right)^2, & \text{gdy } a \leq x \leq \frac{a+b}{2} \\ 1 - 2\left(\frac{x-b}{b-a}\right)^2, & \text{gdy } \frac{a+b}{2} \leq x \leq b \\ 1, & \text{gdy } b \leq x \leq c \\ 1 - 2\left(\frac{x-c}{d-c}\right)^2, & \text{gdy } c \leq x \leq \frac{c+d}{2} \\ 2\left(\frac{x-d}{d-c}\right)^2, & \text{gdy } \frac{c+d}{2} \leq x \leq d \\ 0, & \text{gdy } x \geq d \end{cases}$$



Rys. 7: Wykres funkcji przynależności klasy π^*

- funkcje klasy V

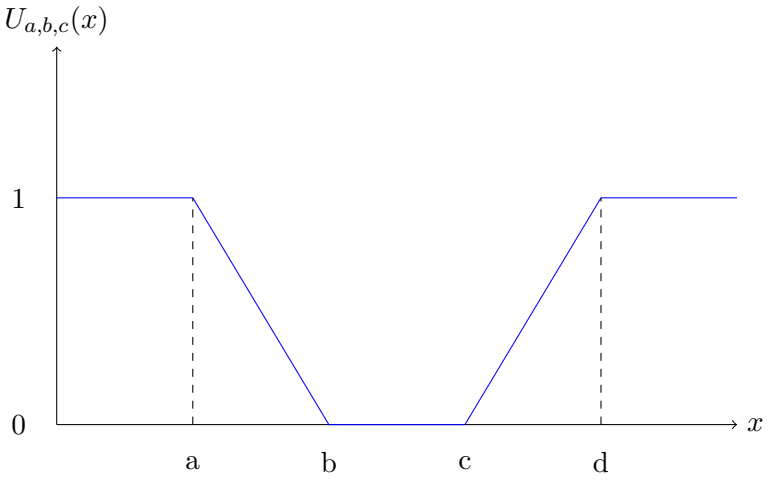
$$V_{a,b,c}(x) = 1 - \Lambda_{a,b,c}(x) = \begin{cases} 1, & \text{gdy } x \leq a \vee x \geq c \\ \frac{b-x}{b-a}, & \text{gdy } a < x \leq b \\ \frac{x-b}{c-b}, & \text{gdy } b < x < c \end{cases}$$



Rys. 8: Wykres funkcji przynależności klasy V

- funkcje klasy U

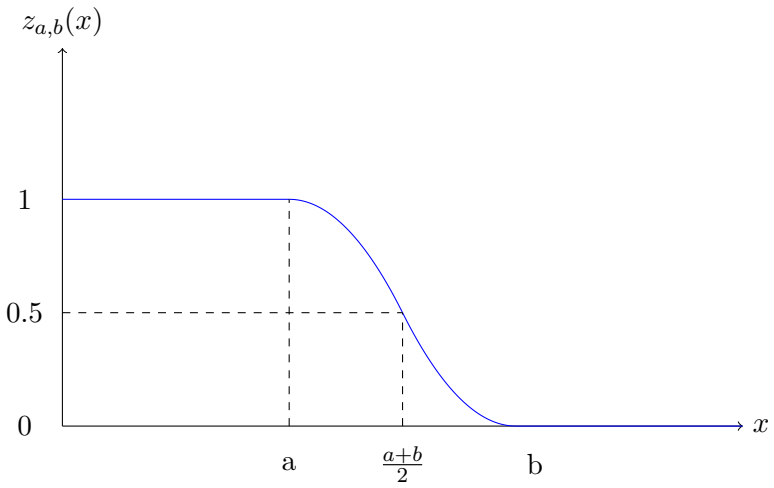
$$U_{a,b,c,d}(x) = 1 - \Pi_{a,b,c,d}(x) = \begin{cases} 1, & \text{gdy } x \leq a \vee x \geq d \\ \frac{b-x}{b-a}, & \text{gdy } a < x \leq b \\ 0, & \text{gdy } b < x \leq c \\ \frac{x-c}{d-c}, & \text{gdy } c < x < d \end{cases}$$



Rys. 9: Wykres funkcji przynależności klasy U

- funkcje klasy z

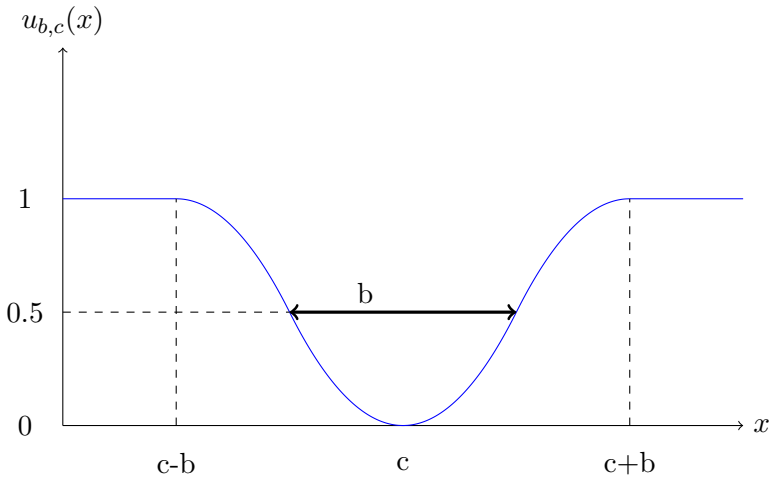
$$z_{a,b} = 1 - s_{a,b}(x) = \begin{cases} 1, & \text{gdy } x \leq a \\ 1 - 2\left(\frac{x-a}{b-a}\right)^2, & \text{gdy } a < x \leq \frac{a+b}{2} \\ 2\left(\frac{x-b}{b-a}\right)^2, & \text{gdy } \frac{a+b}{2} < x < b \\ 0, & \text{gdy } x \geq b \end{cases}$$



Rys. 10: Wykres funkcji przynależności klasy z

- funkcje klasy u

$$u_{b,c} = 1 - \pi_{b,c}(x) = \begin{cases} 1, & \text{gdy } x \leq c - b \vee x \geq c + b \\ 1 - 2\left(\frac{x-c+b}{b}\right)^2, & \text{gdy } c - b < x < c - \frac{b}{2} \\ 2\left(\frac{x-c}{b}\right)^2, & \text{gdy } c - \frac{b}{2} < x \leq c + \frac{b}{2} \\ 1 - 2\left(\frac{x-c-b}{b}\right)^2, & \text{gdy } c + \frac{b}{2} < x < c + b \end{cases}$$



Rys. 11: Wykres funkcji przynależności klasy u

2.3 Operacje na zbiorach rozmytych

Poniżej zostały przedstawione podstawowe operacje na zbiorach rozmytych. Więcej informacji na ich temat możemy znaleźć w pozycjach [6, 7].

2.3.1 Działania

Do najważniejszych typów działań na zbiorach rozmytych należą przecięcie, suma, iloczyn kartezjański, różnica i dopełnienie. W praktyce jednak dwie pierwsze są szczególnie często wykorzystywane, dlatego zostaną omówione dokładniej. Są one opisywane przez klasy norm trójkątnych, których przykłady zostały opisane poniżej.

Definicja 14 *T–normą nazywamy funkcję*

$$T : [0,1]^2 \longrightarrow [0,1]$$

dla $a, b, c, d \in [0,1]$ spełniającą warunki:
1) funkcja T jest niemalejąca względem obu argumentów:

$$T(a,c) \leq T(b,d) \text{ dla } a \leq b \wedge c \leq d$$

2) funkcja T jest przemienna:

$$T(a,b) = T(b,a)$$

3) funkcja T jest łączna:

$$T\left(T(a,b),c\right) = T\left(a,T(b,c)\right)$$

4) funkcja T spełnia warunek brzegowy:

$$T(a,1) = a$$

Parametry a, b utożsamiamy z funkcjami przynależności zbiorów rozmytych A, B przez co otrzymujemy :

$$T(a,b) = \mu_A(x) \overset{T}{*} \mu_B(x)$$

Korzystając z warunku na łączność, można wyprowadzić wzory dla większej ilości zmiennych. Klasa ta odpowiada przecięciu zbiorów rozmytych. Do najpopularniejszych T–norm należą :

- iloczyn mnogościowy :
$$\mu_{A\cap B} = a \wedge b = \min(a,b)$$
- iloczyn algebraiczny :
$$\mu_{A\circ B} = a \cdot b$$
- iloczyn logiczny :
$$\mu_{A\otimes B} = \max(0, a+b-1)$$
- iloczyn drastyczny :
$$\mu_{A\dot{\vee}B} = \begin{cases} 0 & \text{gdy } a < 1 \wedge b < 1 \\ a, & \text{gdy } b = 1 \\ b, & \text{gdy } a = 1 \end{cases}$$
- T–norma Łukasiewicza :
$$T(a,b) = \max(0, a+b-1)$$
- T–norma Dubois i Prade'a :
$$\sigma_\alpha(a,b) = \frac{ab}{\max(a,b,\alpha)} \text{ dla } \alpha \in (0,1)$$
Możemy zauważyć że dla $\alpha \longrightarrow 0$ norma $\sigma_\alpha(a,b)$ zbiega do iloczynu mnogościowego, a dla $\alpha \longrightarrow 1$ zbiega do iloczynu algebraicznego.
- T–norma Yagera :
$$a \overset{YT}{*} b = 1 - \min\left(1, ((1-a)^p + (1-b)^p)^{\frac{1}{p}}\right) \text{ dla } p > 0$$
Dla $p = 1$ otrzymujemy iloczyn logiczny, dla $p \longrightarrow \infty$ iloczyn mnogościowy, zaś dla $p \longrightarrow 0$ iloczyn drastyczny.
- T–norma Hamachera :
$$a \overset{HT}{*} b = \frac{ab}{\gamma + (1-\gamma)(a+b-ab)} \text{ dla } \gamma \geq 0$$
Dla $\gamma = 1$ otrzymujemy iloczyn algebraiczny, zaś dla $\gamma \longrightarrow \infty$ iloczyn drastyczny.
- T–norma Sugeno :
$$a \overset{ST}{*} b = \max\left(0, (\lambda+1)(a+b-1) - \lambda \cdot a \cdot b\right) \text{ dla } \lambda \geq -1$$

Dla $\lambda = -1$ norma Sugeno przechodzi w iloczyn algebraiczny, dla $\lambda = 0$ staje się iloczynem logicznym, zaś dla $\lambda \longrightarrow \infty$ iloczynem drastycznym.

Definicja 15 *S–normą nazywamy funkcję*

$$S : [0,1]^2 \longrightarrow [0,1]$$

dla $a, b, c, d \in [0,1]$ *spełniającą warunki:*
1) funkcja S jest niemalejąca względem obu argumentów:

$$S(a,c) \leq S(b,d) \text{ dla } a \leq b \wedge c \leq d$$

2) funkcja S jest przemienna:

$$S(a,b) = S(b,a)$$

3) funkcja S jest łączna:

$$S\left(S(a,b),c\right) = S\left(a,S(b,c)\right)$$

4) funkcja S spełnia warunek brzegowy:

$$S(a,0) = a$$

Podobnie jak wcześniej, parametry a, b utożsamiamy z funkcjami przynależności zbiorów rozmytych A, B przez co otrzymujemy :

$$S(a,b) = \mu_A(x) \overset{S}{*} \mu_B(x)$$

Korzystając z warunku na łączność, można wyprowadzić wzory dla większej ilości zmiennych. Klasa ta odpowiada sumie zbiorów rozmytych Do najpopularniejszych S-norm należą

- suma mnogościowa :

$$\mu_{A \cup B} = a \vee b = \max(a,b)$$

- suma algebraiczna :

$$\mu_{A \square B} = a + b - a \cdot b$$

- suma logiczna :

$$\mu_{A \otimes B} = \min(1, a+b)$$

- suma drastyczna :

$$\mu_{A \dot{\vee} B} = \begin{cases} 1 & \text{gdy } a > 0 \wedge b > 0 \\ a, & \text{gdy } b = 0 \\ b, & \text{gdy } a = 0 \end{cases}$$

- S–norma Łukasiewicza :

$$S(a,b) = \min(a+b, 1)$$

- S–norma Dubois i Prade'a :

$$a \star b = \frac{a + b - ab - \min(a, b, 1 - \alpha)}{\max(1 - a, 1 - b, \alpha)} \text{ dla } \alpha \in (0, 1)$$

- S–norma Yagera :

$$a \overset{YS}{*} b = \min\left(1, ((a^p + b^p)^{\frac{1}{p}}\right) \text{ dla } p > 0$$

  Dla $p = 1$ otrzymujemy sumę logiczną, dla $p \longrightarrow \infty$ sumę mnogościową, zaś dla $p \longrightarrow$ sumę drastyczną.

- S–norma Hamachera :

$$a \overset{HS}{*} b = \frac{ab(\gamma - 2) + a + b}{ab(\gamma - 1) + 1} \text{ dla } \gamma \geq 0$$

  Dla $\gamma = 1$ otrzymujemy sumę algebraiczną, zaś dla $\gamma \longrightarrow \infty$ sumę drastyczną.

- S–norma Sugeno :

$$a \overset{SS}{*} b = \min\left(1, a + b + \lambda \cdot a \cdot b\right) \text{ dla } \lambda \geq -1$$

  Dla $\lambda = -1$ norma Sugeno przechodzi w sumę algebraiczną, dla $\lambda = 0$ staje się sumą logiczną, zaś dla $\lambda \longrightarrow \infty$ sumą drastyczną.

Iloczyn kartezjański podobnie jak większość działań posiada kilka definicji. W tej pracy ustalamy że będziemy posługiwać się następującą definicją

Definicja 16 *Iloczynem kartezjańskim zbiorów rozmytych $A_1 \subseteq X_1, A_2 \subseteq X_2, \ldots, A_n \subseteq X_n$ oznaczonym jako $A_1 \times A_2 \times \ldots \times A_n$ nazywamy :*

$$\mu_{A_1 \times A_2 \times \ldots \times A_n}(x_1, x_2, \ldots, x_n) = \min(\mu_{A_1}(x_1), \mu_{A_2}(x_2), \ldots, \mu_{A_n}(x_n))$$

Definicja 17 *Niech $A, B \subseteq X$. Wtedy różnicę zbiorów A i B oznaczaną jako $A \setminus B$ definiujemy jako :*

$$\mu_C(x) = \min(\mu_A(x), 1 - \mu_B(x))$$

Definicja 18 *Niech $A \subseteq X$. Wtedy dopełnieniem mnogościowym zbioru A na uniwersum X nazywamy*

$$\mu_{\neg A}(x) = 1 - \mu_A(x)$$

2.3.2 Relacje

Relacje ostre stwierdzają istnienie lub brak istnienia związku między elementami zbiorów. W przypadku relacji rozmytych, mamy możliwość stwierdzenia w jakim stopniu elementy zbiorów są ze sobą związane nieprecyzyjnymi sformułowaniami typu "mniej więcej równe", "znacznie mniejsze" itp.

Definicja 19 *Rozmytą relację binarną R między dwoma, niepustymi zbiorami ostrymi $X, Y \subseteq X$ nazywamy zbiór rozmyty określony na iloczynie kartezjańskim $X \times Y$:*

$$\mu_R : X \times Y \longrightarrow [0,1]$$

$$R \subseteq X \times Y = \{(x,y) \mid x \in X, y \in Y\}$$

Zgodnie z przyjętą konwencją , możemy zapisywać relację R jako

$$R = \sum_{X \times Y} \mu_R(x,y)|(x,y) \tag{4}$$

$$R = \int_{X \times Y} \mu_R(x,y)|(x,y) \tag{5}$$

W przypadku zbiorów skończonych $X = \{x_1, x_2, \ldots, x_n\}$, $Y = \{y_1, y_2, \ldots, y_m\}$ relację rozmytą możemy reprezentować macierzą rozmytą

$$R = \begin{bmatrix} \mu_R(x_1,y_1) & \mu_R(x_1,y_2) & \ldots & \mu_R(x_1,y_m) \\ \mu_R(x_2,y_1) & \mu_R(x_2,y_2) & \ldots & \mu_R(x_2,y_m) \\ \ldots & \ldots & \ldots & \ldots \\ \mu_R(x_n,y_1) & \mu_R(x_n,y_2) & \ldots & \mu_R(x_n,y_m) \end{bmatrix} \tag{6}$$

Wartym podkreślenia jest fakt, że relacja rozmyta jest zbiorem rozmytym (przy uwzględnieniu uproszczenia), przez co nadal możemy stosować wprowadzone działania na zbiorach rozmytych.

Definicja 20 *Złożeniem typu sup–T relacji rozmytych $R \subseteq X \times Y$ i $S \subseteq Y \times Z$ nazwiemy relację rozmytą $R \circ S \subseteq X \times Z$ o funkcji przynależności*

$$\mu_{R \circ S}(x,z) = \sup_{y \in Y} \{\mu_R(x,y) \overset{T}{*} \mu_S(y,z)\}$$

Definicja 21 *Złożeniem typu inf–S relacji rozmytych $R \subseteq X \times Y$ i $S \subseteq Y \times Z$ nazwiemy relację rozmytą $R \square S \subseteq X \times Z$ o funkcji przynależności*

$$\mu_{R \square S}(x,z) = \inf_{y \in Y} \{\mu_R(x,y) \overset{S}{*} \mu_S(y,z)\}$$

Definicja 22 *Kompozycją typu sup–T zbioru rozmytego $A \subseteq X$ i relacji rozmytej $R \subseteq X \times Y$ oznaczaną jako $B = A \circ R$, gdzie $B \subseteq Y$ jest zbiorem wynikowym nazywamy*

$$\mu_B(y) = \sup_{x \in X} \{\mu_A(x) \overset{T}{*} \mu_R(x,y)\}$$

Definicja 23 *Kompozycją typu inf–S zbioru rozmytego $A \subseteq X$ i relacji rozmytej $R \subseteq X \times Y$*

oznaczaną jako $B = A\Box R$, gdzie $B \subseteq Y$ jest zbiorem wynikowym nazywamy

$$\mu_B(y) = \inf_{x \in X} \{\mu_A(x) \overset{S}{*} \mu_R(x, y)\}$$

2.4 Logika rozmyta i rozszerzenia z logiki wielowartościowej

W logice klasycznej dysponujemy dwuwartościowym systemem oceny prawdziwości zdań. Dlatego możemy stwierdzić tylko o całkowitej prawdziwości lub nieprawdziwości danego stwierdzenia.W przypadku logiki wielowartościowej mamy stany pośrednie między prawdą i fałszem, co zapewnia bardziej elastyczne podejście.Przykładami logiki wielowartościowej są np: logika wielowartościowa Łukasiewicza, system Posta czy zaproponowana przez Loftiego Zadeha logika rozmyta operująca na wartościach rzeczywistych, która została wykorzystana w poniższej pracy.
Zaczynam od zdefiniowania operatorów koniunkcji, alternatywy, implikacji, negacji i równoważności, przy czym zostaną podane przykładowe, najpopularniejsze próby definicji na podstawie pozycji [6]. Niech α, β oznaczają rozmyte zmienne lingwistyczne, np: "(Dzisiaj jest) ciepło", "(Woda jest) trochę za chłodna" itp. Oznaczmy przez $f(\alpha), f(\beta)$ dowolne zbiory rozmyte reprezentujące rozmyte wartości logiczne zmiennych α, β. Wtedy otrzymujemy w wyniku operacji zbiory wynikowe o poniższych funkcjach przynależności.

2.4.1 Operator koniunkcji

Definicja 24 *Definicją koniunkcji w sensie logiki Łukasiewicza $\aleph_1$ nazywamy*

$$\mu_{f(\alpha \wedge \beta)}(x) = \sup_{i,j \in [0,1],\ \dot{z}e\ x\ =\ \min(i,j)} \{\min\left(\mu_{f(\alpha)}(i), \mu_{f(\beta)}(j)\right)\}$$

Definicja 25 *Definicją koniunkcji w sensie T–normy nazywamy*

$$\mu_{f(\alpha \wedge \beta)}(x) = T\left(\mu_{f(\alpha)}(x), \mu_{f(\beta)}(x)\right)$$

2.4.2 Operator alternatywy

Definicja 26 *Definicją alternatywy na podstawie logiki Łukasiewicza $\aleph_1$ nazywamy*

$$\mu_{f(\alpha \vee \beta)}(x) = \sup_{i,j \in [0,1],\ \dot{z}e\ x\ =\ \max(i,j)} \{\min\left(\mu_{f(\alpha)}(i), \mu_{f(\beta)}(j)\right)\}$$

Definicja 27 *Definicją alternatywy na podstawie S–normy nazywamy*

$$\mu_{f(\alpha \vee \beta)}(x) = S\left(\mu_{f(\alpha)}(x), \mu_{f(\beta)}(x)\right)$$

2.4.3 Operator implikacji

Definicja 28 *Implikacją rozmytą nazwiemy funkcję :*

$$I : [0,1]^2 \longrightarrow [0,1]$$

dla $a_1, a_2, a_3 \in [0,1]$ *spełniającą warunki*
1) Jeżeli $a_1 \leq a_3$, *to* $I(a_1, a_2) \geq I(a_3, a_2)$
2) Jeżeli $a_2 \leq a_3$, *to* $I(a_1, a_2) \leq I(a_1, a_3)$
3) $I(0, a_2) = 1 \ \forall a_2 \in [0,1]$
4) $I(a_1, 1) = 1 \ \forall a_1 \in [0,1]$
5) $I(1,0) = 0$

Implikację oznaczamy także jako $A \longrightarrow B$ lub $A \Rightarrow B$,, a jej funkcję przynależności jako $\mu_{A \longrightarrow B}(x,y)$ lub jako $\mu_{A \Rightarrow B}(x,y)$ Duża część implikacji zawiera dodatkowe warunki, które decydują o ich działaniu. Przykłady implikacji rozmytych :

- Kleene-Dienes (binarna) :
$$\max\{1-a, b\}$$
- Łukasiewicz :
$$\min\{1, 1-a+b\}$$
- Reichenbach :
$$1-a+a\cdot b$$
- Fodor :
$$\begin{cases} 1, & \text{gdy } a \leq b \\ \max\{1-a, b\}, & \text{gdy } a > b \end{cases}$$
- Rescher :
$$\begin{cases} 1, & \text{gdy } a \leq b \\ 0, & \text{gdy } a > b \end{cases}$$
- Goguen :
$$\begin{cases} 1, & \text{gdy } a = 0 \\ \min\left\{1, \frac{b}{a}\right\}, & \text{gdy } a > 0 \end{cases}$$
- Gödel :
$$\begin{cases} 1, & \text{gdy } a \leq b \\ b, & \text{gdy } a > b \end{cases}$$
- Yager :
$$\begin{cases} 1, & \text{gdy } a = 0 \\ b^a, & \text{gdy } a > 0 \end{cases}$$
- Zadeh :
$$\max\{\min\{a, b\}, 1-a\}$$
- Willmott :
$$\min \begin{cases} \max\{1-a, b\} \\ \max\{a, 1-b, \min\{1-a, b\}\} \end{cases}$$

- Dubois-Prade :

$$\begin{cases} 1-a, & \text{gdy } b=0 \\ b, & \text{gdy } a=1 \\ 1, & \text{w.p.p} \end{cases}$$

Przy czym wartości a, b utożsamiamy z funkcjami $\mu_A(x), \mu_B(y)$
Bardzo często wykorzystywaną implikacją jest implikacja w sensie Mamdaniego – $T(a,b)$, jednak nie jest ona implikacją rozmytą w sensie definicji 28.
Innym przykładem implikacji jest implikacja wynikająca z rozszerzenia operacji zdaniotwórczych logiki $\aleph_1$ na przedmiot logiki rozmytej

Definicja 29 *Implikacją rozmytą w sensie logiki $\aleph_1$ nazywamy*

$$\mu_{f(\alpha\Rightarrow\beta)}(x) = \sup_{i,j\in[0,1],\ \dot{z}e\ x\ =\ \min(1,1-i+j)} \{\min\left(\mu_{f(\alpha)}(i), \mu_{f(\beta)}(j)\right)\}$$

2.4.4 Operator negacji

Definicja 30 *Operator negacji definiujemy jako*

$$\mu_{f(\neg\alpha)}(x) = \mu_{f(\alpha)}(1-x)$$

2.4.5 Operator równoważności

Definicja 31 *Operatorem równoważności definiujemy jako*

$$\mu_{f(\alpha\Longleftrightarrow\beta)}(x,y) = \mu_{f(\alpha\Longrightarrow\beta)}(x,y) \wedge \mu_{f(\beta\Longrightarrow\alpha)}(x,y)$$

2.5 Wnioskowanie rozmyte

Po zapoznaniu się z podstawami teorii zbiorów rozmytych oraz logiki, możemy przejść do zdefiniowania różnych typów oraz systemów wnioskowania. Zostały one opisane zgodnie z pozycją [7].

2.5.1 Rozmyte, uogólnione wnioskowanie *modus ponens*

Definicja 32 *Dla zbiorów rozmytych $A, A' \subseteq X$ i $B, B' \subseteq Y$ oraz zmiennych lingwistycznych α, β uogólnionym wnioskowaniem rozmytym typu modus ponens nazywamy system wnioskowania*

Przesłanka : α is A'

*Implikacja : **IF** α is A **THEN** β is B*

Wniosek : β is B'

By wyznaczyć zbiór B' korzystamy z kompozycji typu sup–T tj. $B' = A' \circ (A \Rightarrow B)$ w sensie definicji 22. W efekcie otrzymujemy

$$\mu_{B'}(y) = \sup_{x\in X}\{\mu_{A'}(x) \overset{T}{*} \mu_{A\Rightarrow B}(x,y)\} \tag{7}$$

Gdzie $\mu_{A\Rightarrow B}(x,y) = \mu_R(x,y),\ R \in X \times Y$. Możemy ponad to zauważyć, że dla $A' = A$ oraz $B' = B$ otrzymujemy klasyczną regułę *modus ponens*.

2.5.2 Rozmyte, uogólnione wnioskowanie *modus tollens*

Definicja 33 *Dla zbiorów rozmytych $A, A' \subseteq X$ i $B, B' \subseteq Y$ oraz zmiennych lingwistycznych α, β uogólnionym wnioskowaniem rozmytym typu modus tollens nazywamy system wnioskowania :*

$$\textit{Przesłanka : } \beta \textit{ is } B'$$

$$\textit{Implikacja : } \textbf{\textit{IF}}\ \alpha \textit{ is } A\ \textbf{\textit{THEN}}\ \beta \textit{ is } B$$

$$\textit{Wniosek : } \alpha \textit{ is } A'$$

By wyznaczyć zbiór A' korzystamy z kompozycji typu sup–T tj. $A' = (A \Rightarrow B) \circ B'$ w sensie definicji 22. W efekcie otrzymujemy

$$\mu_{A'}(x) = \sup_{y \in Y} \{\mu_{A\Rightarrow B}(x,y) \overset{T}{*} \mu_{B'}(y)\} \tag{8}$$

Gdzie $\mu_{A\Rightarrow B}(x,y) = \mu_R(x,y),\ R \in X \times Y$ Możemy ponad to zauważyć, że dla $A' = \bar{A}$ oraz $B' = \bar{B}$ otrzymujemy klasyczną regułę *modus tollens*.

2.5.3 Baza reguł i blok wnioskowania

Niech $R^{(k)}, k = 1, 2, \ldots, N$ oznacza bazę rozmytych reguł, a N zaś jest ich liczbą. Załóżmy że
1) $A_i^k \subseteq \boldsymbol{X}_i \subset \boldsymbol{R},\ i = 1, 2, \ldots, n$ – zbiory wejściowe dla danych atrybutów i reguły
2) $B^k \subseteq \boldsymbol{Y} \subset \boldsymbol{R}$ – zbiór wyjściowy dla danej reguły
3) $[x_1, x_2, \ldots, x_n]^T = \boldsymbol{x} \in \boldsymbol{X} = \in \boldsymbol{X}_1 \times \boldsymbol{X}_2 \times \ldots \times \boldsymbol{X}_n$ – wektor wejściowy
4) $y \in \boldsymbol{Y}$ – skalarna wartość wyjściowa
5) $A^k = A_1^k \times A_2^k \times \ldots \times A_n^k$ – iloczyn kartezjański zbiorów rozmytych względem danej reguły
6) $R^{(k)} \subset \boldsymbol{X} \times \boldsymbol{Y}\ :\ A^k \Rightarrow B^k,\ k = 1, 2, \ldots N$ – relacja rozmyta będąca funkcją przynależności implikacji rozmytej
7)$\mu_{R^{(k)}}(\boldsymbol{x}, y) = \mu_{A^k \Rightarrow B^k}(\boldsymbol{x}, y)$

Zakładając że reguły są połączone spójnikami **OR** definiujemy

Definicja 34 *Modelem klasycznym wnioskowania (dla implikacji w sensie Mamdaniego, zwaną też modelem Mamdaniego) nazwiemy*

$$R^{(k)}\ :\ \textbf{\textit{IF}}\ x_1 \textit{ is } A_1^k\ \textbf{\textit{AND}}\ \ x_2 \textit{ is } A_2^k\ \textbf{\textit{AND}}\ \ldots\ \textbf{\textit{AND}}\ \ x_n \textit{ is } A_n^k\ \textbf{\textit{THEN}}\ y_k \textit{ is } B^k$$

Możemy wtedy zdefiniować

- Na wyjściu otrzymujemy N zbiorów rozmytych $\bar{B}^k \subseteq \boldsymbol{Y}$ z uogólnioną, rozmytą regułą *modus*

ponens

$$\bar{B}^{(k)} = A' \circ \left(A^k \Rightarrow B^k\right),\ k = 1, 2, \ldots N \tag{9}$$

$$\mu_{\bar{B}^{(k)}}(y) = \sup_{\boldsymbol{x} \in \boldsymbol{X}} \{\mu_{A'}(\boldsymbol{x}) \overset{T}{*} \mu_{A^k \Rightarrow B^k}(\boldsymbol{x}, y)\} \tag{10}$$

- Na wyjściu otrzymujemy jeden zbiór rozmyty $B' \subseteq \boldsymbol{Y}$:

$$B' = \bigcup_{k=1}^{N} A' \circ \left(A^k \Rightarrow B^k\right) \tag{11}$$

$$\mu_{B'}(y) = \sum_{k=1}^{N} \mu_{\bar{B}^{(k)}}(y) \tag{12}$$

Definicja 35 *Modelem wnioskowania Takagi–Sugeno (lub Takagi–Sugeno–Kanga) nazwiemy*

$$R^{(k)}\ :\ \textbf{\textit{IF}}\ x_1\ \textit{is}\ A_1^k\ \textbf{\textit{AND}}\ \ x_2\ \textit{is}\ A_2^k\ \textbf{\textit{AND}}\ \ldots\ \textbf{\textit{AND}}\ \ x_n\ \textit{is}\ A_n^k\ \textbf{\textit{THEN}}\ y_k = f_k(x_1, x_2, \ldots, x_n)$$

Wtedy jako wartość wyjściową y przyjmujemy

$$\alpha_k = T\left(\mu_{k,1}(x_1), \mu_{k,2}(x_2), \ldots, \mu_{k,n}(x_n)\right) \tag{13}$$

$$y = \frac{\sum_{k=1}^{N} y_k \cdot \alpha_k}{\sum_{k=1}^{N} \alpha_k} \tag{14}$$

2.6 Klasyczne systemy wnioskowania

2.6.1 Rozmywanie (fuzyfikacja) wartości wejściowych

Niech $\boldsymbol{x} = [\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}]^T \in \boldsymbol{X} = \boldsymbol{X}_1 \times \boldsymbol{X}_2 \times \ldots \times \boldsymbol{X}_n$ oznacza wektor wartości wejściowych, oraz $A' \subseteq \boldsymbol{X}$

Definicja 36 *Rozmywaniem typu singleton nazwiemy działanie opisane wzorem*

$$\mu_{A'}(\boldsymbol{x}) = \delta\left(\boldsymbol{x} - \overline{\boldsymbol{x}}\right) = \begin{cases} 1 & gdy\ \boldsymbol{x} = \overline{\boldsymbol{x}} \\ 0, & gdy\ \boldsymbol{x} \neq \overline{\boldsymbol{x}} \end{cases}$$

Definicja 37 *Rozmywaniem typu gaussowskiego nazwiemy działanie typu*

$$\mu_{A'}(\boldsymbol{x}) = \exp\left(-\frac{\left(\boldsymbol{x} - \overline{\boldsymbol{x}}\right)^T \left(\boldsymbol{x} - \overline{\boldsymbol{x}}\right)}{\delta}\right),\ \delta > 0$$

2.6.2 Wyostrzanie (defuzyfikacja) wartości wyjściowych

Zadaniem wyostrzania jest przedstawienie jednej, ostrej wartości $\overline{y} \in \boldsymbol{Y}$, która definiuje wyjście systemu rozmytego. Poniższe metody wyostrzania zostały przedstawione w oparciu o pracę [7].

- Niech $\mu_{\bar{B}^{(k)}}(y),\ k = 1, 2, \ldots N$ reprezentują funkcje przynależności wyjściowych zbiorów rozmytych $\bar{B}^{(k)}$, wtedy definiujemy następujące metody wyostrzania wartości rozmytych

  Definicja 38 *Metodą* ***center average defuzzification*** *nazwiemy*

  $$\overline{y} = \frac{\sum_{k=1}^{N} \mu_{\bar{B}^{(k)}}\left(\overline{y}^k\right) \overline{y}^k}{\sum_{k=1}^{N} \mu_{\bar{B}^{(k)}}\left(\overline{y}^k\right)}$$

  gdzie $\overline{y}^k$ *jest takim punktem ze*

  $$\mu_{B^k}\left(\overline{y}^k\right) = \max_{y \in \boldsymbol{Y}} \left(\mu_{B^{(k)}}(y)\right)$$

  Definicja 39 *Metodą* ***center of sum defuzzification*** *nazwiemy*

  $$\overline{y} = \frac{\int_{\boldsymbol{Y}} y \sum_{k=1}^{N} \mu_{\bar{B}^{(k)}}(y) dy}{\int_{\boldsymbol{Y}} \sum_{k=1}^{N} \mu_{\bar{B}^{(k)}}(y) dy}$$

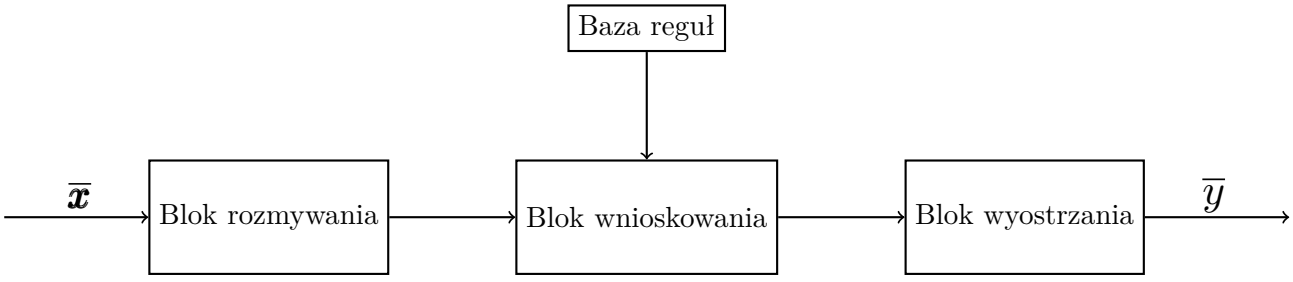
- W przypadku gdy na wyjściu systemu otrzymujemy jeden zbiór $B' \subseteq \boldsymbol{Y}$ możemy dodatkowo zdefiniować

  Definicja 40 *Metodą* ***maksimum funkcji przynależności*** *nazwiemy*

  $$\mu_{B'}\left(\overline{y}\right) = \sup_{y \in \boldsymbol{Y}} \mu_{B'}\left(y\right)$$

  przy założeniu że funkcja $\mu_{B'}\left(y\right)$ *jest unimodalna.*

2.6.3 Klasyczny system wnioskowania



Rys. 12: Schemat klasycznego systemu wnioskowania [7]

- **Blok rozmywania** – jego zadaniem jest rozmywanie wartości wejściowych systemu $\overline{\boldsymbol{x}}$
- **Baza reguł** – zawiera reguły na podstawie których wnioskuje system
- **Blok wnioskowania** – na podstawie dostępnych reguł i rozmytego wektora wejściowego zwraca (w ogólności rozmyte) zbiory stanowiące odpowiedź systemu

- **Blok wyostrzania** – w przypadku gdy blok wnioskowania zwrócił zbiory rozmyte, blok ten wyostrza wartości i zwraca odpowiedź ostrą $\overline{y}$

3 Algorytm genetyczny i strategia ewolucyjna

Opisywany zarys teorii algorytmów genetycznych i strategii ewolucyjnych przedstawiony został zgodnie z pozycjami [7, 22].

3.1 Algorytm genetyczny

Głównym założeniem algorytmów genetycznych jest poprawa rozwiązania zadania po przez systematyczną selekcję potencjalnych rozwiązań oraz wprowadzanie w nich losowych zmian. Jej cechą charakterystyczną jest to że rozwiązania te są reprezentowane przez ciągi binarne. Na przykład, jeśli problem ma rozwiązania w postaci wektora $v = [v_1, v_2, \dots v_n] \in \mathbb{R}^n$ to gen reprezentujący rozwiązanie może mieć postać

$$\Phi = \phi_1 \frown \phi_2 \dots \phi_n \tag{15}$$

gdzie ϕ_i oznacz reprezentację binarną v_i, a $\frown$ operator konkatenacji ciągów. Ogólny schemat algorytmu genetycznego wygląda wtedy następująco

1: wylosuj populację początkową rozwiązań P
2: dla iteracji $i = 1, 2 \dots N$:
  3: zastosuj operatory genetyczne
  3: oceń przystosowanie genów
  4: wybierz osobników do następnego pokolenia
5: **koniec** dla iteracji
Jako operatory genetyczne rozumiemy zazwyczaj

- krzyżowanie – dwa wybrane geny wymieniają się między sobą podciągami
- mutacje – na wybranym genie i na wybranej pozycji następuje negacja bitu

Wybór genów do populacji potomnej następuje zazwyczaj względem jakości przystosowania, choć nie jest to warunek konieczny. Jedną z najpopularniejszych technik jest tzw. metoda ruletki – prawdopodobieństwo wyboru genu do populacji potomnej jest wprost proporcjonalne do jakości przystosowania danego genu.

3.2 Strategia ewolucyjna

Strategia ewolucyjna jest rozwinięciem pomysłu algorytmu genetycznego, polegająca na zamianie ciągów binarnych w ciągi liczb rzeczywistych

$$\Phi = v \tag{16}$$

W takim przypadku operatory genetyczne działają na wartościach rzeczywistych przez co rozumiemy je zazwyczaj jako

- krzyżowanie – powstanie uśrednionego potomstwa z dwóch wybranych osobników, po przez uśrednienie wybranych pozycji

- mutacja – dodanie do wybranego osobnika na wybranej pozycji, liczby z określonego rozkładu

Jeśli rozwiązywany problem jest problemem o wartościach rzeczywistych, to strategia ewolucyjna jest bliższa fenotypowo oczekiwanej odpowiedzi, przez co ma zazwyczaj większe prawdopodobieństwo sukcesu niż algorytm genetyczny.

4 Rozmyty algorytm klasyfikacji

W tym rozdziale zostanie zaprezentowany implementowany algorytm, jego autorskie modyfikacje oraz algorytmy referencyjne, które zostały wybrane do części badawczej.

4.1 Algorytm Sorena

Jako system podlegający implementacji został wybrany algorytm z pracy [9], który został nazwany skrótowo algorytmem Sorena (od nazwiska jednego z twórców). Został on zrealizowany w tzw. *schemacie lokalnym* polegającym na wyborze konkretnego operatora agregacji oraz konkretnego zbioru rozmytego (w przeciwieństwie do schematu globalnego w którym jest użyte wiele funkcji przynależności oraz operatorów agregacji). Jedną z jego cech jest przypisanie każdemu atrybutowi pojedynczej funkcji przynależności.

4.1.1 Pseudokod algorytmu Sorena

Ucz
1: inicjalizuj parametry strategii ewolucyjnej `[params]`
2: inicjalizuj pusty zbiór reguł `rules`
3: dla każdej kategorii `c` $\in$ `Y` wykonaj:
  4: wyznacz podzbiór przykładów `X_c` $\subseteq$ `X_train` o danej kategorii `c`
  5: na podstawie `X_c` znajdź granice funkcji przynależności `x_min`, `x_max`
  6: stwórz binarny wektor kategorii `y_bin`, gdzie wartość 1 odpowiada pozycji w której pierwotny wektor klas `Y` skojarzony z `X_train` posiada kategorię `c` oraz 0 w przeciwnym przypadku
  7: uruchom `GeneticAlgortihm([params], X, y_bin, x_min, x_max)`, która zwróci osobnik reprezentujące nośniki `ind`
  8: przekształć `ind` w połączeniu z `x_min`,`x_max` oraz `c` w parametry nośników oraz odpowiadającą im kategorię tworząc `rule`
  9: dodaj `rule` do `rules`
10: **koniec** dla każdej kategorii

Testuj
1: inicjalizuj pusty wektor wyników `predictions`
2: dla każdego przykładu `x` $\in$ `X_test`:
  3: inicjalizuj pusty wektor zagregowanych wartości `outputs`
  4: dla każdej reguły `rule` $\in$ `rules`:
    5: oblicz `agg_r` = `aggregated_output([x], rule)`
    6: dodaj `agg_r` do `outputs`
  7: **koniec** dla każdej reguły
  8: znajdź indeks dla którego wartość w `outputs` jest największa i dodaj na tej podstawie przewidzianą kategorię do `predictions`
9: **koniec** dla każdego przykładu

4.2 Optymalizacja algorytmem strategii ewolucyjnej

Poniżej został przedstawiony pseudokod działania zbudowanego algorytmu strategii ewolucyjnej. W kolejnych podrozdziałach przedstawiono szczegółowo związek rozmytego systemu klasyfikacji oraz optymalizacji algorytmem strategii ewolucyjnej.

4.2.1 Pseudokod algorytmu strategii ewolucyjnej

1: inicjalizuj pusty wektor przechowujący najlepiej dostosowanego reprezentanta `hof`
2: inicjalizuj populację początkową:
  3: inicjalizuj pustą populację `population`
  4: dla $i = 1, 2, \ldots$ `size_of_initial_population`:
    5: inicjalizuj pusty wektor parametrów `param`
    6: dla $j = 1, 2, \ldots$ `number_of_attributes`:
      7: losuj parametr `b` z przedziału $[x\_min[j], x\_max[j]]$ z rozkładem jednostajnym
      8: losuj parametr `c` z przedziału $[b, x\_max[j]]$ z rozkładem jednostajnym
      9: dodaj `[b, c]` do `param`
    10: **koniec** dla `j`
    11: skonkatenuj wartości w `param` tworząc osobnika i dodaj je do `population`
  12: **koniec** dla `i`
13: **koniec** inicjalizuj populację początkową
14: dla $i = 1, 2, \ldots$ `liczba_epok`:
  15: mutuj, krzyżuj i rozmnażaj osobniki z `population` używając strategii `varOr` oraz uwzględniając funkcję `checkBounds`, tworząc `offspring`
  16: dla każdego osobnika w `offspring`:
    17: zamień osobniki na punkty nośników tworząc `rule`
    18: oblicz `agg_r = aggregated_output(X_train, rule)`
    19: oblicz RMSE między wektorem `y_bin` oraz `agg_r`
    20: przypisz obliczone RMSE osobnikowi
    21: zaktualizuj zbiór zawierający reprezentanta z najmniejszym RMSE `hof`
  22: **koniec** dla każdego osobnika
  23: wybierz określoną liczbę osobników z najmniejszym RMSE z `offspring` i przypisz do `population`
24: **koniec** dla `i`

aggregated_output
1: inicjalizuj pusty wektor wartości `output`
2: dla każdego przykładu `x` $\in$ `X`:
  3: inicjalizuj pusty wektor wartości `values`
  4: dla `i` $= 1, 2, \ldots$ `number_of_attributes`:
    5: oblicz wartość funkcji przynależności `val = pi_function(x[i], min[i], parrameters[i][0], parrameters[i][1], max[i])` i dodaj do `values`

6: **koniec** dla `i`
7: uruchom `aggregation_operator` z `values` i dodaj wynik do `output`
8: **koniec** dla każdego przykładu

check_bounds
1: inicjalizuj parametr δ
2: dla każdego obiektu `child` $\in$ `offspring`:
3: dla każdego `i` $= 1, 2, \ldots$ `number_of_attributes:`
4: jeśli `child[2i + 0] > max[i]`:
5: `child[2i + 0] = max[i] -` $\delta \cdot$`|max[i]|`
6: jeśli w przeciwnym razie `child[2i + 0] < min[i]`:
7: `child[2i + 0] = min[i] +` $\delta \cdot$`|min[i]|`
8: **koniec** jeśli
9: jeśli `child[2i + 1] < child[2i + 0]`:
10: `child[2i + 1] = child[2i + 0] +` $\delta \cdot$`|child[2i + 0]|`
11: jeśli w przeciwnym razie `child[2i + 1] > max[i]`:
12: `child[2i + 1] = max[i] -` $\delta \cdot$`|max[i]|`
13: **koniec** jeśli
14: **koniec** dla każdego `i`

4.3 Autorskie modyfikacje i założenia

- Algorytm optymalizacji genetycznej został zastąpiony strategią ewolucyjną (która operuje na danych liczbowych rzeczywistych, bliższych fenotypowo atrybutom wejściowym, co może rodzić nadzieję na lepsze rezultaty) – wymiana w punkcie 7. uczenia algorytmu Sorena `GeneticAlgorithm` na `EvolutionStrategy`

- Spodziewane jest osiągnięcie lepszych rezultatów dzięki wprowadzeniu nowego schematu losowania populacji początkowej – punkty 7, 8 w głównej pętli strategii ewolucyjnej z punktu 4.2.1 oraz linie 14, 15 w pliku `EvolutionStrategy.py`

- Wprowadzono możliwość ręcznego poszerzania granic funkcji przynależności znalezionych przez oryginalny algorytm – linia 5 w 4.3.1 oraz linia 44 w pliku `FuzzyAlgorithm.py`

- Zastosowano wstępne przetwarzanie danych wejściowych – funkcja `prepare_data` w pliku `DivideSet.py` (w przypadku systemu testującego implementowany algorytm wraz z algorytmami referencyjnymi) lub `Tools.py` (w przypadku samodzielnie implementowanego systemu)

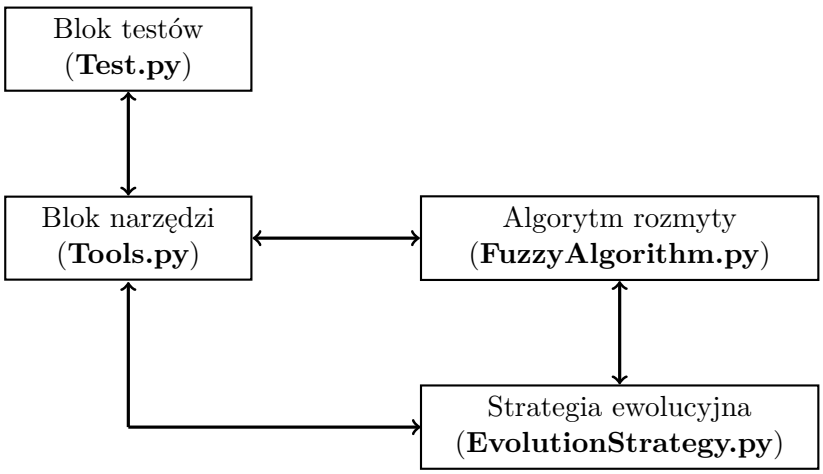
- Zdecydowano się przetrzymywanie najlepiej dostosowanego (pod względem uczonej miary) osobnika w trakcie wszystkich iteracji algorytmu strategii ewolucyjnej i jego zwrócenie przez ten algorytm na końcu procesu uczenia – tzw. *Hall of Fame* (*hof*), linie 1, 21 w 4.2.1 oraz linie 45, 60 w pliku `FuzzyAlgorithm.py`

- Jako selekcję został użyty algorytm `selBest` wybierający określoną ilość osobników z populacji, najlepiej dostosowanych pod względem uczonej miary – linia 23 w 4.2.1 oraz linia 70 w pliku `EvolutionStrategy.py`

4.4 Opis sytemu

Poniżej zostanie przedstawiony opis systemu który powstał z połączenia algorytmu Sorena oraz własnych pomysłów.

4.4.1 Schemat systemu



Rys. 13: Schemat zaimplementowanego systemu

- **Blok testów** – plik ze środowiskiem przeznaczonym dla użytkownika, w którym może przeprowadzać własne działania
- **Blok narzędzi** – plik z funkcjami pomocniczymi, współdzielonymi przez pozostałe bloki. Pełni również funkcję interfejsu z użytkownikiem oraz zawiera funkcję `prepare_data` przeprowadzającą wstępne przetwarzanie danych
- **Algorytm rozmyty** – plik z rozmytym systemem klasyfikującym
- **Strategia ewolucyjna** – plik z algorytmem strategii ewolucyjnej stworzony z użyciem biblioteki *DEAP*

4.4.2 Wstępne przetwarzanie danych

Na wstępną obróbkę danych składa się

- zastępowanie wartości nienumerycznych atrybutów, średnimi po danej kolumnie z dostępnych wartości liczbowych
- normalizacja danych w każdej z kolumn $\frac{x_i - \overline{x}}{\min(x) - \max(x)} \in [-1, 1]$ gdzie i jest numerem przykładu

- losowym podziale na zbiór treningowy i testowy o podanych wielkościach
- ewentualne nadpróbkowanie danych (możliwość włączenia i wyłączenia funkcjonalności) algorytmem **SMOTE** i permutacja

Jako metodę podziału wybrano **walidację Monte Carlo** (tzw. *random split*). Charakteryzuje się ona tym że wybór rekordów ze zbioru pierwotnego następuje bez zwracania (przez co losowane zbiory są rozłączne), a także tym że w kolejnych iteracjach każdy przykład znów z takim samym prawdopodobieństwem może znaleźć się w zbiorze treningowym albo testowym. W porównaniu z inną, powszechnie wykorzystywaną techniką zwaną kroswalidacją krzyżową (tzw. *k fold cross validation*) metoda Monte Carlo zapewnia mniejszą wariancję estymowanych parametrów kosztem większego obciążenia.
System pozwala także na nadpróbkowanie klas mniejszościowych w celu wyrównania przykładów różnych kategorii (do liczby liczby przykładów z klasy największej) algorytmem SMOTE [24, 25] (*Synthetic Minority Over-sampling Technique*) będącego połączeniem techniki centroidów z losowymi zmianami w dostępnych wektorach należących do klas mniejszościowych. W celu uniknięcia nadmiernego dopasowania nadpróbkowanie następuje tylko dla zbioru treningowego.

4.4.3 Opis działania algorytmu

Podstawowym założeniem algorytmu jest wyuczanie się osobno każdej kategorii, przy wykorzystaniu minimalizacji RMSE w trakcie działania strategii ewolucyjnej [22]. Strategia ta wykorzystuje schemat *varOr* który dla każdej iteracji $i = 1, 2, \ldots$ `size_of_offspring` stosuje w zależności od losowania, prawdopodobieństwa mutacji *mutpb* oraz krzyżowania *cxpb* jeden z operatorów

- z pewnym prawdopodobieństwem klonuje wybranego losowo z populacji osobnika a następnie jego kopię **mutuje** i dodaje do populacji potomnej
- z pewnym prawdopodobieństwem klonuje dwa wybrane losowo z populacji osobniki, a następnie ich kopie **krzyżuje** otrzymując dwójkę dzieci, z czego pierwsze jest przeznaczone do populacji potomnej, a drugie odrzucane
- **rozmnaża** po przez losowy wybór jednego z osobników i dodanie z pewnym prawdopodobieństwem jego kopii do populacji potomnej

Przy prawidłowo określonych warunkach powinno zachodzić $mutpb + cxpb \in [0, 1]$.
Jako mutacje został wybrany schemat *mutGaussian* dodający niezależnie do każdego segmentu osobnika z prawdopodobieństwem *indpb* liczbę z rozkładu $\mathcal{N}(\mu, \sigma)$. Z kolei jako procedura krzyżowania została wybrany schemat *cxSimulatedBinary* który wykorzystując zdefiniowany przez użytkownika parametr η oraz osobniki x_1, x_2 działa w następujący sposób

1: dla `i` $\in \{1, 2, \ldots 2 \cdot$ `number_of_attributes`$\}$:
  2: wylosuj przypadkową liczbę `rand` z przedziału $[0, 1]$
  3: jeśli $rand < 0.5$
    3: β = $2 \cdot rand$:

4: w przeciwnym przypadku:
5: $\beta = \frac{1}{2(1-rand)}$
5: koniec **jeśli**
6: $\beta := \beta^{\frac{1}{\eta+1}}$
7: `child_1` $= \frac{1}{2}\left((1+\beta)x_1[i] + (1-\beta)x_2[i]\right)$
8: `child_2` $= \frac{1}{2}\left((1-\beta)x_1[i] + (1+\beta)x_2[i]\right)$
9: **koniec** dla **i**

Każdy z osobników jest oceniany na podstawie uruchomienia systemu rozmytego i sprawdzeniu jak bardzo system się myli w wykrywaniu aktualnie uczonej kategorii c na podstawie obliczenia RMSE między wektorem y_bin reprezentującym przykłady aktualnie uczonej kategorii

$$y\_bin_c = \begin{cases} 1, \text{ gdy } y_i = c \\ 0, \text{ gdy } y_i \neq c \end{cases} \text{ dla } i = 1, 2, \ldots \text{liczba\_przykladow} \tag{17}$$

oraz wektorem agg_r reprezentującym wynik działania systemu. Wektor agg_r dla danej kategorii $c \in Y$ powstaje jako agregacja wartości funkcji przynależności, przy czym stosowane jest rozmywanie typu singleton dla wektora danych $\overline{x} = [\overline{x}_1, \overline{x}_2, \ldots \overline{x}_n]$

$$agg\_r_c = \lambda\left(\mu_{c,1}\left(\overline{x}_1\right), \mu_{c,2}\left(\overline{x}_2\right), \ldots \mu_{c,n}\left(\overline{x}_n\right)\right) \tag{18}$$

Funkcja λ jest operatorem odnośnie którego zakładamy tylko tyle że

$$\lambda : [0,1]^n \longrightarrow [0,1] \tag{19}$$

gdzie n jest liczbą atrybutów. W szczególności może to być T – norma lub S – norma. W pracy zdecydowano się na wybór operatora λ jako średniej arytmetycznej, ze względu na założenie o równouprawnieniu wszystkich atrybutów.
Decyzja o przydzieleniu etykiety przykładowi następuje na podstawie wyboru tej kategorii dla której wartość agg_r jest największa

$$\hat{c} = \arg\max_{k \in Y}\left(agg\_r_1, agg\_r_2, \ldots agg\_r_k, \ldots agg\_r_m\right) \tag{20}$$

gdzie $m = |Y|$
Zdecydowano się zastosować funkcję przynależności klasy π^*, która przyjmuje cztery parametry

- a - lewy kraniec nośnika
- d - lewy kraniec nośnika
- b, c - dwa pozostałe parametry

Punkt charakterystyczne nośnika muszą spełniać zależność $a \leq b \leq c \leq d$, co jest zadaniem dla funkcji *check_Bounds* działającej podczas uczenia systemu. Ponad to, ponieważ parametry a, d

są z góry ustalone, optymalizacji podlegają tylko wartości b, c. Dlatego każdy osobnik możemy przedstawić w postaci konkatenacji wszystkich parametrów z atrybutów w ustalonym porządku

$$\Phi = [b_1, c_1, b_2, c_2, \ldots b_n, c_n] \tag{21}$$

gdzie ponownie n jest liczbą atrybutów.
Efektem działania algorytmu optymalizacji jest zwrócenie osobnika z najmniejszym RMSE jaki pojawił się kiedykolwiek wśród ewolucji populacji (jest to obiekt z tzw. *hof – Hall of Fame*).

4.5 Algorytmy referencyjne

Poniżej zostały zaprezentowane algorytmy referencyjne, które będą podlegały porównaniu z implementowanym systemem w części doświadczalnej pracy. Zdecydowano na wybór języka *R* oraz pakietu *frbs* [10].

4.5.1 Metoda Chi

Rozszerzenie metody Wangla i Mendla polegającej na

1. Równym podziale przestrzeni wejściowej i wyjściowej zmiennych na określoną liczbę podzbiorów
2. Wygenerowaniu reguł postaci *IF–THEN* pokrywających dane wejściowe na podstawie obliczenia wartości funkcji przynależności dla każdego dostępnego rekordu i każdej zmiennej lingwistycznej. Zostaje wybrana ta zmienna, której funkcja przynależności posiadała największą wartość dla danego rekordu, a cały proces jest powtarzany dla wszystkich przykładów.
3. Znalezieniu stopnia każdej reguły, po przez agregację stopni funkcji przynależności w poprzedniku i następniku przy użyciu operatora agregacji typu produkt.
4. Zbudowaniu ostatecznej bzy reguł przez usunięcie reguł nadmiarowych na podstawie wyliczonych stopni.

Metoda Chi zastępuje następniki kategoriami [11].

4.5.2 Metoda Ishibuchi & Nakashima

Rozbudowuje system regułowy o stopnie pewności (wagi) w następnikach reguł.Poprzedniki są generowane na podstawie metody *grid–type fuzzy partition*, a klasa w następniku jest definiowana jako klasa dominująca w podprzestrzeni poprzednika w każdej z reguł. Nowa klasa dla danych wejściowych jest wyznaczana jako klasa następnika o największej wartości na podstawie zagregowanych stopni funkcji przynależności poprzedników oraz stopni pewności generowanych na podstawie współczynnika w klasach następników [12].

5 Dobór narzędzi do rozwiązania zadania

W trakcie poszukiwań możliwych narzędzi przeznaczonych do realizacji celu pracy zostały przeanalizowane poniższe języki programowania oraz biblioteki pod kątem przydatności do ewentualnej implementacji.

5.1 Matlab

- **Fuzy Logic Toolbox**
  Pakiet zawierający funkcje, aplikację oraz bloki w programie Simulink do analizy, projektowania i symulacji systemów opartych o logikę rozmytą [13]. Możemy tworzyć w nim zarówno pełne systemy wnioskujące jak i w połączeniu z oprogramowaniem Simulink przeprowadzać symulację złożonych systemów.

5.2 C++

- **fuzzylite**
  Pakiet zawierający bibliotekę ze sterownikami opartymi o logikę rozmytą [14], przeznaczony pod różne systemy operacyjne. Został zaimplementowany przy wykorzystaniu programowania obiektowego. W zależności od trybu wykorzystania bibloteki wymaga definiowania różnych makr. Nie wymaga programów trzecich.

5.3 Java

- **jfuzzylite**
  Jest pakietem przeznaczonym do uruchamiania w środowisku Java, będąc odpowiednikiem fuzzylite w C++ [14].

- **jFuzzyLogic**
  Bardzo popularny pakiet, przeznaczony zarówno do badań jak i zastosowań przemysłowych. Stanowi najprawdopodobniej najbardziej kompletny tego typu zestaw w klasie pakietów Java [16].

- **xFuzzy**
  Jest pakietem stworzonym przez *Instituto de Microelectronica de Sevilla*, mającym w zamyśle integrować ze sobą liczne narzędzia wykorzystywane przez użytkownika w trakcie planowania, projektowania i symulacji układów rozmytych [17]. Korzysta z języka XFL pozwalającego na definiowanie i interpretowanie wyrażeń rozmytych oraz definiowanie w sposób swobodny przez uzytkownika operatorów rozmytych. Zawiera równierz narzędzia typu CAD.

5.4 Python

- **pyfuzzy**
  Stosunkowo prosty pakiet, przeznaczony do wykorzystania przy logice rozmytej [18].

- **gfuzzy**
  Pakiet przeznaczony do tworzenia wnioskowania opartego o logikę rozmytą, wykorzystujący biblioteki trzecie do konfiguracji [19].

- **scikit-fuzzy**
  Pakiet przeznaczony do pomocy w implementacji logiki rozmytej [20], będący jednocześnie częścią większego pakietu narzędzi naukowych : *SciKit*. Dzięki niezłej, konkretnej dokumentacji oraz łatwości w użytkowaniu zyskał w niedługim okresie na popularności.

5.5 R

- **frbs**
  Bogaty pakiet posiadający wiele implementacji algorytmów opartych o systemy rozmyto – regułowe [10]. Dzięki dostępnemu wachlarzowi rozwiązań, są możliwe zarówno zadania klasyfikacji jak i aproksymacji.

- **FuzzyR**
  Pakiet posiadający w swoim wnętrzu głównie systemy adaptacyjno–neuronowo–rozmyte. Oferuje również interfejs graficzny [21].

5.6 Porównanie

Język	Pakiet	Użyteczność	Łatwość użytkowania	Inne
Matlab	Fuzzy Logic Toolbox	Duża	Duża	Pakiet komercyjny dobra dokumentacja
C++	fuzzylite	Duża	Średnia	Pakiet wolny dobra dokumentacja
Java	jfuzzylite	Duża	Duża	Pakiet wolny
Java	jFuzzyLogic	Duża	Duża	Pakiet wolny Duża popularność dobra dokumentacja
Java	xFuzzy	Duża	Duża	Program wolny
Python	pyfuzzy	Średnia	Duża	Pakiet wolny mała popularność słaba dokumentacja
Python	gfuzzy	Średnia	Duża	Pakiet wolny mała popularność bardzo słaba dokumentacja
Python	scikit-fuzzy	Duża	Duża	Pakiet wolny duża popularność dobra dokumentacja
R	frbs	Duża	Średnia	Pakiet wolny duża popularność dobra dokumentacja
R	FuzzyR	Średnia	Średnia	Pakiet wolny mała popularność dobra dokumentacja

Tab. 1: Porównanie bibliotek do układów rozmytych w różnych językach

5.7 Konkluzja

W świetle przytoczonych opisów i subiektywnej oceny zdecydowano, że żaden z wymienionych pakietów nie spełnia wymaganych kryteriów. Głównym problem okazała się niewystarczająca adaptowalność gotowych rozwiązań do potrzeba pracy. Dlatego zdecydowano się ostatecznie na implementację autorską algorytmu rozmytego oraz użycie gotowej biblioteki do optymalizacji systemu.
Ustalono także że najlepszym wyborem do zaimplementowania przedstawionego systemu w poniższej pracy będzie język Python w wersji 3.6. Dużą zaletą języka Python jest prostota użytkowania, mnogość różnych bibliotek, dobra dokumentacja oraz możliwość łatwego łączenia go z innymi językami (np: pisanie sekcji krytycznych w C, czy powstanie nowego języka Jython). W ostatnim czasie mogliśmy zaobserwować również duży wzrost popularności na rynku pracy rzeczonego języka, co pozwala przypuszczać o jego perspektywiczności i zapotrzebowaniu. Posiada on jednak przede wszystkim bogatą ofertę pakietów, co ułatwiło zadanie znalezienia odpowiedniej, już zaimplementowanej strategi ewolucyjnej. W tym celu została wybrana biblioteka *DEAP* [23].

6 Badania i wnioski do wyników

W tym dziale zostaną przedstawione wyniki badań oraz ich interpretacje dla testowanych systemów, z uwzględnieniem prób modyfikacji niektórych parametrów algorytmów.

6.1 Procedury eksperymentów

6.1.1 Dobór parametrów i założenia

Wszystkie parametry zostały ustalone w sposób heurystyczny, wynikający z prób oraz doświadczenia. Zapewniają one dostatecznie dobre własności badanych systemów do przeprowadzenia doświadczeń i porównań.

- Wielkość zbioru treningowego ustalono na *70 %*, zaś testowego na *30 %* liczby elementów w zbiorze podlegającym badaniom.
- Atrybuty systemów muszą być wartościami liczbowymi (poza kategorią, która dla zaimplementowanego systemu może być dowolna, a dla algorytmów w języku R musi być liczbą naturalną)
- Ze względu na wymogi systemów z języka R, zdecydowano się na zmianę sposobu skalowania na $\frac{x-\min(x)}{\max(x)-\min(x)} \in [0,1]$
- W celu zapewnienia jak najlepszej rozróżnialności klas zdecydowano się na zastosowanie algorytmu SMOTE podczas badań
- W zaimplementowanym systemie została wprowadzona możliwość ręcznej zmiany wyszukanych granic funkcji przynależności obliczanych dla danego atrybutu względem kategorii przykładów treningowych. W trakcie eksperymentów zdecydowano się na poszerzenie znalezionych granic o stałą wartość równą *0,01*. Taką samą wartość wybrano do skalowania granic w systemie ewolucyjnym w przypadku gdy atrybut osobnika wyszedł poza rozważany przedział.
- Liczbę iteracji systemu ewolucyjnego ustalono na *500*
- Wielkość populacji początkowej ustalono na *25* osobników
- Liczba osobników końcowych w każdej iteracji została ustalona na *50*
- Do kolejnego pokolenia przechodzi *25* najlepiej przystosowanych osobników
- przez wszystkie pokolenia system trzyma jednego, najlepiej przystosowanego na zbiorze treningowym (w tzw. *Hall of Fame*) osobnika. Może być on zastąpiony innym tylko wtedy gdy nowy przedstawiciel posiada mniejszą wartość RMSE.
- Prawdopodobieństwo mutacji losowo wybranego osobnika z populacji ustalono na *10 %*
- Prawdopodobieństwo krzyżowania dwóch losowo wybranych osobników ustalono na *50 %*

- Każdy z atrybutów osobnika przeznaczonego do mutacji, mutuje z prawdopodobieństwem *10 %* po przez dodanie wylosowanej liczby z rozkładu Gaussa o parametrach $\mathcal{N}(\mu, \sigma)$, gdzie $\mu = 0$ oraz $\sigma = 0.1$
- Krzyżowanie osobników następuje w sposób opisany w części teoretycznej z parametrem $\eta = 1$
- Liczba powtórzeń całej procedury treningowo – testowej została ustalona na *10*
- Dla przykładów referencyjnych y_i oraz przewidzianych przez dany algorytm y_{pred_i}, których liczebność wynosi n (przy czym przykłady te mogą to być kategorie lub miary), badanymi parametrami były :

  1. **RMSE**
  $$RMSE = \sqrt{\frac{\sum_{i=1}^{n} \left(y_i - y_{pred_i}\right)^2}{n}} \tag{22}$$
  2. **celności predykcji (accuracy)**
  $$accuracy = \frac{m}{n} \text{ gdzie } m \text{ jest liczbą zgodnych stanów } y_i, y_{pred_i} \tag{23}$$
  3. **współczynnik Kappa Cohena** – ocenia jakość rozróżnialności klas przez algorytmy
  $$\kappa_C = \frac{p_o - p_e}{1 - p_e} \text{ gdzie } p_o \text{ to } \textbf{accuracy} \text{ a } p_e = \frac{1}{n^2} \sum_k i_{k,1} i_{k,2} \tag{24}$$
  oraz $i_{k,1}, i_{k,2}$ są ilością razy kiedy pierwsza i druga wyrocznia przewidziały kategorię k

  Przez wyrocznie rozumiemy wektor prawdziwych klas y_i oraz system rozmyty zwracający wektor przewidzianych klasy y_{pred_i}

  Ponad to w celu porównania zaimplementowanego algorytmu z gotowymi algorytmami w języku R wyznaczone zostały wartość minimalna, maksymalna oraz mediana (pozwalająca na separację od błędów grubych bez stosowania dodatkowych obliczeń) dla celności predykcji i współczynnik Kappa Cohena (z 10 iteracji). W przypadku algorytmu w języku Python liczone będą one na podstawie osobnika z *Hall of Fame* dla pięćsetnej iteracji strategii ewolucyjnej. W przypadku RMSE stosowane jest podwójne uśrednianie – pierwsze po liczbie kategorii a drugie po liczbie iteracji całego systemu równej 10.
- W danej iteracji dla zapewnienia jak najlepszego porównania, wylosowany zbiór treningowy i testowy jest taki sam dla wszystkich algorytmów. Zostało to zrealizowane po przez przesunięcie bloku odpowiadającego za wstępne przetwarzanie danych – funkcja `prepare_data` z zaimplementowanego algorytmu do głównej pętli obsługującej wszystkie systemy (plik `DivideSet.py`). Zostały one połączone językiem powłoki *Bash*
- W trakcie pracy z algorytmami zaimplementowanymi w języku R, stwierdzono że gaussowska funkcja przynależności będzie stanowić podstawę do testów w wymienionym środowisku (inne funkcje przynależności powodowały czasem w zależności od losowania i testowanego

zbioru błędy wewnętrzne systemu, na które użytkownik nie ma wpływu bez ingerencji w same algorytmy). Inne parametry pozostawiono jako domyślne.

6.1.2 Schemat ogólny procesu testów

Uproszczony schemat testowania systemów został przedstawiony poniżej

1: wykonaj pętlę `10` razy:
- **2**: wykonaj uzupełnianie brakujących wartości i skalowanie
- **3**: losuj przy użyciu `monte_carlo_split` zbiór treningowy o wielkości 70 % zbioru pierwotnego oraz zbiór testowy wielkości 30 %
- **4**: wykonaj nadpróbkowanie zbioru uczącego algorytmem *SMOTE*
- **5**: uruchom i przetestuj system `python_implementation`
- **6**: uruchom i przetestuj systemy `FRBCS.CHI` oraz `FRBCS.W`
- **7**: uśrednij statystyki systemu `python_implementation` dla każdej z `500` iteracji systemu ewolucyjnego względem kategorii

8: **koniec** wykonaj pętlę

9: znajdź wartość minimalną, maksymalną oraz medianę dla `10` iteracji algorytmu `python_implementation` dla `celności predykcji` i `współczynnika Kappa Cohena` względem ostatniej iteracji systemu ewolucyjnego

10: znajdź wartość minimalną, maksymalną oraz medianę dla `10` iteracji algorytmu `FRBCS.CHI` oraz `FRBCS.W` dla `celności predykcji` i `współczynnika Kappa Cohena`

W przypadku badania wpływu parametrów, algorytmy *Chi* oraz *Ishibuchi & Nakashima* są zastępowane odpowiednimi modyfikacjami zaimplementowanego systemu.

6.2 Plan badań

6.2.1 Badanie wpływu określenia granic nośników atrybutów względem kategorii oraz wpływu ograniczeń narzuconych na losowanie punktów nośników w populacji początkowej

- W oryginalnym algorytmie w celu wyznaczenia krańców nośnika dla danego atrybutu, wyznacza się wartość minimalną i maksymalną (lub ich poszerzone odpowiedniki jak w badanej implementacji) względem uczonej w danej chwili kategorii. W niniejszej pracy został zbadany wpływ takiej implementacji w porównaniu z wyznaczaniem krańców nośnika po wszystkich przykładach dla danego atrybutu. Badanie dotyczy linii 5. w algorytmie Sorena *Ucz* z punktu 4.1.1 oraz linii 43, 44 z pliku `FuzzyAlgorithm.py` – wyznaczanie granic nośników dla danego atrybutu może odbywać się względem danej kategorii, czyli po przykładach `X_c` (wtedy te granice będą różne w zależności od aktualnie wyuczanej kategorii) lub też po zbiorze wszystkich przykładów `X_train` (wtedy dla wszystkich wyuczanych kategorii, granice funkcji przynależności dla danego atrybutu będą takie same)

- Podobne badanie dotyczy losowania populacji początkowej. Jedną z wprowadzonych zmian

jest propozycja poprawienia losowania wartości atrybutów osobników w populacji startowej. Jak zostało to opisane we wstępie teoretycznym, dla danego atrybutu x_i zakres zmienności parametrów funkcji przynależności b_{x_i}, c_{x_i} jest ograniczony wartością minimalną i maksymalną $\min(x_i)$, $\max(x_i)$. Muszą one wtedy spełniać zależność $\min(x_i) \leq b_{x_i} \leq c_{x_i} \leq \max(x_i)$. W innym przypadku następuje korekta ich wartości w trakcie procesu uczenia. Poprawione losowanie populacji początkowej polega na wylosowaniu najpierw jednej wartości (przypuśćmy że jest to b_{x_i}) z przedziału $[\min(x_i), \max(x_i)]$ a następnie losowaniu $c_{x_i} \in [b_{x_i}, \max(x_i)]$. Powyższe podejście zestawione zostało z naiwnym losowaniem postaci $b_{x_i} \in [\min(x_i), \max(x_i)]$, $c_{x_i} \in [\min(x_i), \max(x_i)]$. Badanie tyczy się linii 7, 8 ze schematu głównego strategii ewolucyjnej z punktu 4.2.1, oraz linii 14, 15 z pliku `EvolutionStrategy.py`

Do obu eksperymentów został wybrany zbiór *Iris* posiadający trzy, zrównoważone klasy (po 50 przykładów każda) oraz niewielką liczbę atrybutów (4 atrybuty + kolumna klasyfikująca). Posiada on również bardzo dobre własności predykcyjne, dzięki czemu możemy spodziewać się wysokich wyników klasyfikatorów co może pomóc w interpretacji rezultatów.
Oba eksperymenty zostały przeprowadzone wspólnie (cztery równoległe implementacje – każda z badanych własności może być włączona lub wyłączona) rozszerzając system z punktu 4. o dodatkowe podsystemy, w efekcie czego mamy możliwość porównania wszystkich wariantów na wspólnych zbiorach treningowych i testowych.

6.2.2 Badanie osiągów zaimplementowanego systemu oraz wybranych algorytmów z języka R – zbiory danych

W pracy wykorzystano sześć zbiorów danych. Pierwszy z nich dotyczy klasyfikacji podgatunków irysa [26] i został wykorzystany przy ocenie wpływu zmian parametrów. Drugi zbiór dotyczy cukrzycy u Indian Pima [27], kolejny zaś chorób serca [28]. Dwa kolejne zbiory dotyczą białaczki oraz chorób płuc [29], przy czym ostatni wymieniony zbiór był badany również w wersji z wyrównanymi kategoriami metodą podpróbkowania.

Nazwa zbioru	Liczba przykładów	Liczba przykładów najliczniejszej kategorii	Liczba kategorii	Liczba atrybutów predykcyjnych
Cukrzyca	768	500	2	8
Serce	294	188	5	13
Białaczka	72	38	3	25
Rak płuc	203	139	5	25

Tab. 2: Porównania zbiorów danych

Uwagi:

- W przypadku zbiorów *Białaczka* i *Rak płuc* w celu optymalizacji czasu testów zdecydowano się na redukcję liczby atrybutów do 25 najbardziej znaczących.
- Dodatkowo w przypadku zbioru *Rak płuc* ze względu na to że algorytm SMOTE nie będzie działał poprawnie (liczba przykładów najmniejszej kategorii jest równa 6, co jest liczbą zbyt

małą w stosunku do liczby pozostałych kategorii równej 4), zdecydowano się na testowanie zbioru w dwóch wersjach – oryginalnej oraz z podpróbkowaniem do liczby przykładów kategorii najmniej licznej. Oba warianty, podobnie jak pozostałe zbiory zostały przetestowane równolegle, by zapewnić takie samo pokrycie zbiorami uczącymi oraz testowymi.

6.3 Wyniki badań

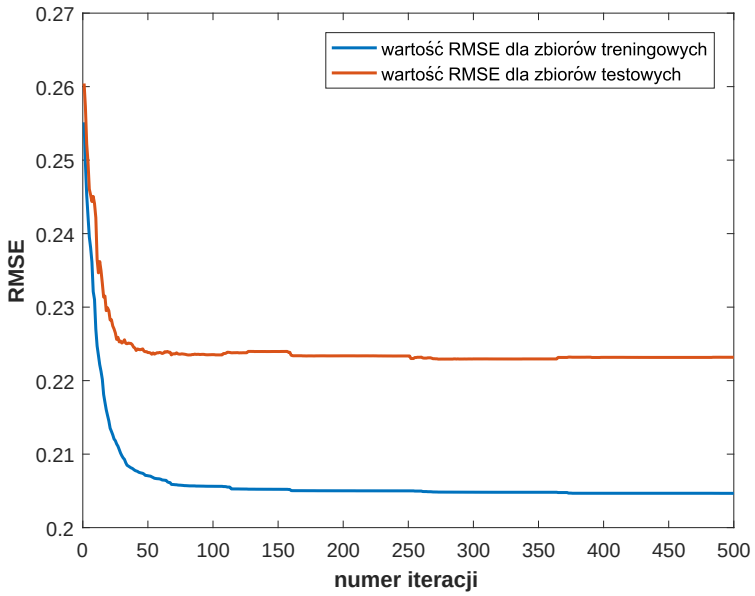
6.3.1 Badanie wpływu określania granic nośników oraz sposobu losowania populacji początkowej

W celu zapewnienia większej czytelności:

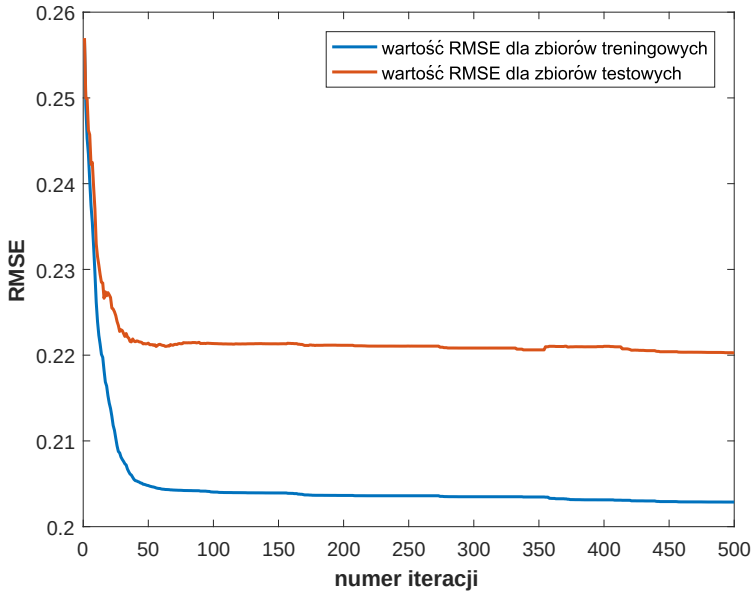
- Sposób określania granic nośników oznaczymy jako (*), przy czym jeśli zastosowano wyznaczanie względem danej kategorii, to przyjmujemy oznaczenie (*)ON, a dla wyznaczania względem wszystkich przykładów (*)OFF
- Sposób losowania populacji początkowej oznaczamy jako (**), przy czym jeśli losowanie parametrów następuje w sposób sekwencyjny (pierwszy z całego zakresu, drugi z zawężonego względem pierwszego) to oznaczamy to jako (**)ON, w przeciwnym przypadku gdy wszystkie parametry zostają wylosowane z tego samego przedziału, oznaczamy to jako (**)OFF

Zdecydowano się także na pozostawienie pewnych zachowań stochastycznych, dlatego ziarno funkcji **rand** nie zostało ustalone wspólne dla wszystkich przypadków i iteracji.

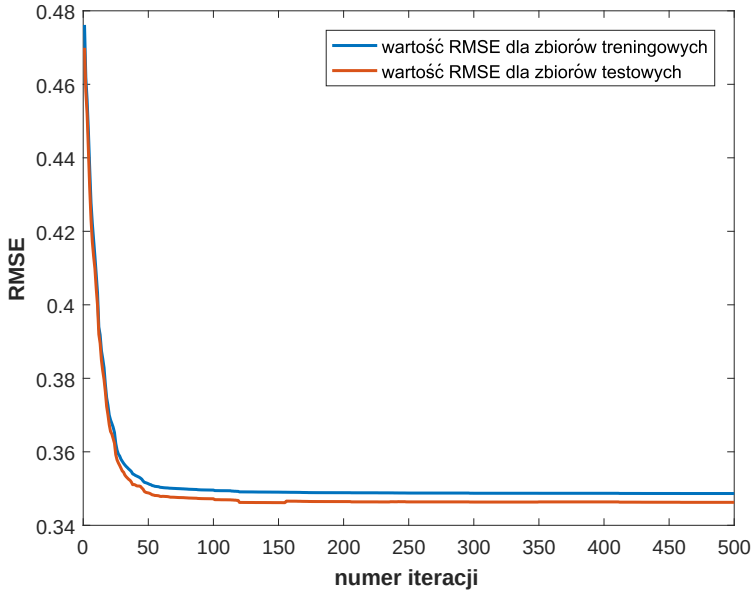
1. **Średnie RMSE**



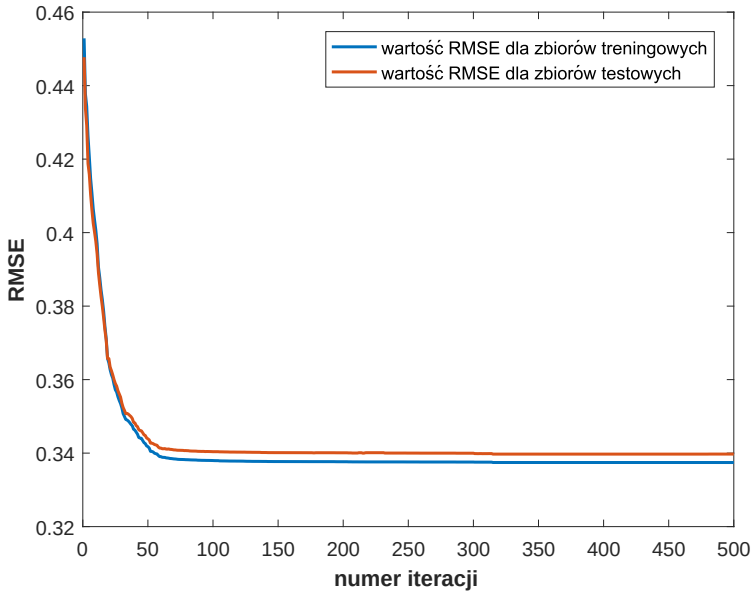
Rys. 14: Wykres RMSE dla (*)ON i (**)ON



Rys. 15: Wykres RMSE dla (*)ON i (**)OFF



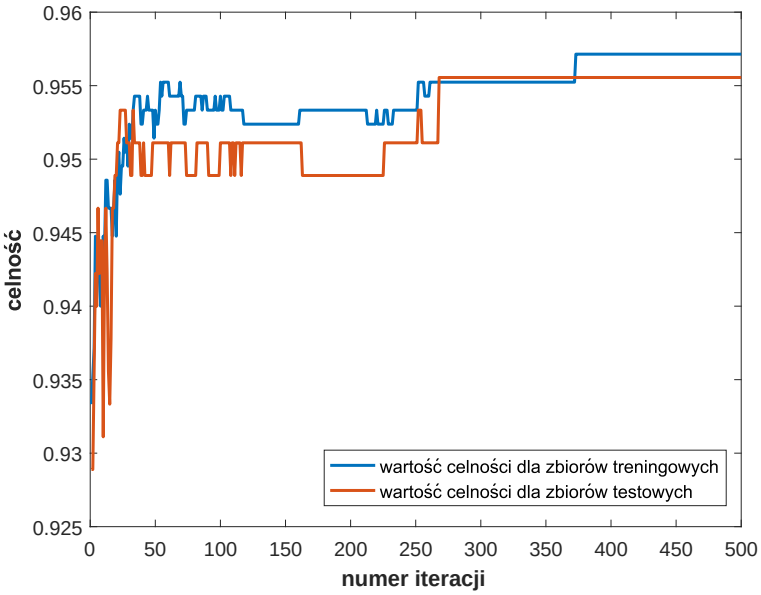
Rys. 16: Wykres RMSE dla (*)OFF i (**)ON



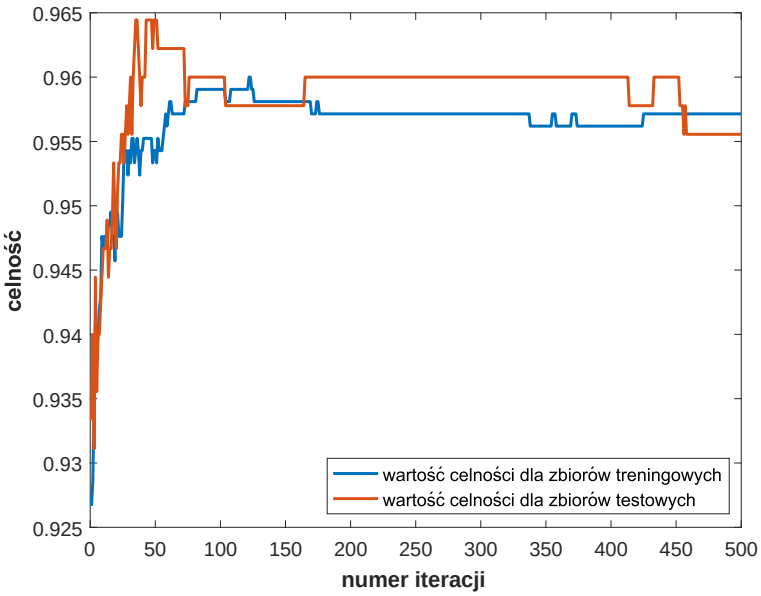
Rys. 17: Wykres RMSE dla (*)OFF i (**)OFF

Możemy zaobserwować że wpływ parametru (**) na przebieg RMSE jest minimalny i skutkuje nieznacznym obniżeniem poziomu RMSE. Parametr (*) okazuje się mieć decydujący wpływ na zachowanie wykresów. Jego włączenie powoduje dużo większy spadek wartości RMSE zarówno dla zbiorów treningowych jak i testowych. Możemy także zauważyć, że w przypadku jego deaktywacji wykresy dla uczenia i testowania przebiegają znacznie bliżej siebie. Na rysunku 16 możemy zaobserwować sytuację w której błąd uczenia jest większy od błędu dla testowania, nie jest to jednak zjawisko przeuczenia. Taka sytuacja może wynikać z wielu czynników jak na przykład przypadkowość czy specyfika zbioru danych przy czym z racji 10 powtórzeń eksperymentu, drugi czynnik jest najbardziej prawdopodobny. Sam dobór parametrów (*) i (**) ma małe szanse na spowodowanie takich rezultatów.

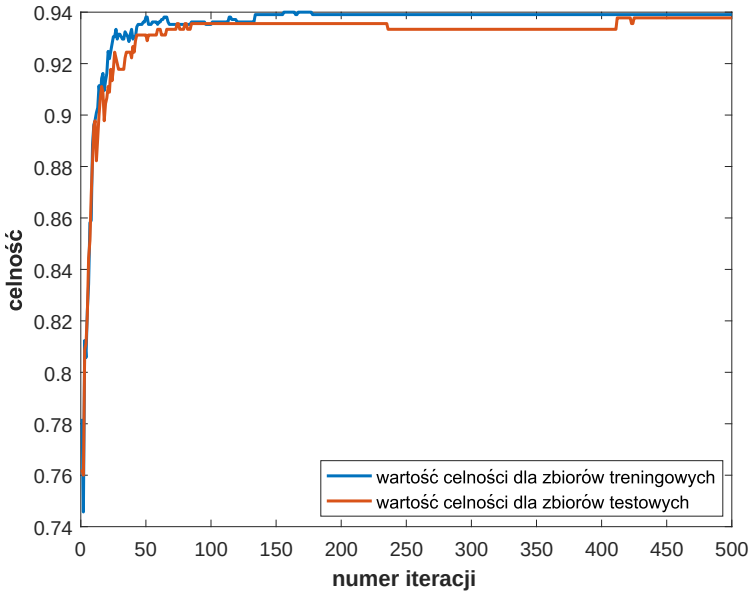
2. **Średnia celność predykcji**



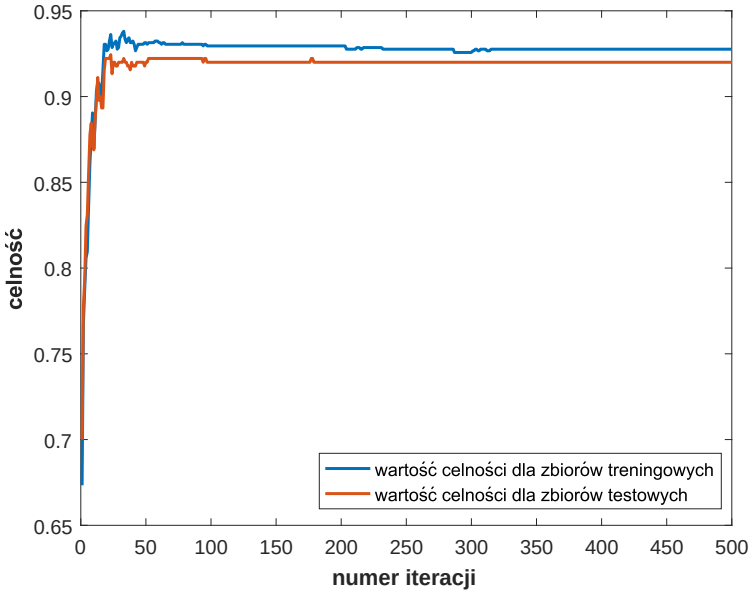
Rys. 18: Wykres celności predykcji dla (*)ON i (**)ON



Rys. 19: Wykres celności predykcji dla (*)ON i (**)OFF



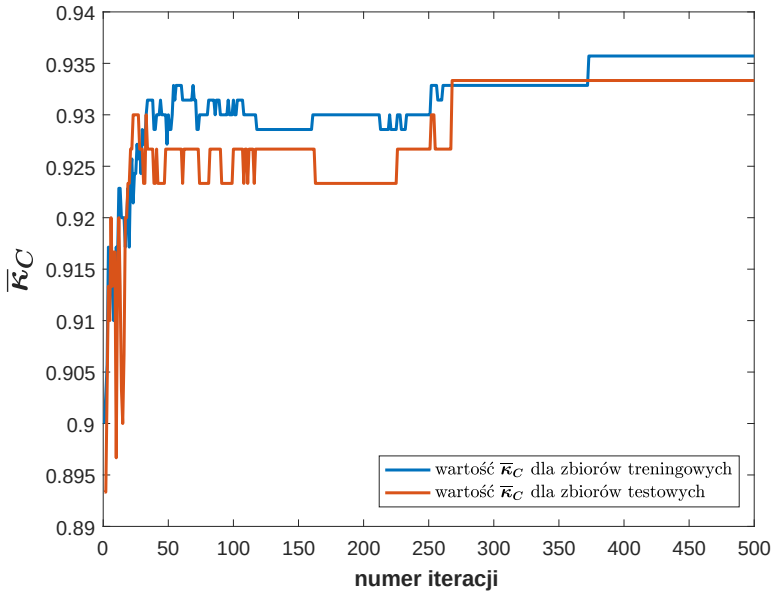
Rys. 20: Wykres celności predykcji dla (*)OFF i (**)ON



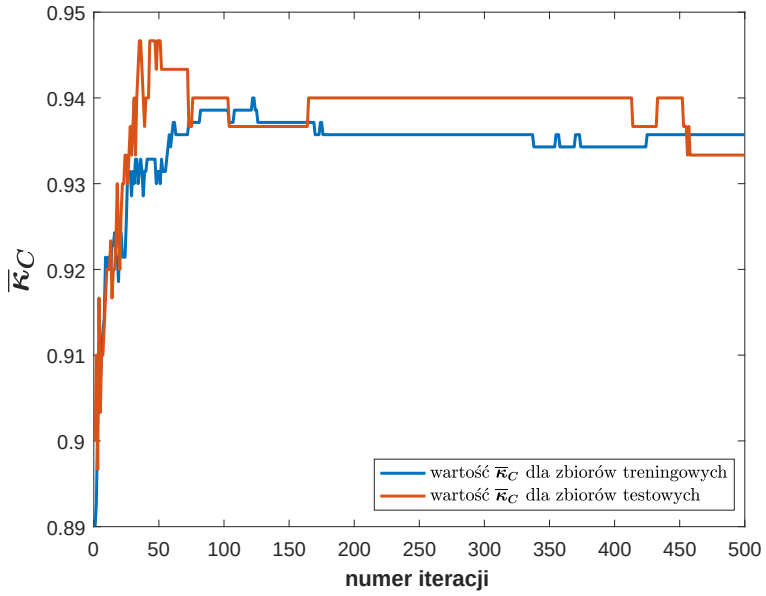
Rys. 21: Wykres celności predykcji dla (*)OFF i (**)OFF

Podobnie jak w przypadku RMSE zauważamy że wpływ parametru (**) jest stosunkowo mały (jeśli chodzi o zakres wartości celności predykcji) i największe znaczenie ma przy wyłączonej własności (*) (rysunki 20 oraz 21). Ponownie największe znaczenie ma parametr (*), którego wyłączenie powoduje rozpoczęcie wykresów dla obu rodzajów zbiorów od około 65 % – 75 %, oraz ich gładki przebieg. W przypadku włączenia parametru (*) wykresy rozpoczynają się od ok 93 % i mają bardzo skokowe przebiegi. Widzimy że włączenie własności (*) powoduje osiągnięcie lepszych rezultatów, jednak ze względu na to że, system rozpoczął swoją naukę z bardzo wysokiego pułapu powstaje efekt zaburzeń i brak jednoznacznego trendu, spowodowany osiągnięciem limitów wydajności predykcyjnej.

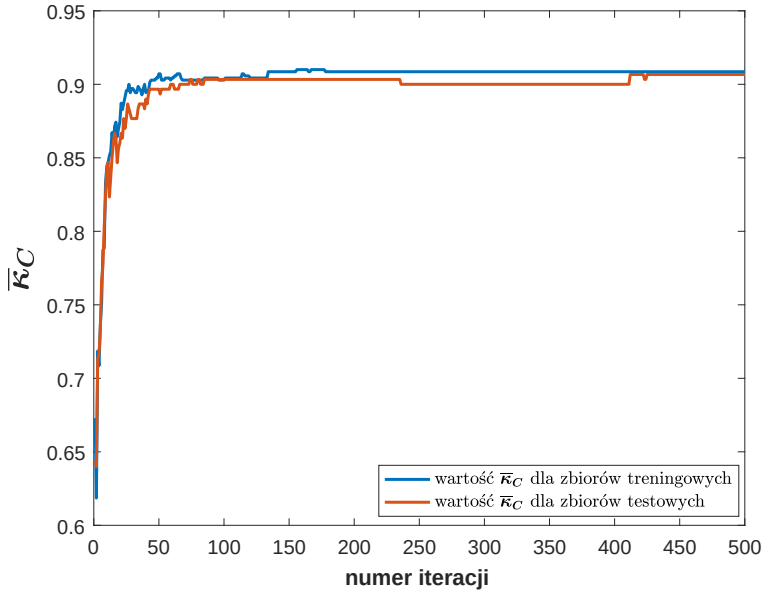
3. **Średni współczynnik Kappa – Cohena**



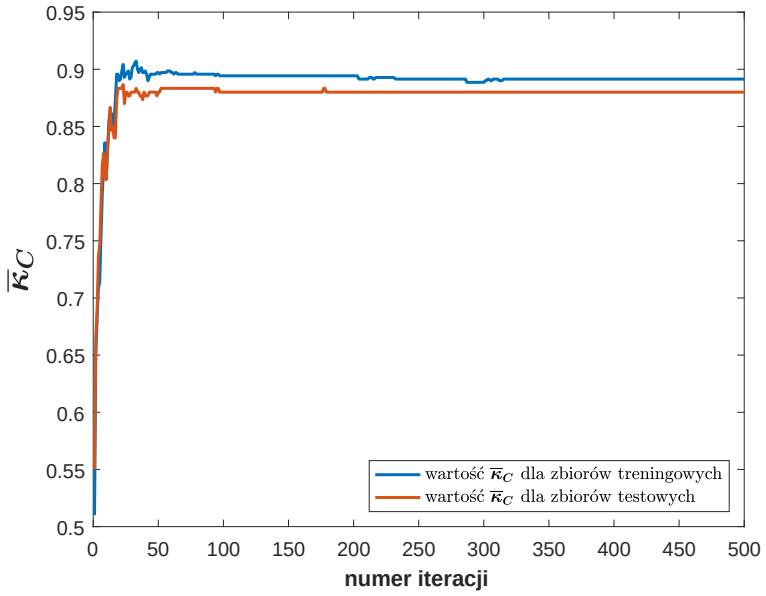
Rys. 22: Współczynnik $\overline{\kappa}_C$ dla (*)ON i (**)ON



Rys. 23: Współczynnik $\overline{\kappa}_C$ dla (*)ON i (**)OFF



Rys. 24: Współczynnik $\overline{\kappa}_C$ dla (*)OFF i (**)ON



Rys. 25: Współczynnik $\overline{\kappa}_C$ dla (*)OFF i (**)OFF

Po przyjrzeniu się wykresom, możemy wysnuć takie same wnioski jak w przypadku analizy celności predykcji.

4. **Ocena predykcji**

	minimum [%]	**mediana [%]**	**maksimum [%]**
(*) ON (**) ON	91,1	95,6	100,0
(*) ON (**) OFF	91,1	95,6	100,0
(*) OFF (*) ON	86,7	93,3	97,8
(*) OFF (*) OFF	86,7	93,3	97,8

Tab. 3: Zestawienie celności predykcji

	minimum	**mediana**	**maksimum**
(*) ON (**) ON	0,867	0,933	1,000
(*) ON (**) OFF	0,867	0,933	1,000
(*) OFF (*) ON	0,800	0,900	0,967
(*) OFF (*) OFF	0,800	0,900	0,967

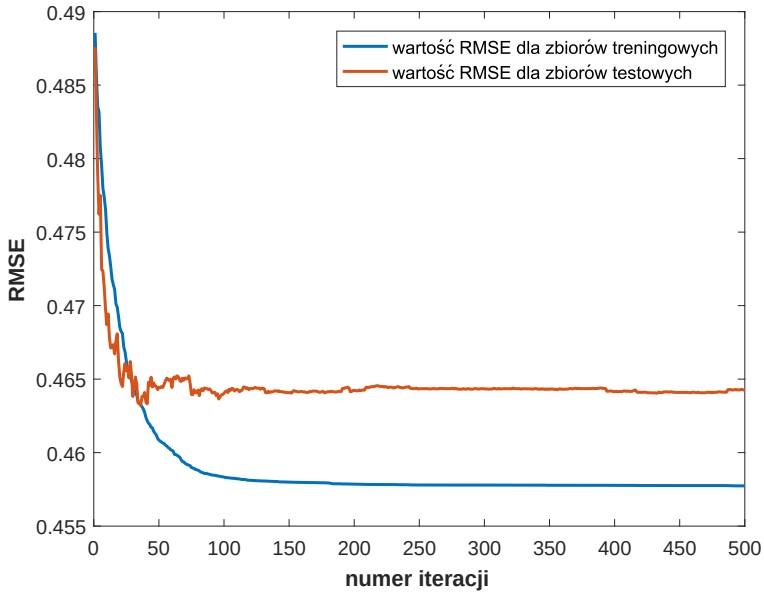
Tab. 4: Zestawienie współczynników Kappa – Cohena

5. **Wnioski końcowe**
   Po analizie wyników możemy ostatecznie stwierdzić że włączona własność (**) w niewielkim stopniu wspomaga proces uczenia i poprawia otrzymane rezultaty, co wynika najprawdopodobniej z działania funkcji `check_Bounds` lub charakterystyki samego zbioru danych. Największy wpływ ma użycie własności (*) powodujące poprawę otrzymanych rezultatów, bez konieczności stosowania dużej liczby iteracji strategii ewolucyjnej. Można więc w praktyce wykorzystać własność (*) jako przyspieszenie zbieżności algorytmu, widząc jednocześnie że spadek wartości RMSE nie musisz zawsze oznaczać poprawy celności predykcji i funkcji oceny rozróżnialności klas.

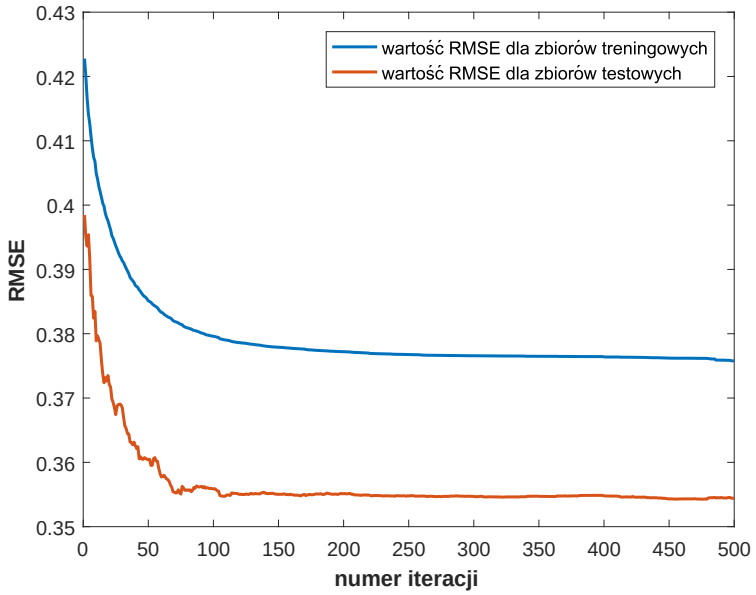
6.3.2 Testy implementacji na zbiorach danych oraz porównanie wyników z innymi algorytmami

Na potrzeby doświadczeń zdecydowano na zastosowanie parametrów (*) i (**) w wersji ON.

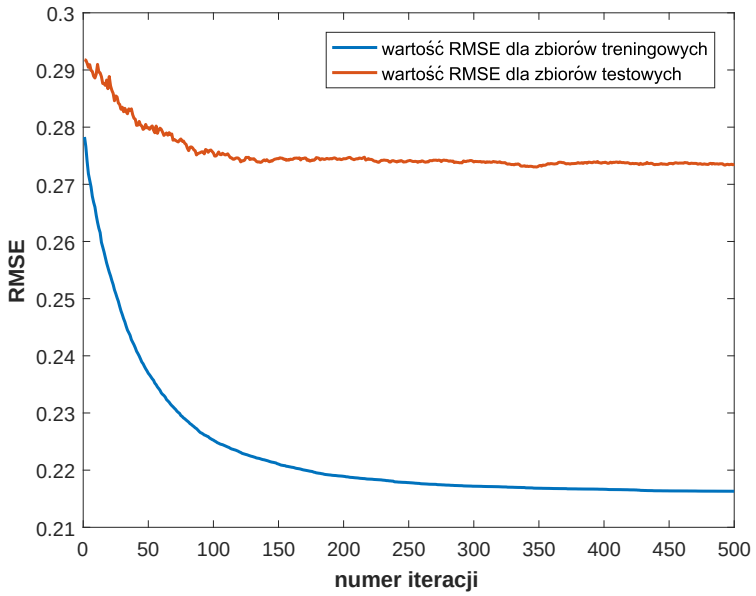
1. **Średnie RMSE**



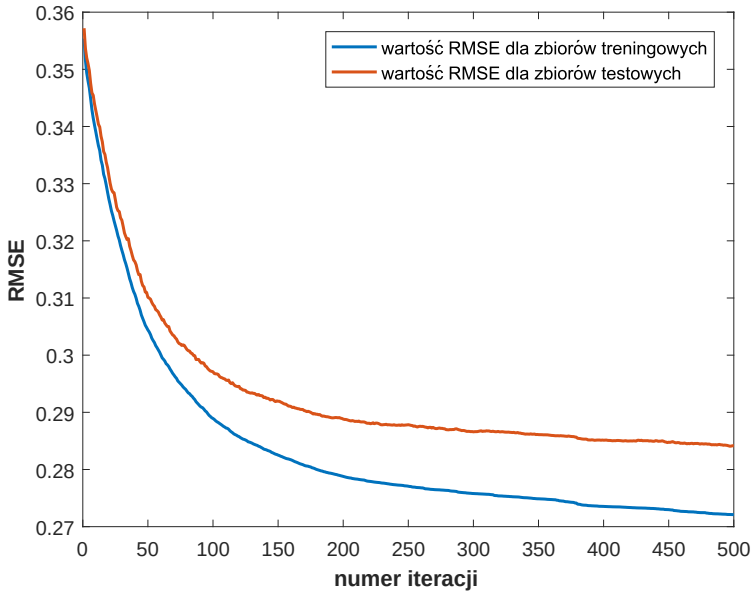
Rys. 26: Wykres RMSE dla zbioru *Cukrzyca*



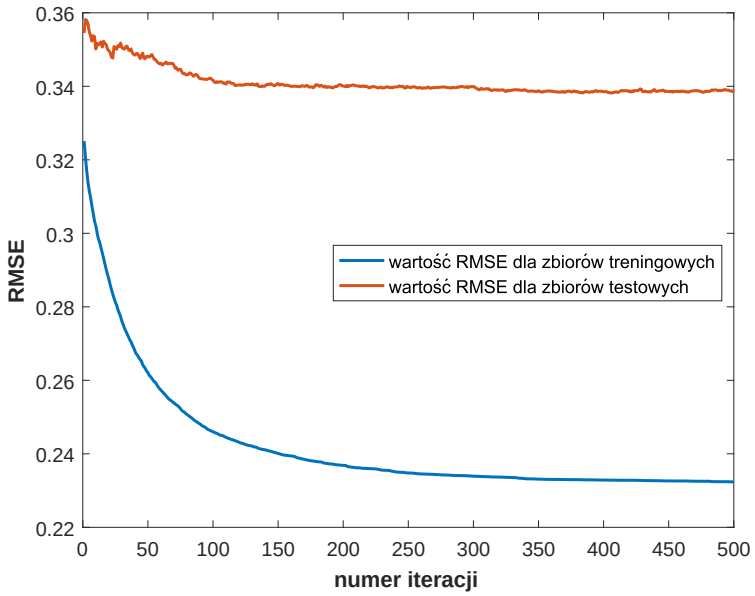
Rys. 27: Wykres RMSE dla zbioru *Serce*



Rys. 28: Wykres RMSE dla zbioru *Białaczka*



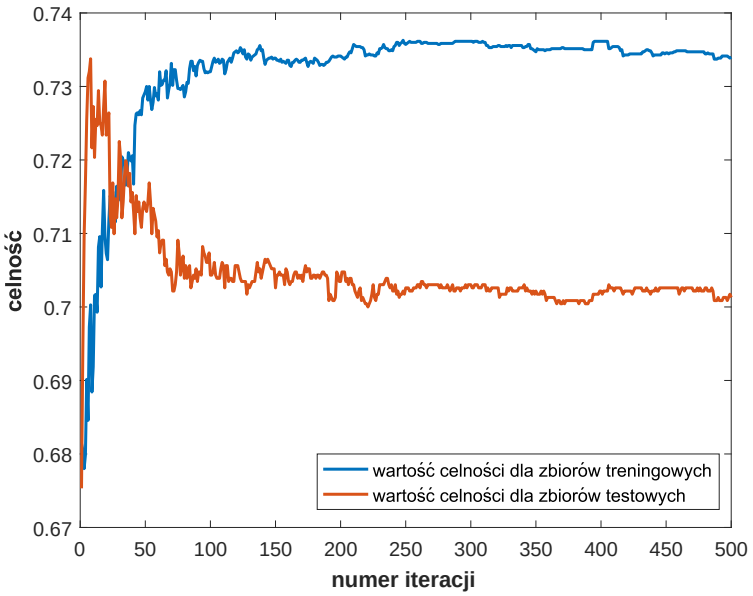
Rys. 29: Wykres RMSE dla zbioru *Rak płuc*



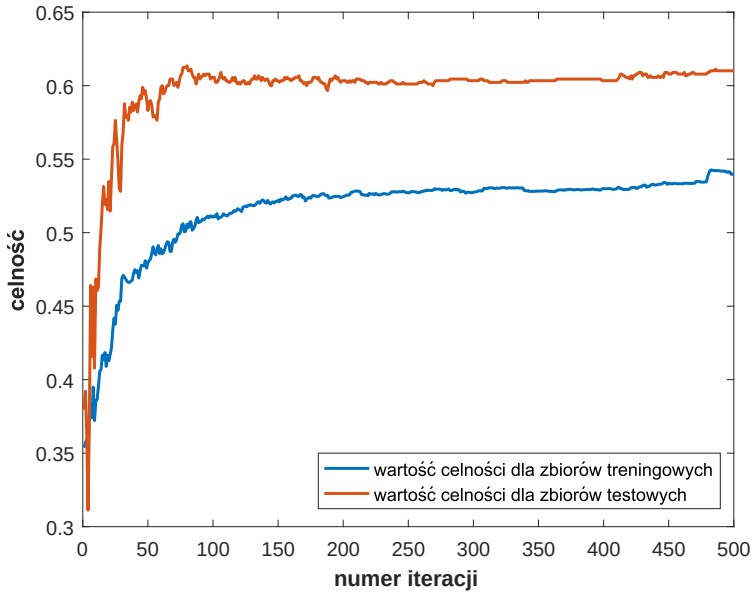
Rys. 30: Wykres RMSE dla zbioru *podpróbkowany Rak płuc*

We wszystkich przypadkach obserwujemy brak zjawiska przeuczenia się systemu. Możemy zaobserwować, że wartości RMSE nie zależą w sposób bezpośredni od liczby kategorii – zbiór *Cukrzyca*, posiada klasyfikację binarną (każdy inny zbiór ma więcej kategorii) jednocześnie osiągając największe wartości RMSE. Wartość RMSE jest więc najprawdopodobniej uzależniona od jakości skorelowania danych między atrybutami, a więc także od ich liczności. Zdają się to potwierdzać zbiory *Serce* oraz *podpróbkowany Rak płóc* dla których mimo takiej samej liczby kategorii widzimy znaczące różnice. Zauważamy również, że mimo tego że zbiór *Białaczka* posiada najmniejszą liczbę przykładów, to osiągane na nim wyniki są najlepsze w porównaniu z innymi zbiorami o zrównoważonych kategoriach. Warto zauważyć również dwa ciekawe przypadki. Pierwszym z nich jest to że, na zbiorze *Serce* osiągane RMSE jest mniejsze dla zbiorów testowych niż dla treningowych. Możemy jednak stwierdzić z dużym prawdopodobieństwem że, jest to cecha charakterystyczna tego zbioru i/lub pewnej dozy losowości wpisanej w algorytm, przy czym z racji dziesięciokrotnego uśredniania możemy raczej założyć że pierwsza opcja jest najbardziej prawdopodobna. Drugim wartym odnotowania przypadkiem jest porównanie wyników dla zbiorów *Rak płuc* i *podpróbkowany Rak płuc*. Widzimy że wyrównanie kategorii powoduje dużo wcześniejszą stabilizację RMSE zarówno dla trenowania jak i testowania, przy czym różnica poziomów stabilizacji dla testowania w obu przypadkach jest znaczna. W przypadku braku wyrównywania kategorii oba wykresy przebiegają znacznie bliżej siebie. Jest to efekt dominacji jednej z klas w znaczącym stopniu nad pozostałymi (klasa dominująca stanowi ok 66.5 % wszystkich przykładów) oraz dużej liczbie kategorii (5).

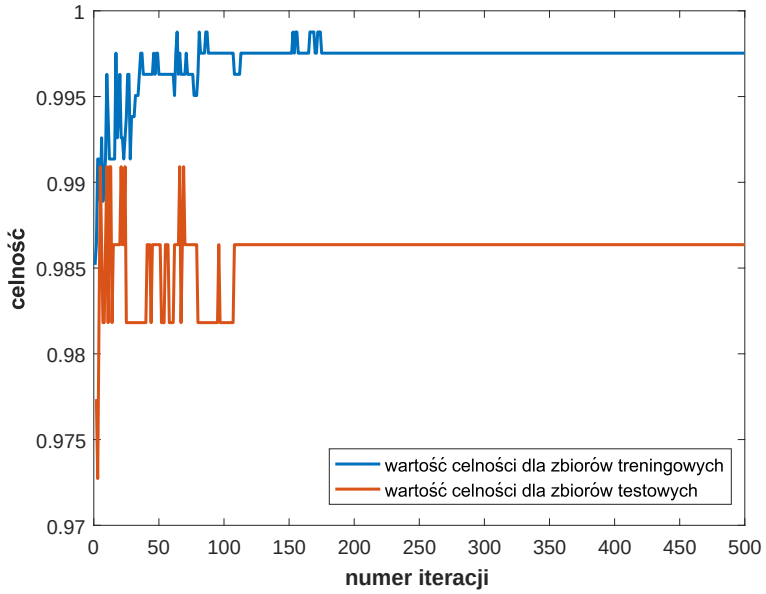
2. **Średnia wartość predykcji**



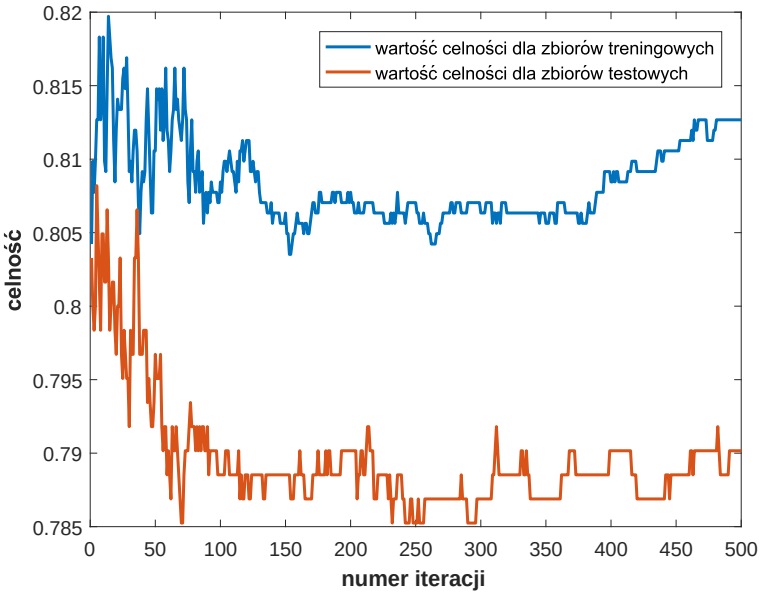
Rys. 31: Wykres celności predykcji dla zbioru *Cukrzyca*



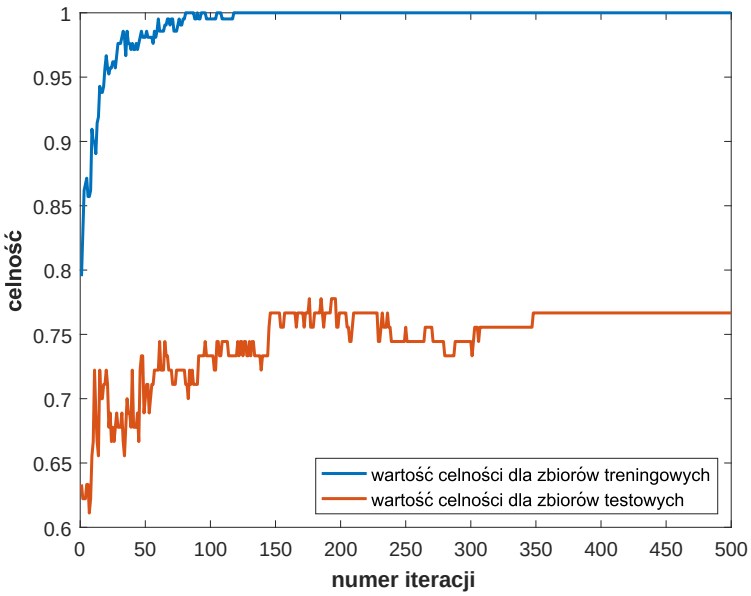
Rys. 32: Wykres celności predykcji dla zbioru *Serce*



Rys. 33: Wykres celności predykcji dla zbioru *Białaczka*



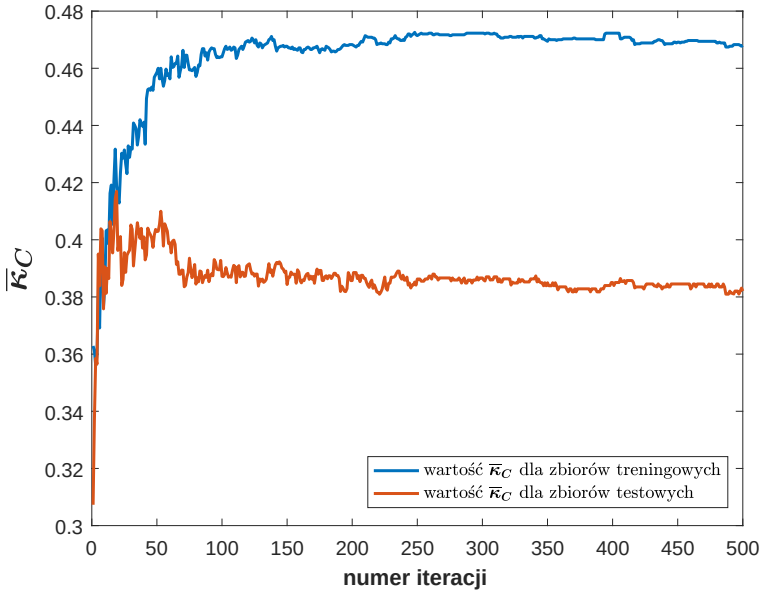
Rys. 34: Wykres celności predykcji dla zbioru *Rak płuc*



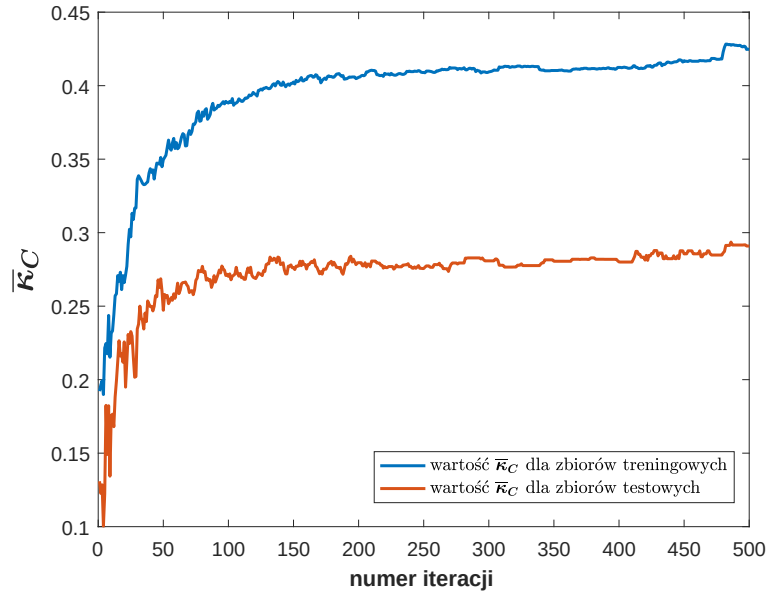
Rys. 35: Wykres celności predykcji dla zbioru *podpróbkowany Rak płuc*

Możemy zauważyć że, spadkowi wartości RMSE nie zawsze towarzysz wzrost celności predykcji, szczególnie w przypadku zbiorów testowych (rysunki 31 i 34), mimo iż we wszystkich sytuacjach (poza zbiorem *Rak płuc*) następuje systematyczny wzrost tej wartości w trakcie uczenia. Dla zbioru *Rak płuc* celność predykcji nawet w trakcie procesu uczenia potrafi systematycznie maleć mimo spadku wartości RMSE. Zbiór o wyrównanych kategoriach *podprób-kowany Rak płuc* zapewnia w przybliżeniu monotonicznie rosnące wartości celności predykcji, jak i mniej skokowy ich przebieg. Ostatecznie w trzech z czterech zrównoważonych zbiorów dłuższe uczenie przynosi lepsze rezultaty. W przypadku zbioru *Cukrzyca* oraz *Rak płuc* najlepsze rezultaty osiągnęliśmy już po kilku – kilkunastu iteracjach, po czym zaczęły się one pogarszać. Widzimy również że wielkość RMSE nie musi mieć bezpośredniego przełożenia na jakość predykcji – przykładowo zbiór *Cukrzyca* posiada większe wartości poziomów odniesienia RMSE niż zbiór *Serce*, a jednak na zbiorze *Serce* osiągane rezultaty celności predykcji są lepsze.

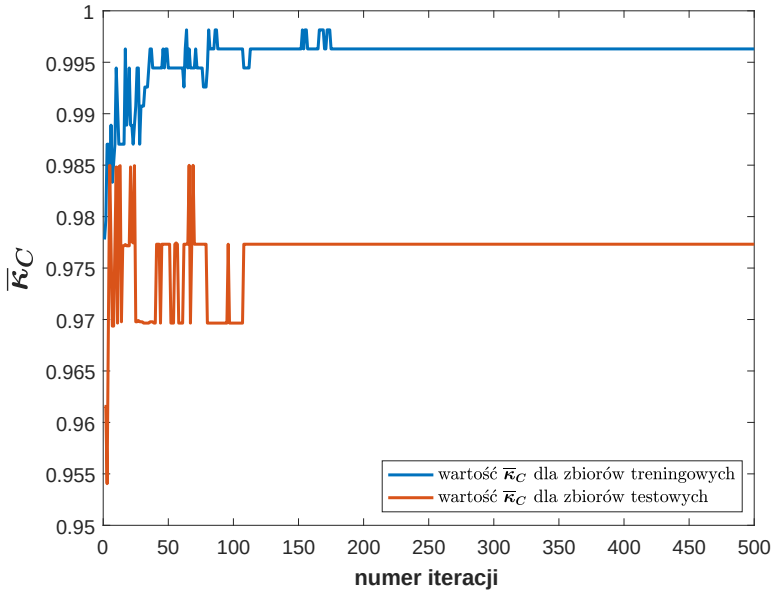
3. **Średni współczynnik Kappa – Cohena**



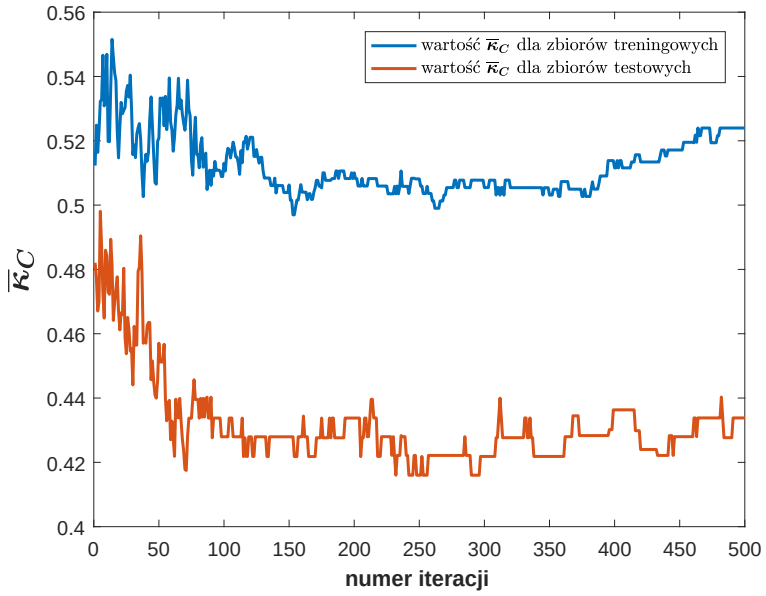
Rys. 36: Wykres $\overline{\kappa}_C$ dla zbioru *Cukrzyca*



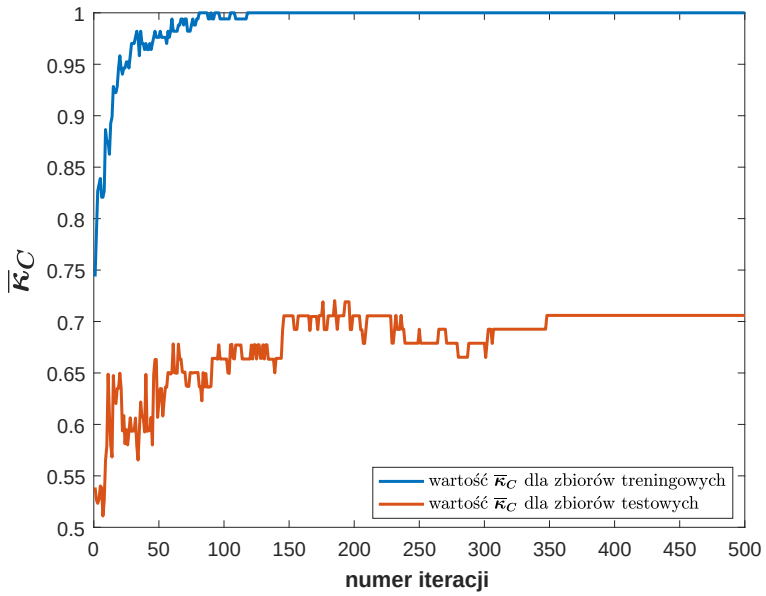
Rys. 37: Wykres $\overline{\kappa}_C$ dla zbioru *Serce*



Rys. 38: Wykres $\overline{\kappa}_C$ dla zbioru *Białaczka*



Rys. 39: Wykres $\overline{\kappa}_C$ dla zbioru *Rak płuc*



Rys. 40: Wykres $\overline{\kappa}_C$ dla zbioru *podpróbkowany Rak płuc*

Wykresy funkcji oceny rozróżnialności klas dla zbiorów *Serce*, *Białaczka*, *Rak płuc* oraz *podpróbkowany Rak płuc* mają podobne przebiegi jak ich odpowiedniki dla celności predykcji (choć z oczywistych względów istnieje inny zakres wartości). Pierwszym faktem istotnym odnotowania jest to że, dla zbioru *Cukrzyca* maksimum wartości $\overline{\kappa}_C$ dla testowania jest stosunkowo niewielkie w porównaniu z poziomem stabilizacji funkcji, w przeciwieństwie do celności predykcji. Drugim ciekawym faktem jest to że dla zbioru **Serca** mimo iż wartości RMSE oraz celności predykcji były lepsze w przypadku testowania, to wartości $\overline{\kappa}_C$ są lepsze w przypadku trenowania. Porównując Rysunki 39 oraz 40 zauważamy również że podpróbkowanie przyniosło faktycznie z godnie z oczekiwaniami lepszą rozróżnialność klas.

4. **Porównanie ocen implementowanego klasyfikatora z algorytmami z języka R**

		Cukrzyca	**Serce**	**Białaczka**	**Rak płuc**	**podpróbkowany Rak płuc**
implementacja	minimum [%]	65,8	50,6	95,5	75,4	66,7
	mediana [%]	69,9	61,2	100,0	79,5	77,8
	maksimum [%]	74,9	65,2	100,0	80,3	88,9
Chi	minimum [%]	48,5	49,4	95,5	70,5	55,6
	mediana [%]	59,3	56,2	100,0	81,1	77,8
	maksimum [%]	64,5	60,7	100,0	88,5	100,0
Ishibuchi & Nakashima	minimum [%]	65,8	53,9	95,5	77,0	44,4
	mediana [%]	70,1	59,0	100,0	82,0	77,8
	maksimum [%]	72,2	61,8	100,0	86,9	77,8

Tab. 5: Porównanie wartości celności predykcji dla różny zbiorów i algorytmów

Poza zbiorem *rak płuc* oraz *podpróbkowany rak płuc* osiągi celności predykcji implementowanego algorytmu są w większości przypadku lepsze lub niemal porównywalne z dwoma pozostałymi testowanymi algorytmami. W przypadku zbioru *Rak płuc* wyniki wartości minimalnych i median są dość zbliżone, a największe różnice obserwujemy w wielkościach maksymalnych. Dla zbioru *podpróbkowany Rak płuc*, wartość minimalna okazuje się najlepsza dla implementowanego algorytmu a wartości median wszystkich algorytmów są takie same. Ponownie obserwujemy największe różnice dla wartości maksymalnych, przy czym w tym przypadku algorytm okazuje się lepszy od systemu *Ishibuchi & Nakashima* oraz gorszy od systemu *Chi*. Możemy więc stwierdzić że, wyrównanie kategorii pozytywnie wpłynęło na ocenę implementacji w porównaniu z pozostałymi algorytmami. We wszystkich przypadkach poza wartościami maksymalnymi dla implementowanego algorytmu oraz systemu *Chi* podpróbkowanie zgodnie z oczekiwaniami spowodowało spadek wartości celności predykcji.

		Cukrzyca	**Serce**	**Białaczka**	**Rak płuc**	**podpróbkowany Rak płuc**
implementacja	minimum	0,310	0,168	0,923	0,298	0,571
	mediana	0,368	0,282	1,000	0,453	0,719
	maksimum	0,470	0,381	1,000	0,482	0,861
Chi	minimum	0,103	0,214	0,926	0,410	0,446
	mediana	0,204	0,276	1,000	0,617	0,740
	maksimum	0,267	0,354	1,000	0,775	1,000
Ishibuchi & Nakashima	minimum	0,218	0,185	0,927	0,479	0,302
	mediana	0,319	0,250	1,000	0,605	0,727
	maksimum	0,376	0,348	1,000	0,720	0,746

Tab. 6: Porównanie wartości współczynnika Kappa – Cohena

Ponownie obserwujemy, że dla wszystkich zbiorów danych poza *Rak płuc* oraz *podpróbkowany Rak płuc* wyniki funkcji oceniającej rozróżnialność klas są lepsze dla implementowanego algorytmu lub z nim porównywalne. W przypadku zbioru *rak Płuc* algorytm osiąga zdecydowanie najgorsze wyniki ze wszystkich testowanych algorytmów. Po przeprowadzeniu podpróbkowania, implementowany algorytm posiada największą minimalną wartość wśród wszystkich algorytmów, oraz zbliżoną choć mniejszą od pozostałych systemów medianę. Ponownie możemy zauważyć że, implementowany algorytm znajduje się na drugim miejscu pod względem wielkości maksymalnej wartości współczynnika κ_C. Stwierdzamy podobnie jak w przypadku celności predykcji że, wyrównanie kategorii pozwoliło zarówno zwiększyć rozróżnialność klas dla wszystkich algorytmów w większości przypadków (poza wartością minimalną algorytmu *Ishibuchi & Nakashima*) jak i zbliżyć osiągi implementacji w stosunku do dwóch pozostałych algorytmów. Wyniki na zbiorze *Rak płuc* oraz *podpróbkowany rak Płuc* możemy tłumaczyć najprawdopodobniej albo specyfiką samego zbioru danych albo dużą liczba klas i bardzo małą liczbą przykładów, przez co system uczący się w schemacie RMSE może osiągać gorsze wyniki. Zauważmy również że, wyrównanie kategorii wpływa pozytywnie na działanie systemu jeśli porównywać go w tej sytuacji z innymi algorytmami.

7 Zakończenie

7.1 Wnioski końcowe

W wyniku przeprowadzonych badań oraz doświadczeń możemy stwierdzić że zaimplementowany algorytm jest godnym kandydatem dla innych algorytmów rozmytych. Do jego głównych cech możemy zaliczyć

Zalety

- dobre osiągi zarówno pod względem celności predykcji jak i rozróżnialności klas dla dużej liczby zbiorów
- spójność – zawsze osiągniemy odpowiedź, w przeciwieństwie do systemów stricte regułowych, które by osiągnąć taki sam efekt muszą albo mieć zdefiniowaną regułę domyślna albo też przeprowadzać aproksymację
- niesprzeczność – nie istniej możliwość dostania kilku różnych odpowiedzi na jeden wektor wejściowy
- w porównaniu z algorytmami z języka R, skalowanie danych nie jest konieczne (choć nie badano jak wpływa to na wyniki), jak również kategorie mogą być dowolnego typu

Wady

- wybranie na metodę optymalizacji strategii ewolucyjnej lub algorytmu genetycznego powoduje dłuższy czas uczenia niż w przypadku badanych algorytmów z języka R
- uczenie systemu po przez RMSE nie zawsze sprawia, że osiągane wskaźniki predykcyjne są lepsze, zwłaszcza na zbiorach testowych
- założenie o tym że, najlepszym kandydatem zawsze jest osobnik z najmniejszym RMSE nie musi zawsze być koniecznie spełnione
- system wyucza się każdej kategorii osobno, co utrudnia przeprowadzenie badań w przypadku celności predykcji i oceny rozróżnialności klas
- najlepsze wyniki osiąga dla dużej liczby przykładów i zrównoważonych kategorii

7.2 Dalsze kierunki rozwoju

- implementacja systemu, który uczyłby się po mierze predykcyjnej zamiast RMSE, co mogło by rodzić nadzieje na lepsze rezultaty w pewnych sytuacjach
- optymalizacja parametrów systemu ewolucyjnego
- testowanie algorytmu dla innych funkcji przynależności oraz operatorów agregacji
- zmiana systemu optymalizacji np: na system rojowy

8 Bibliografia

W przypadku linków dostępność była weryfikowana po raz ostatni *19.01.2018*

[1] *https://expertsystem101.weebly.com/mycin.html*

[2] Cichossz Paweł, *Systemu uczące się*, Wydawnictwo Naukowo – Techniczne WNT, Warszawa 2007, wydanie drugie

[3] Ossowski Stanisław, *Sieci neuronowe do przetwarzania informacji*, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2013, wydanie trzecie

[4] Guzicki Wojciech, Zakrzewski Piotr, *Wykłady ze wstępu do matematyki*, Wydawnictwo Naukowe PWN, Warszawa 2012, wydanie pierwsze – czwarty dodruk

[5] Kraszewski Jan, *Wstęp do matematyki*, Wydawnictwo Naukowo – Techniczne WNT, Warszawa 2014, wydanie drugie zmienione – dodruk

[6] Łachwa Andrzej, *Rozmyty świat zbiorów, liczb, relacji, faktów, reguł i decyzji*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2001,

[7] Rutkowski Leszek, *Metody i techniki sztucznej inteligencji*, Wydawnictwo Naukowe PWN, Warszawa 2012, wydanie drugie zmienione – trzeci dodruk

[8] *https://www.mathworks.com/help/fuzzy/pimf.html?requestedDomain=true*

[9] *https://www.researchgate.net/publication/280038962_Local_and_Global_Genetic_Fuzzy_Pattern_Classifiers*

[10] *https://cran.r-project.org/web/packages/frbs/index.html*

[11] Z. Chi, H. Yan, T. Pham, *Fuzzy algorithms with applications to image processing and pattern recognition*

[12] H. Ishibuchi, T. Nakashima *Effect of rule weights in fuzzy rule-based classification systems*

[13] *https://www.mathworks.com/products/fuzzy-logic.html*

[14] *https://www.fuzzylite.com/cpp/*

[15] *https://www.fuzzylite.com/java/*

[16] *http://jfuzzylogic.sourceforge.net/html/index.html*

[17] *http://www2.imse-cnm.csic.es/Xfuzzy/*

[18] *https://pypi.python.org/pypi/pyfuzzy*

[19] *http://johmathe.name/gfuzzy.html*

[20] *https://pypi.python.org/pypi/scikit-fuzzy*

[21] *https://cran.r-project.org/web/packages/FuzzyR/index.html*

[22] Michalewicz Zbigniew, *Algorytmy genetyczne + struktury danych = programy ewolucyjne*, Wydawnictwo Naukowo – Techniczne WNT, Warszawa 2003, wydanie trzecie

[23] *https://deap.readthedocs.io/en/master/*

[24] *https://www.jair.org/media/953/live-953-2037-jair.pdf*

[25] *http://contrib.scikit-learn.org/imbalanced-learn/stable/generated/imblearn.over_sampling.SMOTE.html*

[26] *https://archive.ics.uci.edu/ml/datasets/iris*

[27] *https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes*

[28] *http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/reprocessed.hungarian.data*

[29] *http://www.ire.pw.edu.pl/~trubel/dydaktyka/mpb/proj.html#data*