Rozdział 3

WPŁYW DYSKRETYZACJI DANYCH NA SKUTECZNOŚĆ DZIAŁANIA ALGORYTMÓW SELEKCJI NEGATYWNEJ

Izabela Kartowicz-Stolarska*

Streszczenie Algorytmy selekcji negatywnej (ang. *negative selection algorithm*, NSA) są jednym z podejść sztucznych układów immunologicznych, które wzorują się na mechanizmach obronnych zachodzących w naturalnych organizmach. Przez ostatnie lata algorytmy te znalazły szerokie zastosowanie w ochronie danych oraz procesach związanych z bezpieczeństwem sieci komputerowych. Jak pokazują dotychczasowe badania, NSA z powodzeniem może być również używany jako klasyfikator, zarówno binarny, jak i dwuklasowy i wieloklasowy. Niniejsza praca jest krótkim wprowadzeniem do teorii i zastosowania sztucznych systemów immunologicznych w kontekście dyskretyzacji danych. W rozdziale przedstawiono przegląd i porównanie wybranych rozwiązań na przykładzie algorytmu selekcji negatywnej. Omówiono tu ogólne zasady konstruowania algorytmów selekcji negatywnej, jak również szczegóły implementacji podanych rozwiązań. Następnie zbadano wpływ dyskretyzacji atrybutów na działanie poszczególnych implementacji. Z uwagi na możliwość przedstawienia wyników eksperymentów z danymi przed i po dyskretyzacji, w przeglądzie uwzględniono algorytmy NSA, w których przeciwciała reprezentowane są jako liczby rzeczywiste. Badania zostały przeprowadzone na kilku zbiorach danych. Do eksperymentów wykorzystano różne algorytmy dyskretyzacji, w tym metody oparte na teorii zbiorów przybliżonych.

Słowa kluczowe: sztuczne systemy immunologiczne, selekcja negatywna, klasyfikacja, dyskretyzacja atrybutów

Wprowadzenie

Mechanizmy działania naturalnego systemu immunologicznego (ang. *natural immune system*, NIS) kręgowców, a w szczególności jego adaptacja do zmiennego otoczenia oraz możliwość rozpoznania patogenów (m.in. wirusów, bakterii czy pierwot-

* Wydział Informatyki, Politechnika Białostocka, Wiejska 45A, 15-351 Białystok, i.stolarska@pb.edu.pl

DOI 10.24427/978-83-66391-58-1_3

niaków), stały się inspiracją dla naukowców do opracowania i rozwoju sztucznych systemów immunologicznych.

Układ odpornościowy jest rozproszony i unikatowy dla każdego osobnika. Ma zdolność uczenia się i zapamiętywania patogenów, z którymi zetknął się w czasie swojego funkcjonowania, a do wykrycia ich wystarczy mu tylko znajomość struktur komórek własnych (Wierzchoń, 2001). Za rozpoznanie komórek obcych w organizmie i ich eliminację odpowiadają produkowane w szpiku kostnym krwinki białe (limfocyty). Limfocyty można podzielić na limfocyty typu T i typu B (Lydyard, Whelan i Fanger, 2001).

Limfocyty typu T dojrzewają w grasicy, gdzie przechodzą proces nauki rozpoznawania struktur komórek własnych. Dopiero tak wykształcone krwinki białe typu T trafiają do krwi obwodowej i narządów limfatycznych, gdzie są odpowiedzialne za rozpoznanie patogenów i podejmowanie akcji obronnej organizmu. Proces nauki rozpoznawania komórek własnych jest w immunologii nazywany selekcją negatywną. W dużym uproszczeniu polega on na eliminacji niedojrzałych limfocytów typu T, które rozpoznają komórki własne jako obce, stanowiące zagrożenie dla organizmu.

Limfocyty typu B odpowiadają za zapamiętanie i niszczenie patogenów. Znany organizmowi patogen nazywany jest antygenem. Na powierzchni limfocytów typu B znajduje się około 100 tysięcy receptorów. Receptory i antygeny posiadają fragmenty, które tworzą wiązanie: paratop w receptorze i epitop w antygenie. Pojedynczy limfocyt posiada receptory z paratopami umiejącymi wiązać tylko jeden rodzaj epitopów (unikatowa swoistość). Istniejące limfocyty ulegają licznym podziałom z uwzględnieniem mutacji. W procesach tych receptory komórek potomnych nieznacznie się zmieniają, umożliwiając tym samym skuteczniejsze od komórki rodzica wiązanie patogenów. Dzięki temu odpowiedź immunologiczna (reakcja organizmu na kontakt z antygenem) jest precyzyjna.

Limfocyty są aktywowane, gdy ich receptory rozpoznają antygen. W efekcie powstaje wiele klonów o identycznej z komórką oryginalną swoistości (Lasek, Lasek i Pęczkowski, 2013). Dzięki temu dochodzi do wytworzenia swoistej odporności organizmu na dany antygen. Opisany powyżej proces nazywany jest w immunologii selekcją klonalną (ang. *clonal selection*). W procesie tym intensywnemu klonowaniu ulegają limfocyty, których receptory są najlepiej dopasowane do antygenu. Klony z receptorami słabo wiążącymi antygen są usuwane z organizmu. Natomiast te, które mają silne powiązania paratop-epitop pozostają, przekształcając się w komórki plazmatyczne lub pamięciowe. W ten sposób zostaje zapewniona pamięć immunologiczna i szybsza reakcja obronna organizmu przy ponownym kontakcie z antygenem.

Wymienione wyżej mechanizmy, jak również inne teorie z dziedziny immunologii (np. teoria niebezpieczeństwa (Matzinger, 1994)) zainspirowały badaczy do przełożenia własności naturalnego układu immunologicznego na pole informatyki i inżynierii matematycznej.

Jedną z pierwszych prac przekładających działanie mechanizmów NIS na działanie sieci komputerowych była teoria sieci idiotypowej zaproponowana przez Jerne

(1973). Artykuł ten stał się podstawą do powstania grupy rozwiązań nazwanych sztucznymi sieciami immunologicznymi. Przełomową pracą dotyczącą modelowania sieci idiotypowych był artykuł Farmera, Packarda i Perelsona z 1986 roku "The Immune System, Adaptation, and Machine Learning", w którym zaproponowano zastosowanie rozwiązań naśladujących działanie układów odpornościowych do ochrony sieci (Wierzchoń, 2001). W kolejnych latach pojawiły się liczne prace dotyczące implementacji algorytmów symulujących działanie naturalnych systemów immunologicznych.

Na podstawie literatury można wnioskować, że wśród badaczy największe zainteresowanie, oprócz sztucznych sieci immunologicznych, znalazły algorytmy selekcji negatywnej i selekcji klonalnej. W przypadku algorytmów selekcji klonalnej najważniejszą cechą jest dopasowanie komórki limfocytów B do antygenów. Ogólna idea ich działania polega na wybraniu najlepiej dopasowanych przeciwciał do antygenów i ich sklonowaniu. W algorytmie uwzględniono proces mutacji przeciwciał, w taki sposób, aby nowe przeciwciała odznaczały się lepszym dopasowaniem do antygenu, niż przed mutacją. Algorytmy te znalazły zastosowanie głównie w zagadnieniach optymalizacji i rozpoznawania wzorców. Z kolei algorytmy selekcji negatywnej znalazły szerokie zastosowanie przy ochronie danych i procesów realizowanych w sieci (Hofmeyr, 1999) oraz w wykrywaniu nieprawidłowości takich, jak wirusy komputerowe bądź zakłócenia w szeregach czasowych (Dasgupta i Forrest, 1995). Z powodzeniem mogą być również używane do zagadnień klasyfikacji (Esponda, Forrest i Helman, 2003). W ostatnich latach popularne są szczególnie rozwiązania hybrydowe, łączące różne dziedziny z zakresu informatyki, np. wspomagające wspomnianą wyżej klasyfikację obiektów (Hońko, 2018) lub optymalizację funkcji multimodalnych (Lucińska, 2010), a nawet symulujące układ odpornościowy okrętu (Praczyk, 2010).

Dyskretyzacja jest jednym z istotnych zadań wykonywanych na potrzeby eksploracji danych (García, Luengo, Sáez, López i Herrera, 2013). Co prawda, jest ona wykorzystywana przy wykrywaniu anomalii, ale w innych obszarach niż algorytmy immunologiczne. W pracy ”Continuous Features Discretization for Anomaly Intrusion Detectors Generation” zaproponowano zastosowanie dyskretyzacji w algorytmach genetycznych do generowania detektorów wykrywających nieprawidłowości w sieci. W związku z tym, że dane wykorzystywane w systemach do wykrywania włamań są wielowymiarowe, dyskretyzacja jest wskazana jako wstępny etap przetworzenia zbiorów (Aziz, Azar, Hassanien i Hanafy, 2014) i wyostrzenia pewnych różnic między obiektami.

Celem poniższej pracy jest wstępne zbadanie, w jaki sposób dyskretyzacja cech wpływa na klasyfikację algorytmów immunologicznych na przykładzie wybranych rozwiązań. Aby wyjaśnić tę kwestię, w dalszej części pracy szczegółowo omówiono trzy różne implementacje algorytmu symulującego działanie mechanizmu selekcji negatywnej. Następnie porównano wyniki dla danych zdyskretyzowanych za po-

mocą różnych algorytmów. Scenariusz eksperymentów i jego wyniki opisano w rozdziale 3.2, w ostatniej części zawarto podsumowanie.

3.1 Algorytm selekcji negatywnej

Algorytm selekcji negatywnej (ang. *negative selection algorithm*, NSA) wzoruje się na mechanizmie usuwania limfocytów typu T, które rozpoznały własne struktury jako patogenne. Po raz pierwszy został wdrożony przez Forrest, Perelson, Allen i Cherukuri (1994). Przez ostatnie 25 lat algorytm ten był wielokrotnie modyfikowany, ale główne cechy zaproponowane w oryginalnym rozwiązaniu pozostały bez zmian.

Algorytm selekcji negatywnej składa się z dwóch podstawowych kroków:

- generowanie detektorów (przeciwciał) i ich cenzurowanie,
- monitorowanie danych.

Na wejściu algorytm generowania detektorów otrzymuje zbiór z wzorcami własnych komórek. Następnie losowo generuje zbiór detektorów reprezentujących przeciwciała. Podczas cenzurowania usuwane są te detektory, które rozpoznają własne struktury. W efekcie na wyjściu otrzymuje się zbiór tylko tych detektorów, które nie rozpoznały własnych komórek jako obcych. W uproszczeniu algorytm można opisać za pomocą pseudokodu algorytm 3.1.

Algorytm 3.1 Generowanie i cenzurowanie detektorów

INPUT:
S - zbiór wzorców własnych komórek
OUTPUT:
D - zbiór detektorów komórek obcych

```
1: D ← losowo wygeneruj detektory
2: for d ∈ D do
3:     if d rozpoznaje komórkę własną z S then
4:         D ← D \ {d}
5:     end if
6: end for
7: return D
```

Kolejne kroki algorytmu selekcji negatywnej służą do rozpoznania komórek obcych w monitorowanym zbiorze danych. Proces monitorowania przedstawiony został za pomocą pseudokodu algorytm 3.2.

Algorytm 3.2 na wejściu otrzymuje wzorce wszystkich komórek: zarówno własnych, jak i obcych oraz zbiór wygenerowanych detektorów. Na podstawie tych da-

Algorytm 3.2 Monitorowanie danych

INPUT:
D - zbiór detektorów
OUTPUT:
S_{self} - zbiór komórek rozpoznanych jako własne
$S_{nonself}$ - zbiór komórek rozpoznanych jako obce

1: $S_{self} \leftarrow \emptyset$
2: $S_{nonself} \leftarrow \emptyset$
3: **for** $s \in S$ **do**
4: **if** $\exists_{d \in D} match(d, s)$ **then**
5: $S_{nonself} \leftarrow S_{nonself} \cup \{s\}$
6: **else**
7: $S_{self} \leftarrow S_{self} \cup \{s\}$
8: **end if**
9: **end for**

nych oraz reguły dopasowania (ang. *matching rule*) struktury komórki rozpoznawane są jako własne lub obce.

Reguła dopasowania jest wykorzystywana zarówno podczas generowania detektorów, jak i w fazie wykrywania anomalii. Można ją zdefiniować jako odległość między dwoma punktami: detektorem, a komórką danych. Co ciekawe dwa punkty nie muszą być dokładnie takie same, aby zostały uznane za zgodne. Jest to możliwe dzięki wprowadzeniu do algorytmu progu dopasowania (Ji i Dasgupta, 2007).

Wybór reguły dopasowania lub progu w regule dopasowania jest specyficzny dla aplikacji i zależny od reprezentacji (ang. *data representation*).

Reprezentacja danych jest jedną z podstawowych różnic w implementacjach algorytmu selekcji negatywnej. Wyróżnia się dwie główne reprezentacje: niskiego i wysokiego poziomu. Reprezentacja niskiego poziomu to ciągi znaków (ang. *string representation*), do których zaliczana jest również reprezentacja binarna. Z kolei do reprezentacji wysokiego poziomu zaliczają się wektory wartości liczb rzeczywistych (ang. *real-value representation*). Reprezentacja wektorów liczb rzeczywistych została wprowadzona, by przyśpieszyć proces generowania detektorów. Ponadto, dzięki wykorzystaniu takiej reprezentacji nie są tracone informacje o poszczególnych cechach zbioru (Gonzalez i Dasgupta, 2003).

Z uwagi na fakt przeprowadzenia eksperymentów w kontekście dyskretyzacji danych, w dalszej części pracy skoncentrowano się na omówieniu wybranych algorytmów selekcji negatywnej w reprezentacji liczb rzeczywistych. Porównane zostały algorytmy:

- Real-Valued Negative Selection (Gonzalez i Dasgupta, 2003),
- V-detector (Ji i Dasgupta, 2004),
- algorytm detektorów RST (Chmielewski, 2017).

3.1.1 Algorytm Real-Valued Negative Selection (RNS)

Algorytm RNS został po raz pierwszy zaproponowany w 2002 roku przez Gonzaleza. Głównym założeniem, jakim odróżnia się on od standardowego algorytmu selekcji negatywnej jest reprezentacja danych osadzona w przestrzeni liczb rzeczywistych. W algorytmie tym przestrzeń komórek własnych i obcych reprezentowana jest jako punkty w przestrzeni N-wymiarowej, których współrzędne są wartościami liczb rzeczywistych z zakresu [0, 1]. Detektor (przeciwciało) jest definiowany jako N-wymiarowa hipersfera, której środek określa N-wymiarowy wektor, a promień jest stały i wyraża się liczbą rzeczywistą.

W procesie generowania detektorów (algorytm 3.3) w pierwszym kroku losowanych jest T_{max} detektorów o zadanym promieniu l (funkcja $GenerateRandomDetector(l)$). Następnie dla każdego wylosowanego detektora wybieranych zostaje k- najbliższych sąsiadów spośród komórek własnych. Za ten krok w pseudokodzie odpowiada funkcja $GetKNearestCells(d,S)$, gdzie d to wylosowany detektor, a S to zbiór komórek własnych. Dalej pobierany jest środkowy element spośród posortowanych według odległości od detektora sąsiadów (funkcja $GetMedianNearestCell$). A następnie za pomocą funkcji odległości euklidesowej sprawdzana jest odległość między detektorem, a pobraną komórką. Jeśli w badanym obszarze znajdzie się więcej, niż połowa z najbliższych komórek, uznaje się, że detektor je rozpoznaje i w związku z tym detektor zostaje przesunięty na większą odległość. Istotnym elementem przy wyliczaniu wartości powyższej odległości jest funkcja $\mu_d(x)$, która określa stopień nachodzenia detektora na komórkę (własną lub obcą). Funkcja zdefiniowana jest jako:

$$\mu_d(x) = e^{\frac{-\|d-x\|^2}{2r^2}} \tag{3.1}$$

gdzie:

- $\|.\|$ - norma euklidesowa,
- r - promień detektora,
- d - współrzędne środka detektora,
- x - współrzędne komórki.

Proces przesuwania detektorów jest powtarzany, aż mniej niż połowa sąsiadów jest wykrywana lub przekroczono liczbę założonych iteracji algorytmu. Powyższe warunki są niewystarczające, aby algorytm był zbieżny. Początkowe tempo adaptacji oraz tempo zanikania adaptacji powodują przesuwanie detektora o coraz mniejsze odległości, zapewniając zbieżność algorytmu.

Może się zdarzyć, że wylosowany detektor będzie otoczony przez komórki własne w taki sposób, że przesunięcie go w dowolnym kierunku nie spowoduje zmniejszenia wykrywalności komórek własnych. Aby zapobiec takiej sytuacji, z każdą iteracją detektory starzeją się (zmienna age_d) i są usuwane, gdy osiągną założoną dojrzałość, a wciąż będę wykrywały zbyt wiele komórek własnych.

Algorytm 3.3 RNS - generowanie detektorów

INPUT:

l - promień komórki
S - zbiór komórek własnych
r - promień detektora
η_0 - początkowe tempo adaptacji
τ - tempo zanikania adaptacji
t - wiek, w którym detektor uznawany jest za dojrzały
T_{max} - liczba detektorów do wygenerowania
I_{max} - liczba iteracji

OUTPUT:

D - zbiór detektorów

1: $D \leftarrow \{GenerateRandomDetector_i(l) | i \in 1 \rightarrow T_{max}\}$
2: $i \leftarrow 0$
3: **while** $i < I_{max}$ **do**
4: $\quad \eta \leftarrow \eta_0 * e^{-i/\tau}$
5: $\quad$ **for** $d \in D$ **do**
6: $\quad\quad NearCells \leftarrow GetKNearestCells(d, S)$
7: $\quad\quad NearSelf \leftarrow GetMedianNearestCell(NearCells)$
8: $\quad\quad$ **if** $distance(d, NearestSelf) < r$ **then**
9: $\quad\quad\quad dir \leftarrow \frac{\sum_{c \in NearCells}(d-c)}{|NearCells|}$
10: $\quad\quad\quad$ **if** $age_d > t$ **then** $\quad \triangleright\ age_d$ – wiek detektora d
11: $\quad\quad\quad\quad d \leftarrow GenerateRandomDetector(l)$
12: $\quad\quad\quad$ **else**
13: $\quad\quad\quad\quad age_d \leftarrow age_d + 1$
14: $\quad\quad\quad\quad d \leftarrow d + \eta * dir$
15: $\quad\quad\quad$ **end if**
16: $\quad\quad$ **else**
17: $\quad\quad\quad age_d \leftarrow 0$
18: $\quad\quad\quad dir \leftarrow \frac{\sum_{d' \in D} \mu_d(d') * (d - d')}{\sum_{d' \in D} \mu_d(d')}$
19: $\quad\quad\quad d \leftarrow d + \eta * dir$
20: $\quad\quad$ **end if**
21: $\quad$ **end for**
22: $\quad i \leftarrow i + 1$
23: **end while**

Warunek stopu algorytmu 3.3 określany jest przez liczbę iteracji. Dzięki temu złożoność czasowa algorytmu wyraża się jako:

$$O(I_{max} * |D| * (|D| + |S|)) \tag{3.2}$$

W procesie monitorowania danych (algorytm 3.4) reguła dopasowania wyrażana jest za pomocą funkcji odległości euklidesowej pomiędzy detektorem, a komórką. Komórka, która znajdzie się w promieniu detektora, uznawana jest za komórkę obcą.

Algorytm 3.4 RNS - monitorowanie

INPUT:
D - zbiór detektorów
S - zbiór monitorowanych komórek
OUTPUT:
S_{self} - zbiór komórek rozpoznanych jako własne
$S_{nonself}$ - zbiór komórek rozpoznanych jako obce

1: $S_{self} \leftarrow \emptyset$
2: $S_{nonself} \leftarrow \emptyset$
3: **for** $s \in S$ **do**
4: **if** $\exists_{d \in D} match(d, s)$ **then**
5: $S_{nonself} \leftarrow S_{nonself} \cup \{s\}$
6: **else**
7: $S_{self} \leftarrow S_{self} \cup \{s\}$
8: **end if**
9: **end for**

3.1.2 Algorytm V-detector

Zaproponowany w 2004 algorytm jest modyfikacją algorytmu RNS. W odróżnieniu od powyższego rozwiązania, które bazuje na detektorach o stałej długości promienia, algorytm V-detector umożliwia modyfikację długości promienia danego detektora w procesie generowania detektorów. Komórki własne i obce nie są punktami, ale hipersferami w N-wymiarowej przestrzeni.

W pierwszej kolejności losowany jest jeden detektor. Następnie sprawdzane jest, czy wylosowany detektor znajduje się w przestrzeni monitorowanej już przez inne, istniejące detektory. Jeśli tak, to losowany jest nowy detektor. Promień detektora ustawiany jest jako odległość do najbliższej komórki własnej. Proces jest powtarzany, aż zostanie osiągnięta założona liczba detektorów (algorytm 3.5).

Algorytm zatrzyma się, jeżeli zostanie spełniony jeden z trzech warunków:

- wygenerowana zostanie założona liczba detektorów (T_{max}),
- osiągnięty zostanie założony poziom pokrycia przestrzeni przez detektory (c_0),
- stwierdzone zostanie zbyt duże pokrycie przestrzeni przez komórki własne (maximum self coverage).

Algorytm 3.5 V-detector - generowanie detektorów

INPUT:
l - promień komórki
S - zbiór komórek własnych
R_s - promień komórki własnej
c_0 - oczekiwane pokrycie detektorami
T_{max} - maksymalna liczba detektorów
OUTPUT:
D - zbiór detektorów

1: $D \leftarrow \emptyset$
2: $T \leftarrow 0$
3: **repeat**
4: $t \leftarrow 0$
5: **repeat**
6: $c \leftarrow GenerateRandomDetector(l)$
7: $FallsInsideOtherDetector \leftarrow false$
8: **for** $d \in D$ **do**
9: **if** $distance(d,c) < r(d)$ **then**
10: $t \leftarrow t+1$
11: **if** $t >= \frac{1}{1-c_0}$ **then**
12: **return** D
13: **end if**
14: $FallsInsideOtherDetector \leftarrow true$
15: $T \leftarrow 0$
16: **break**
17: **end if**
18: **end for**
19: **until** $\neg FallsInsideOtherDetector$
20: $r = min_{s \in S} distance(s,c) - R_s$
21: **if** $r > R_s$ **then**
22: $D \leftarrow D \cup \{\langle c, r \rangle\}$
23: **else**
24: $T \leftarrow T+1$
25: **end if**
26: **if** $T > \frac{1}{1-\text{maximum self coverage}}$ **then**
27: **exit**
28: **end if**
29: **until** $|D| = T_{max}$

Złożoność czasową algorytmu 3.5 można wyrazić jako:

$$O(|D| * |S|) \tag{3.3}$$

Proces monitorowania danych, podobnie jak w oryginalnym rozwiązaniu bazuje na funkcji odległości euklidesowej pomiędzy komórką a detektorem (algorytm 3.4).

Celem algorytmu V-detector jest przygotowanie takich detektorów, które maksymalnie pokryją przestrzeń nienależącą do komórek własnych. Wykorzystanie detektorów o zmiennych promieniach umożliwia pokrycie dużej powierzchni przy mniejszej liczbie detektorów. Dzięki parametrowi T_{max} możliwe jest sterowanie liczbą detektorów, co przekłada się na szybkość monitorowania. Natomiast parametr c_0 umożliwia dobór progu skuteczności wykrywania komórek obcych.

3.1.3 Algorytm detektorów RST

Algorytm jest rozszerzeniem algorytmu V-detector i jest inspirowany teorią zbiorów przybliżonych (Pawlak, 1991). Modyfikacja dotyczy rozbudowania procesu generowania detektorów i ich monitorowania o użycie dwóch zbiorów detektorów: D_{upp} i D_{low}, zamiast jednego.

W procesie generowania detektorów (algorytm 3.6) używane są dwa zbiory komórek własnych: pierwszy z oryginalnymi komórkami o promieniu $R_{s_{upp}}$ oraz drugi z komórkami z granicą tolerancji o promieniu $R_{s_{low}}$. Zbiór komórek z granicą tolerancji powstaje na podstawie oryginalnego zbioru komórek własnych, a promień ich wartości zwiększany jest o zadaną wartość progową. Na podstawie powyższych zbiorów tworzone są zbiory detektorów:

- zbiór D_{upp} jest generowany za pomocą algorytmu V-detector,
- zbiór D_{low} powstaje na podstawie zbioru D_{upp} przez skrócenie promienia każdego z detektorów o różnicę promieni między komórkami własnymi ze zbiorów $R_{s_{upp}}$ i $R_{s_{low}}$ (r_{diff}).

Algorytm 3.6 Algorytm detektorów RST - generowanie detektorów

INPUT:
D_{upp} - detektory wygenerowane algorytmem V-detector promieniem $R_{s_{upp}}$
$R_{s_{upp}}$ - promień komórki własnej
$R_{s_{low}}$ - promień komórki własnej z granicą tolerancji
$R_{s_{low}} > R_{s_{upp}}$
OUTPUT:
D_{upp} - zbiór detektorów
D_{low} - zbiór bardziej tolerancyjnych detektorów

1: $D_{low} \leftarrow \emptyset$
2: $R_{diff} \leftarrow R_{s_{low}} - R_{s_{upp}}$
3: **for** $\langle c, r \rangle \in D_{upp}$ **do**
4: $\quad D_{low} \leftarrow D_{low} \cup \{\langle c, r - r_{diff} \rangle\}$
5: **end for**

Złożoność czasową algorytmu 3.6 można wyrazić jako:

$$O(|D| * |S|) \tag{3.4}$$

W trakcie monitorowania danych (algorytm 3.7) komórki są weryfikowane za pomocą bardziej tolerancyjnego zbioru D_{low}. Jeśli detektory ze zbioru D_{low} rozpoznają komórkę, jest ona uznana za obcą. W przeciwnym wypadku jest ona weryfikowana przez detektory ze zbioru D_{upp}. Jeżeli komórka nie zostanie rozpoznana przez detektor ze zbioru D_{upp}, oznacza to, że jest to komórka własna. W przeciwnym razie komórka zostaje uznana za niepewną, co wiąże się z jej dalszą analizą (np. przez algorytm klasyfikujący).

Algorytm 3.7 Algorytm detektorów RST - monitorowanie

INPUT:
S - zbiór monitorowanych komórek
D_{upp} - zbiór detektorów
D_{low} - zbiór bardziej tolerancyjnych detektorów
OUTPUT:
S_{self} - zbiór komórek rozpoznanych jako własne
$S_{nonself}$ - zbiór komórek rozpoznanych jako obce
$S_{uncertain}$ - zbiór komórek do dalszej analizy

1: $S_{self} \leftarrow \emptyset; S_{nonself} \leftarrow \emptyset; S_{uncertain} \leftarrow \emptyset$
2: **for** $s \in S$ **do**
3: **if** $\exists_{d \in D_{low}} match(d,s)$ **then**
4: $S_{nonself} \leftarrow S_{nonself} \cup \{s\}$
5: **else**
6: $S_{uncertain} \leftarrow S_{uncertain} \cup \{s\}$
7: **end if**
8: **end for**
9: **for** $s \in S_{uncertain}$ **do**
10: **if** $\nexists_{d \in D_{upp}} match(d,s)$ **then**
11: $S_{uncertain} \leftarrow S_{uncertain} \setminus \{s\}$
12: $S_{self} \leftarrow S_{self} \cup \{s\}$
13: **end if**
14: **end for**

3.2 Eksperymenty

W poniższej części opisano eksperymenty, które badają wpływ dyskretyzacji atrybutów na skuteczność wybranych algorytmów selekcji negatywnej. Badania zostały

przeprowadzone na kilku zbiorach danych z wykorzystaniem różnych algorytmów dyskretyzacji.

Implementacje algorytmów zostały zrealizowane w języku Python 3.8 z wykorzystaniem modułów numpy i pandas oraz uruchomione na środowisku Arch Linux.

Tabela 3.1: Charakterystyka zbiorów

Nazwa	**Skrót**	**Licz. atrybutów**	**Licz. obiektów**	**Dystrybucja klas**
Wine	Wine	13	178	59:71:48
Mammographic Mass	Mamm	5	961	516:445
KDDCup'99 (10%)	Kdd	41	499020	97277:396743

Do badań użyto trzech zbiorów danych (tabela 3.1) dostępnych na repozytorium UCI Repository (`https://archive.isc.uci.edu/ml`): Wine, Mammographic Mass i KDDCup'99 (10%).

Zbiór Wine posiada trzynaście atrybutów i trzy klasy decyzyjne. W eksperymencie jako komórki własne uznano elementy należące do klasy o wartościach 1. Pozostałe elementy są traktowane jako komórki obce.

Zbiór Mamm ma pięć atrybutów i dwie klasy decyzyjne. Jako komórki obce traktowane są dane wskazujące na guz złośliwy. Mimo że, klasyfikacja algorytmami NSA powyższego zbioru nie daje najlepszych rezultatów, to został on umieszczony w przeglądzie ze względu na możliwość sprawdzenia, jak dyskretyzacja cech może wpłynąć na bardziej problematyczne dane.

Ostatni zbiór Kdd jest zbiorem największym ze względu na ilość danych oraz liczbę atrybutów. Za anomalię w przypadku tego zbioru przyjęto wszystkie próby ataków sieciowych. W tabeli 3.1 liczebność zbiorów komórek własnych została oznaczona za pomocą podkreślenia.

Każdy ze zbiorów został znormalizowany w taki sposób, aby wartości atrybutów reprezentowały liczby rzeczywiste ze zbioru [0,1]. Następnie podzielono każdy zbiór na część treningową i testową. Proporcje podziału zostały ustalone empirycznie i dostosowane do zbioru indywidualnie. Komórki określone jako własne z części treningowej zostały użyte do generowania detektorów, natomiast cała część testowa została użyta do monitorowania skuteczności wygenerowanych detektorów. Warto zaznaczyć, że taki podział danych jest istotny przy zagadnieniu wykrywania anomalii w ruchu sieciowym, gdy nie wszystkie struktury komórek własnych i obcych są znane. Na koniec pięciokrotnie uruchomiono algorytmy dla danych niezdyskretyzowanych i uśredniono wyniki.

W kolejnym kroku eksperymentów dokonano dyskretyzacji danych z wykorzystaniem narzędzi Weka (`https://www.cs.waikato.ac.nz/ml/weka/`) oraz RSES (`https://www.mimuw.edu.pl/~szczuka/rses/start.html`). Narzędzie RSES wykorzystuje zbiory przybliżone (Pawlak, 1991) do wykonania cięć, według których generowane są przedziały wartości. Dzięki temu zachowane zostają różnice pomiędzy obiektami z różnych klas zbiorów, a zestaw cięć jest ograniczony do minimum (Bazan, Nguyen, Nguyen, Synak i Wróblewski, 2000). W ramach eksperymentu zdyskretyzowano dane według metody lokalnej i/lub globalnej. Metoda globalna dyskretyzuje dane w odniesieniu do całego zbioru danych. Używa jednego zestawu interwałów w ramach jednego zadania klasyfikacyjnego. Z kolei metoda lokalna tworzy różne zestawy interwałów dla pojedynczego atrybutu. Warto zwrócić uwagę na liczba przedziałów wygenerowanych przez narzędzie RSES. Dla zbioru Mammographic Mass z ponad siedemdziesięciu unikalnych wartości na atrybucie Age udało się zejść odpowiednio do 38 przedziałów dla metody globalnej i 46 przedziałów dla metody lokalnej. Na podstawie zaproponowanych podziałów można wywnioskować, że atrybut Age nie jest dobrym rozróżnikiem klas decyzyjnych. Z kolei na zbiorze KDDCup'99 10% metoda lokalna zaproponowała cięcia na 26 atrybutach generując od 2 do 38 przedziałów (w zależności od atrybutu). W przypadku zbioru Wine przedziały wygenerowane przez metody lokalną i globalną są podobne i dotyczą trzech atrybutów: Alcohol (trzy przedziały), Magnesium (dwa przedziały) i Flavonoids (dwa przedziały).

W przypadku narzędzia Weka dokonano dyskretyzacji na wybranych atrybutach rozpatrywanych zbiorów z podziałem na 5 i 10 przedziałów wartości. Jako metodę dyskretyzacji wybrano metodę nienadzorowaną. Podczas dyskretyzacji uwzględniono również tworzenie przedziałów w sposób binary dla każdego atrybutu oraz bez tego podziału. W wyniku tworzenia przedziałów w sposób binarny, w zbiorach tworzone są nowe atrybuty dla każdego interwału określające w sposób binarny podział wartości. W zestawieniu wyników zastosowanie tej metody oznaczono za pomocą kolumny *atrybuty binarne*. Jeśli użycie metody dyskretyzacji utworzyło nowe atrybuty z przedziałami wartość w kolumnie *atrybuty binarne* wstawiono *tak*. W przeciwnym wypadku użyto *nie*.

Dla każdego algorytmu pięciokrotnie uruchomiono testy na zbiorach po dyskretyzacji, a następnie uśredniono wyniki. Użyto takich samych ustawień i podziałów na zbiory testowe i treningowe, jak w przypadku testów na danych przed dyskretyzacją. W obu przypadkach (przed i po dyskretyzacji) użyto takich samych metryk mierzących skuteczność algorytmu: metryki dokładności (ang. *accuracy*) dla algorytmów RNS i detektorów V-detector oraz współczynnika wyników fałszywie pozytywnych (ang. *false alarm*) dla algorytmu z detektorami RST. Wyniki zostały zaprezentowane w tabelach w poszczególnych podrozdziałach. Rezultaty dla danych niezdyskretyzowanych zostały oznaczone podkreśleniem. Brak wartości oznaczono znakiem –.

3.2.1 Algorytm RNS

W przypadku zbioru Wine zbadano skuteczność algorytmu RNS dla detektorów o promieniu od 1,4 do 3,5. Najwyższy wynik wynoszący 0,82 algorytm osiągnął dla promienia 1,5, a powyżej wartości 1,9 algorytm nie był w stanie dopasować detektorów do wylosowanych zestawów testowych. Po dokonaniu dyskretyzacji wartości trzech atrybutów uzyskano wyższą skuteczność dla promienia 1,5 i 1,9 w przypadku zastosowania metody lokalnej lub globalnej (tabela 3.2). Warto zauważyć, że po dokonaniu dyskretyzacji z podziałem na atrybuty binarne, skuteczność algorytmu dla podanych wyżej wartości promienia spada z uwagi na fakt utworzenia nowych atrybutów. Z drugiej strony zastosowanie tej metody umożliwia uruchomienie algorytmu dla promienia, który w przypadku oryginalnych danych (sprzed dyskretyzacji) nie jest możliwe i uzyskanie wyższej detekcji (np. $r = 3,5$).

Podobne regularności można zaobserwować przy zbiorze Mammographic Mass (tabela 3.3).

Algorytm uruchomiono dla promienia o wartościach z zakresu od 0,45 do 1,5. W tabeli 3.3 zostały umieszczone rezultaty o najwyższej mierze dokładności dla każdej z metod. Podobnie jak w przypadku zbioru Wine najwyższą dokładność uzyskano przy zastosowaniu metody lokalnej i globalnej. Co ciekawe, mimo że atrybut Age nie jest najlepszym rozróżnikiem klas, to na danych zdyskretyzowanych uzyskano nieznacznie wyższe wyniki niż na danych przed dyskretyzacją.

Z kolei na zbiorze KDDCup'99 10% dokonano dyskretyzacji siedmiu atrybutów metodą nienadzorowaną z podziałem na 5 i 10 przedziałów oraz metodą lokalną. Rezultaty z przeprowadzonych testów były nieznacznie niższe od wyników zwróconych przy uruchomieniu algorytmu na danych oryginalnych i spadły z 0,99 do 0,97. Przetestowano również cięcia na 26 atrybutach zaproponowanych przez metodę lokalną. Dyskretyzacja wartości tych atrybutów znacznie pogorszyła wyniki klasyfikacji algorytmu spadając z 0,99 do 0,28. Podobnie jak w przypadku mniejszych zbiorów, zaobserwowano spadek detekcji komórek obcych przy zastosowania atrybutów binarnych. Zwiększenie wartości promienia przy ponownych testach pozwoliło uzyskać zadowalające wyniki na poziomie 0,99.

Tabela 3.2: Działanie algorytmu RNS na zbiorze Wine po dyskretyzacji trzech atrybutów

Metoda	**Liczba przedziałów**	**Atrybuty binarne**	*r*	**Skuteczność**
-			1,5	0,82
-			1,9	-
bez nadzoru	5	tak	1,5	0
bez nadzoru	5	nie	1,5	0,75
bez nadzoru	10	nie	1,5	0,75
lokalna	2-3	nie	1,5	0,88
lokalna	2-3	nie	1,9	0,99
bez nadzoru	5	tak	1,9	0,84
bez nadzoru	10	tak	1,9	0
bez nadzoru	5	nie	1,9	0,55
bez nadzoru	10	nie	1,9	-
globalna	2-3	nie	1,5	0,99
globalna	2-3	nie	1,9	0,99
bez nadzoru	10	tak	3,5	0,97

r - promień detektora, Skuteczność liczona jest metryką dokładności

Tabela 3.3: Skuteczność algorytmu RNS na zbiorze Mammographic Mass po dyskretyzacji atrybutu Age

Metoda	**Liczba przedziałów**	**Atrybuty binarne**	*r*	**Skuteczność**
-			0,45	0,5
-			1	-
bez nadzoru	5	tak	1,5	0
bez nadzoru	5	nie	0,45	0,52
bez nadzoru	10	nie	0,45	0,53
lokalna	46	nie	0,45	0,55
globalna	38	nie	0,45	0,53
bez nadzoru	5	tak	0,45	0,01
bez nadzoru	10	tak	0,45	0,09
bez nadzoru	5	tak	1	0,47
bez nadzoru	10	tak	1	-
bez nadzoru	10	tak	1,5	0,54

r - promień detektora, Skuteczność liczona jest metryką dokładności

3.2.2 Algorytm V-detector

Algorytm V-detector przetestowano z następującymi ustawieniami:

- Wine - r od 0,01 do 0,1, T_{max}=20,
- Mammographic Mass - r=0,45, T_{max}=100,
- KDDCup'99 10% - r=1,5, T_{max}=100,

gdzie r to promień komórki własnej, T_{max} maksymalna liczba detektorów. Oczekiwane pokrycie c_0 wynosiło 0,9. Wyniki eksperymentu przedstawiono zbiorczo dla wszystkich zbiorów w tabeli 3.4.

Tabela 3.4: Skuteczność algorytmu V-detektor po dyskretyzacji danych

Metoda	**Liczba przedziałów**	**Atrybuty binarne**	**Skuteczność**
		Zbiór Wine	
-	-	-	0,84
bez nadzoru	5	tak	0,29
bez nadzoru	5	nie	0,78
bez nadzoru	10	tak	0,40
bez nadzoru	10	nie	0,79
lokalna	2-3	nie	0,59
globalna	2-3	nie	0,83
		Zbiór Mamm	
-	-	-	0,70
bez nadzoru	5	tak	0,60
bez nadzoru	5	nie	0,68
bez nadzoru	10	tak	0,47
bez nadzoru	10	nie	0,69
lokalna	2-3	nie	0,65
globalna	2-3	nie	0,74
		Zbiór Kdd	
-	-	-	0,98
bez nadzoru	5	tak	0,60
bez nadzoru	5	nie	0,98
bez nadzoru	10	tak	0,95
bez nadzoru	10	nie	0,98
lokalna (7)	2-26	nie	0,98
lokalna (26)	2-34	nie	0,27

Skuteczność liczona jest metryką dokładności

W przypadku algorytmu V-detector dla mniejszych zbiorów najlepsze rezultaty w klasyfikacji dała metoda globalna. Dyskretyzacja atrybutów tą metodą pozwala uzyskać wynik zbliżony do tego, który otrzymujemy na danych oryginalnych. W przypadku zbioru KDDCup'99 10% dyskretyzacja siedmiu atrybutów nie wpłynęła znacząco na działanie algorytmu. Zastosowanie cięć zbiorów na ponad połowie atrybutów spowodowało znaczne obniżenie jakości klasyfikacji.

3.2.3 Algorytm detektorów RST

Głównym celem algorytmu detektorów RST jest zmniejszenie liczby rozpoznań komórek własnych jako obcych. W związku z tym jako miarę skuteczności algorytmu wybrano współczynnik wyników fałszywie pozytywnych. Na podstawie otrzymanych rezultatów (tabela 3.5) można wnioskować, że dyskretyzacja danych dla małych zbiorów zminimalizowała błędną klasyfikację komórek własnych jako obcych, zwiększając tym samym liczbę komórek niepewnych, wymagających dalszej analizy. W przypadku zbioru KDDCup'99 10% wartość metryki utrzymało się na poziomie 0, niezależnie od zastosowanej metody dyskretyzacji.

Tabela 3.5: Skuteczność algorytmu detektorów RST po dyskretyzacji danych

Metoda	**Liczba przedziałów**	**Atrybuty binarne**	**Skuteczność**	**Komórki niepewne**
		Zbiór Wine		
-	-	-	0,56	0,02
bez nadzoru	5	tak	0,30	0,03
bez nadzoru	5	nie	0,55	0,02
bez nadzoru	10	tak	0,09	0,02
bez nadzoru	10	nie	0,55	0,02
lokalna	2-3	nie	0,14	0,05
globalna	2-3	nie	0,30	0,03
		Zbiór Mamm		
-	-	-	0,2	0,47
bez nadzoru	5	tak	0,07	0,41
bez nadzoru	5	nie	0,04	0,57
bez nadzoru	10	tak	0,09	0,30
bez nadzoru	10	nie	0,04	0,58
lokalna	2-3	nie	0,05	0,50
globalna	2-3	nie	0,01	0,67

Skuteczność jest określana miarą współczynnika wyników fałszywie pozytywnych

Podsumowanie

W powyższej pracy przedstawiono wstępne badania nad wpływem dyskretyzacji cech na działanie algorytmów selekcji negatywnej.

Dyskretyzacja cech jest procesem polegającym na zmianie atrybutów ciągłych na dyskretne. W związku z tym, może wpłynąć na utratę istotnych informacji opisujących obiekty z różnych klas. Jednak, jak pokazały przeprowadzone w tej pracy eksperymenty, jeżeli obiekty z różnych klas mają bliskie wartości na danym atrybucie, to po dokonaniu dyskretyzacji danego atrybutu, różnice między obiektami mogą być bardziej widoczne. Dzięki temu skuteczność algorytmów selekcji negatywnej może wzrosnąć. Ponadto, dyskretyzacja kilku atrybutów na małych zbiorach (maksymalnie kilkanaście atrybutów) może nieznacznie poprawić działanie algorytmów NSA, nawet jeśli dany atrybut nie jest najlepszym wyróżnikiem klas. Przy dużych zbiorach (kilkadziesiąt atrybutów) dyskretyzacja wartości nieistotnych cech, nie wpływa znacząco na obniżenie jakości klasyfikacji. Niemniej zamiana wartości atrybutów na przedziały dla zbyt dużej liczby (np. połowy) atrybutów może doprowadzić do zupełnego zatarcia różnic między obiektami. W przeprowadzonych badaniach najlepsze rezultaty otrzymano na podstawie cięć zbiorów dokonanych metodą globalną narzędzia RSES. Metoda ta zachowuje informację o klasach obiektów. Można więc wnioskować, że algorytmy NSA zwracają bardziej satysfakcjonujące wyniki, analizując dane zdyskretyzowane metodą z nadzorem i podziałami dokonanymi na podstawie całego zbioru danych.

Badania mogą być w przyszłości rozszerzone o kolejne implementacje algorytmów selekcji negatywnej (uwzględniające również rozwiązania hybrydowe) oraz użycie dodatkowych metod dyskretyzacji, np. z nadzorem na danych wielowymiarowych.

Bibliografia

Aziz, A. S. A., Azar, A. T., Hassanien, A. E. i Hanafy, S. E.-O. (2014). Continuous features discretization for anomaly intrusion detectors generation. W: V. Snášel, P. Krömer, M. Köppen i G. Schaefer (red.), *Soft computing in industrial applications,* Springer, 209–221.

Bazan, J. G., Nguyen, H. S., Nguyen, S. H., Synak, P. i Wróblewski, J. (2000). Rough set algorithms in classification problem. W: L. Polkowski, S. Tsumoto i T. Y. Lin (red.), *Rough set methods and applications: New developments in knowledge discovery in information systems,* Springer, 49–88.

Chmielewski, A. (2017). Application of rough sets to negative selection algorithms. W: T. K. Dang, R. Wagner, J. Küng, N. Thoai, M. Takizawa i E. J. Neuhold (red.), *Future data and security engineering,* Springer, 381–394.

Dasgupta, D., i Forrest, S. (1995). Novelty detection in time series data using ideas from immunology. W: *Proceedings of 8th international conference on intelligent systems,* 6.

Esponda, F., Forrest, S. i Helman, P. (2003). The crossover closure and partial match detection. W: J. Timmis, P. J. Bentley i E. Hart (red.), *Artificial immune systems,* Springer, 249–260.

Forrest, S., Perelson, A. S., Allen, L. i Cherukuri, R. (1994). Self-nonself discrimination in a computer. W: *Proceedings of 1994 ieee computer society symposium on research in security and privacy,* IEEE, 202-212.

García, S., Luengo, J., Sáez, J. A., López, V. i Herrera, F. (2013). A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, *25*, 734-750.

Gonzalez, F., i Dasgupta, D. (2003). Anomaly detection using real-valued negative selection. *Genetic Programming and Evolvable Machines*, *4*, 384–403.

Hofmeyr, S. A. (1999). *An immunological model of distributed detection and its application to computer security*. Unpublished doctoral dissertation, The University of New Mexico.

Hońko, P. (2018). Adaptive positive-negative selection approach. *Journal of Physics: Conference Series*, *1061*, 012-020.

Jerne, N. K. (1973). Towards a network theory of the immune system. *Annals of Immunology*, *125*, 373-389.

Ji, Z., i Dasgupta, D. (2004). Real-valued negative selection algorithm with variable-sized detectors. W: K. Deb (red.), *Genetic and evolutionary computation – gecco 2004,* Springer, 287–298.

Ji, Z., i Dasgupta, D. (2007). Revisiting negative selection algorithm. *Evolutionary Computation*, *15*, 223–251.

Lasek, M., Lasek, W. i Pęczkowski, M. (2013). Od immunologii do modelowania, przetwarzania i analiz danych. *Informatyka Ekonomiczna*, *4*, 196–225.

Lucińska, M. (2010). Hybrid immune algorithm for many optima. W: L. Rutkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh i J. M. Zurada (red.), *Artifical intelligence and soft computing* Springer, 540–547.

Lydyard, P. M., Whelan, A. i Fanger, M. W. (2001). *Immunologia. krótkie wykĺady.* Warszawa: Wydawnictwo Naukowe PWN.

Matzinger, P. (1994). Tolerance, danger, and the extended family. *Annual Review of Immunology*, *12*, 991-1045.

Pawlak, Z. (1991). *Rough sets. Theoretical aspects of reasoning about data*. Dordrecht: Kluwer Academic Publishers Group.

Praczyk, T. (2010). Using real valued detectors in ship immune system. *Computing and Informatics*, *29*, 975–987.

Wierzchoń, S. T. (2001). *Sztuczne systemy immunologiczne. teoria i zastosowania.* Warszawa: Akademicka Oficyna Wydawnicza EXIT.