
Wydział Fizyki i Informatyki Stosowanej

## Praca magisterska

**Kamil Madej**

kierunek studiów: **Informatyka Stosowana**

specjalność: **Grafika Komputerowa i Przetwarzanie Obrazów**

# Sposoby zwiększania bezpieczeństwa systemu informatycznego przy pomocy aplikacji do zarządzania zasobami systemów informatycznych

Opiekun: **dr inż. Antoni Dydejczyk**

**Kraków, czerwiec 2018**

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i nie korzystałem ze źródeł innych niż wymienione w pracy.

.....

(czytelny podpis)

TODO

Kraków, czerwiec 2018

**Tematyka pracy magisterskiej i praktyki dyplomowej Kamila Madeja, studenta V roku studiów kierunku informatyka stosowana, specjalności grafika komputerowa i przetwarzanie obrazów**

Temat pracy magisterskiej: **Sposoby zwiększania bezpieczeństwa systemu informatycznego przy pomocy aplikacji do zarządzania zasobami systemów informatycznych**

| | |
|---|---|
| Opiekun pracy: | dr inż. Antoni Dydejczyk |
| Recenzenci pracy: | dr inż. TODO |
| Miejsce praktyki dyplomowej: | WFiIS AGH, Kraków |

**Program pracy magisterskiej i praktyki dyplomowej**

1. Omówienie realizacji pracy magisterskiej z opiekunem oraz przedstawicielem firmy IBM.
2. Zebranie i opracowanie literatury dotyczącej tematu pracy.
3. Praktyka dyplomowa:
   - Zapoznanie się z ideą Software Asset Management
   - Zapoznanie się z tematyką podatności systemów informatycznych
   - Analiza danych wejściowych
4. Opracowanie algorytmu umożliwiającego wykrywanie podatności w systemie
5. Przeprowadzenie testów, analiza wyników
6. Porównanie otrzymanych wyników z próbami przeprowadzonymi przez IBM
7. Analiza wyników, ich omówienie i zatwierdzenie przez opiekuna
8. Opracowanie redakcyjne pracy

Termin oddania w dziekanacie: TODO czerwca 2018

..... .....

(podpis kierownika katedry) (podpis opiekuna)

Merytoryczna ocena pracy przez opiekuna: TODO

Merytoryczna ocena pracy przez recenzenta: TODO

# Spis treści

**Wstęp** . . . . . **12**

**Cel pracy** . . . . . **13**

**1 Software Asset Management** . . . . . **15**

1.1 Wstęp . . . . . 15

**2 Reprezentacja danych** . . . . . **17**

2.1 Dane publikowane przez National Institute of Standards and Technology 17

2.2 Common Vulnerabilities and Exposures . . . . . 18

**3 Implementacja** . . . . . **21**

3.1 Język programowania R . . . . . 21

3.2 Implementacja PNN . . . . . 22

3.3 Implementacja FPA . . . . . 25

3.4 Integracja algorytmów . . . . . 26

**4 Analiza użyteczności** . . . . . **29**

4.1 Badanie wpływu doboru parametrów . . . . . 29

4.1.1 Wybór jądra estymacji . . . . . 29

4.1.2 Stopnie swobody parametru wygładzania . . . . . 30

4.1.3 Siła modyfikacji parametru wygładzania . . . . . 36

4.1.4 Strojenie parametrów FPA . . . . . 37

4.1.5 Wpływ normalizacji na klasyfikację . . . . . 39

4.1.6 Wybór parametru wygładzania . . . . . 40

4.1.7 Finalny algorytm . . . . . 42

4.2 Porównanie z istniejącymi algorytmami . . . . . 42

**5 Podsumowanie** . . . . . **45**

**6 Kierunki dalszego rozwoju** . . . . . **47**

**Bibliografia** . . . . . **49**

**Spis rysunków** . . . . . **51**

**Spis tabel** . . . . . **52**

**Spis listingów** . . . . . **53**

**Dodatki** . . . . . **53**

# Wstęp

Software Asset Management jest pojęciem ściśle związanym z dynamicznie rozwijającym się rynkiem oprogramowania. Pojawiające się różne rodzaje licencji określające sposób, w jaki można korzystać z poszczególnych rozwiązań, stworzyły przestrzeń dla firm takich, jak IBM, na stworzenie aplikacji ułatwiającej zarządzanie licencjami w systemach informatycznych przedsiębiorstw. W dzisiejszych realiach poważne przedsiębiorstwa muszą działać transparentnie, nie mogą sobie bowiem pozwolić na oskarżenia o nadużywanie lub nieprzestrzeganie wymogów licencyjnych. Wiąże się to nie tylko z kosztami ponoszonymi wskutek naliczanych kar, ale też ze stratami wizerunkowymi.

Każdego dnia pojawiają się nowe podatności i zagrożenia, na jakie wystawione są systemy informatyczne. Twórcom oprogramowania zależy, aby ich produkt był jak najbardziej na nie odporny. Wymusza to publikowanie nowych wersji oprogramowania, w którym poprawiono błędy i załatano luki umożliwiające niepożądane działanie systemu.

Niniejsza praca jest próbą połączenia powyższych pojęć w jednym narzędziu. Wykorzystując informacje gromadzone w aplikacji IBM Big Fix Inventory, można informować użytkownika o tym, że w jego systemie informatycznym zainstalowane jest oprogramowanie, w którym wykryta została podatność. Właściciel systemu na podstawie informacji o rodzaju podatności i o stopniu zagrożenia z nią związanego, może zdecydować jakie akcje należy podjąć, by zapewnić stabilność i bezpieczeństwo systemu.

Algorytm opracowany w ramach pracy umożliwia wykrycie w systemie informatycznym wersji oprogramowania znajdujących się na liście obciążonych podatnościami, publikowanej przez amerykańską agencję NIST. Jego działanie polega na dopasowaniu informacji o oprogramowaniu zainstalowanym i używanym w systemie, zebranych przez IBM Big Fix Inventory, z informacjami zebranymi przez NIST.

# Cel pracy

Niniejsza praca stanowi badanie możliwości wykorzystania różnych algorytmów porównania tekstu oraz ich połączeń w celu wykrywania oprogramowania, które może powodować podatność systemu informatycznego na zagrożenia. W ramach pracy zostaną wykonane następujące czynności:

- Analiza poszczególnych algorytmów pod kątem danych wejściowych.
- Na podstawie powyższej analizy zostanie opracowany algorytm pozwalający na połączenie dostępnych informacji o podatnościach występujących oprogramowaniu informatycznym oraz informacji o oprogramowaniu zainstalowanym w badanym systemie.
- Implementacja algorytmu w języku Python
- Przeprowadzenie testów skuteczności algorytmu
- Analiza otrzymanych wyników

Podczas opracowywania algorytmu szczególna uwaga zwrócona będzie w kierunku skuteczności wykrycia zagrożonej wersji oprogramowania zainstalowanej w systemie informatycznym monitorowanym przez narzędzie IBM Big Fix Inventory. Efektem pracy będą przedstawione wnioski dotyczące efektywności poszczególnych algorytmów porównania tekstu oraz ich połączeń opracowanych w ramach pracy. Wnioski zostaną wyciągnięte na podstawie wyników testów poszczególnych rozwiązań.

# 1 Software Asset Management

## 1.1 Wstęp

Eksploracja danych zajmuje się analizą dużych zbiorów danych dotyczących różnych zjawisk. Mogą to być dane finansowe, wyniki obserwacji fizyczny czy dane pacjentów. Danych najczęściej jest zbyt dużo aby mógł je przeanalizować człowiek, dlatego też istnieje zapotrzebowanie na automatyczne algorytmy operujące na całym zbiorze danych. Zbiór zwykle przedstawiany jest w postaci macierzy której kolumnami są cechy opisujące zjawisko, a wiersze to kolejne obserwacje. Analizowane dane mogą mieć następujące typy:

- Liczbowe - Numeryczny wynik jakiegoś pomiaru, na przykład wzrost, waga
- Kategoryczne - Wartość z ograniczonego zbioru, na przykład rozróżnienie kobiet i mężczyzn, czy grupy krwi
- Tekstowe - Bloki tekstu opisujące zjawisko, na przykład treść wiadomości e-mail

# 2 Reprezentacja danych

## 2.1 Dane publikowane przez National Institute of Standards and Technology

National Institute of Standards and Technology (NIST) jest amerykańską agencją federalną, która zajmuje się szeroko pojętą metrologią. Wśród obszarów zainteresowań NIST znalazły się również podatności w systemach informatycznych[?]. National Vulnerability Database (NVD) jest wynikiem prac prowadzonych przez Instytut, które miały na celu usystematyzowanie i jak najlepsze opisanie podatności oraz ich wpływu na systemy informatyczne. NVD jest rządowym repozytorium, które za pomocą protokołu Secutiry Content Automation Protocol (SCAP) pozwala na ustandaryzowane automatyzowanie zarządzania podatnościami. NVD zawiera listy kontrolne bezpieczeństwa, wady oprogramowania związane z bezpieczeństwem, błędy konfiguracyjne, nazwy produktów oraz ocenę ich wpływu na system. Dane w NVD aktualizowane są na bieżąco, zaraz po przeanalizowaniu ich przez zespół ekspercki[?].

W protokole SCAP wykorzystuje się system Common Vulnerabilities and Exposures (CVE), który jest zarządzany i rozwijany przez firmę Mitre Corporation. Bazy danych NVD zawierają indeksy CVE rozszerzone dodatkowo o ocenę podatności według metryk Common Vulnerability Scoring System (CVSS). Pozwala to na podjęcie automatycznej decyzji o akcjach związanych z pojawieniem się podatności w systemie informatycznym. CVE znajdujące się w bazie NVD mogą znajdować się w jednym z siedmiu stanów:

Received - CVE został niedawno opublikowany do słownika CVE i został odebrany przez NVD

Awaiting Analysis - CVE został przeznaczony do analizy eksperckiej ze strony NVD, która zwykle zajmuje do 24 godzin

Undergoing Analysis - CVE jest obecnie analizowane przez ekspertów NVD

Analyzed - CVE zostało całkowicie przeanalizowane oraz wszystkie dodatkowe dane zostały do niego przypisane. Każda z analiz może być opisana jednym z trzech podtypów:

Initial - używany by pokazać, że została przeprowadzona dopiero pierwsza analiza danego CVE

Modified - używana by pokazać, że została przeprowadzona analiza związana z modyfikacją danych w CVE

Reanalysis - używany by pokazać, że ponowna analiza została przeprowadzona z powodu innego niż modyfikacja CVE

Modified - CVE było poprawione przez źródło, które je opublikowało, co oznacza, że wyniki analizy NVD mogą być już nieaktualne

Deferred - jeśli CVE znajduje się w tym planie oznacza to, że nie jest planowana jego analiza

Rejected - CVE w tym stanie znajduje się w bazie danych NVD, nie wyświetla się jednak w wynikach wyszukiwania

Dodatkową kategorią indeksu CVE jest kategoria "Reserved". CVE występujące w słowniku, które są oznaczone takim polem, nie zostają włączone do bazy danych NVD[?].

## 2.2 Common Vulnerabilities and Exposures

Common Vulnerabilities and Exposures jest systemem zapewniającym publiczne, darmowe informacje na temat ekspozycji i podatności systemów informatycznych. Celem stworzenia CVE było utworzenie standardu, który ułatwiłby dzielenie się informacjami pomiędzy różnymi narzędziami zajmującymi się tematyką cyberbezpieczeństwa. Dzięki CVE możliwa była komunikacja pomiędzy tymi narzędziami, co pozwoliło na bardziej efektywną walkę z atakami na systemy informatyczne.

W CVE podatność zdefiniowana jest jako słabość w logice obliczeniowej, znaleziona w oprogramowaniu lub komponentach sprzętowych, które, jeśli zostałyby odkryte, niekorzystnie wpłynęły by na poufność, integralność lub dostępność systemu. Naprawa podatności zwykle wymaga zmian w kodzie programu, czasem również zmian w specyfikacji. Ekspozycją (TODO przemyśleć tłumaczenie "exposure") jest błędna konfiguracja systemu lub błąd w oprogramowaniu, który daje dostęp do informacji i zdolności systemu, użytymi później przez hakera jako wytrych[2].

Lista Common Vulnerabilities and Exposures składa się CVE Entry, czyli wpisów CVE, które identyfikują poszczególne podatności. Każdy wpis na liście CVE składa

się z:

- CVE ID, który jest unikatowym identyfikatorem wpisu. Zapisany jest w formacie CVE-YYY-NNNN, gdzie YYYY jest liczbą oznaczającą rok publikacji wpisu, a NNNN jest numerem identyfikacyjnym w danym roku. pole NNNN składa się co najmniej z czterech cyfr, jednak od 2014 roku może ich być więcej niż cztery.
- Krótki opis podatności lub ekspozycji systemu, który może zwierać takie informacje, jak nazwa producenta podatnego oprogramowania.
- Odnośniki do innych, powiązanych wpisów CVE oraz wszelkie inne odnośniki, takie jak raporty i porady podane przez producenta.

Wpisy tworzone są przez zespół Mitre zajmujący się CVE, producentów oprogramowania, których aplikacje zawierają podatności oraz tak zwanych CVE Numbering Authority (CNA). CNA to organizacja zajmująca się dystrybuowaniem CVE ID. Ma ona ograniczone możliwości przypisywania CVE ID, dokładnie określone i udokumentowane, zwykle obejmujące programy lub sprzęt pochodzące od kilku producentów. Ograniczenia te stworzone są po to, by kompetencje zbyt wielu osób nie nachodziły na siebie, co mogłoby powodować chaos[3].

CVE ID jest polem wpisu CVE. Jest to unikatowy numer identyfikacyjny pozwalający jednoznacznie odróżnić od siebie wpisy na liście CVE. Składa się on z trzech podidentyfikatorów: identyfikatora listy, identyfikatora roku oraz identyfikatora wpisu w danym roku. Identyfikator listy może przyjmować dwie wartości:

- CVE - najczęściej używany, oznacza, że wpis widnieje na liście CVE
- CAN - pochodzący od słowa "candidate", czyli kandydat, oznacza, że w momencie publikacji listy eksperci z CVE jeszcze nie zatwierdzili tego wpisu, jako zweryfikowanej podatności. Od pewnego momentu niemożliwym stało się, by zarząd CVE zajmował się każdym wpisem z osobna, gdyż było ich zbyt dużo, w związku z czym zaprzestano stosować identyfikatora CAN.

Kolejny podidentyfikator składa się z czterech cyfr i oznacza rok, w którym wpis został opublikowany. Możliwa jest sytuacja, że podatność została wykryta wcześniej, ale przy przypisywaniu CVE ID liczy się rok, w którym to ID zostało przypisane.

Ostatni człon CVE ID zbudowany jest z co najmniej czterech cyfr oznaczających identyfikator unikatowy w skali roku. Od 2014 roku, w związku z rosnącą ilością wpisów na liście CVE, przyjmuje się, że ten człon może posiadać więcej niż cztery cyfry.

Opis wpisu CVE stworzony jest zwykle przez zespół CVE, organizację CNA lub osoby indywidualne zgłaszające odkrytą podatność. Opisy powinny zapewniać informacje na temat produktu w którym została wykryta podatność, identyfikatora wersji tego produktu oraz jego dostawcy. Powinien też zostać wyszczególniony typ podatności, jej wpływ na systemy informatyczne, a także informacje jak głębokiego dostępu do sytemu potrzebuje haker, by móc skorzystać z danej podatności. Możliwe jest też wskazanie części kodu lub konkretnego komponentu odpowiadającego za podatność. Niestety, autor wpisu CVE nie zawsze jest w posiadaniu wszystkich informacji. Wynika to z faktu, że nie wszystkie z powyższych informacji są dostępne publicznie, co znacząco utrudnia wykonanie pełnego, oddającego całość problemu opisu. Z uwagi na znaczenie tego pola z punktu widzenia niniejszej pracy magisterskiej, zostanie ono dodatkowo przeanalizowane w osobnej sekcji.

# 3 Implementacja

## 3.1 Język programowania R

W pracy wykorzystano interpretowany język programowania R. R to język służący do przeprowadzania obliczeń i analiz statystycznych i wizualizacji ich wyników. Obecnie jest jednym z najczęściej wykorzystywanych języków w analizie danych i uczeniu maszynowym zaraz po Pythonie. Największą zaletą tego języka jest ogromna ilość dostępnych bibliotek i narzędzi, duża ilość algorytmów z dziedzin statystyki i analizy danych jest już zaimplementowana i dostępna w postaci pakietów. Kolejną zaletą języka R jest fakt że jest dostępny na otwartej licencji GNU. Ilość dostępnych bibliotek i wbudowanych funkcjonalności umożliwiają szybkie pisanie algorytmów, ich testowanie i prototypownie, jednakże niska wydajność języka i brak funkcjonalności związanych z bezpieczeństwem powodują że R nie nadaje się do budowy finalnych aplikacji. R jest językiem który znacznie lepiej sprawdzi się w pracy naukowej. Środowiskiem programistycznym jakie wykorzystano jest darmowe RStudio.


Rysunek 1: Logo języka R

## 3.2 Implementacja PNN

Probabilistyczna sieć neuronowa została zrealizowana jako klasa referencyjna języka R. Taki zabieg ma na celu umożliwienie przechowywania stanu sieci, aby pozwolić na wielokrotne wykorzystanie gotowego modelu do klasyfikacji wielu elementów. Zaimplementowana sieć posiada możliwość wyboru następujących parametrów:

- Parametr wygładzania:
  1. W formie skalara, jeden parametr wygładzania dla wszystkich elementów.
  2. W formie wektora o długości odpowiadającej ilości wymiarów, aby uzyskać inne wygładzanie dla każdego stopnia swobody.
  3. W formie wektora o długości równej liczbie klas, aby zapewnić różne wygładzanie dla każdej klasy.
  4. W formie macierzy aby uzyskać różne parametry dla każdej klasy i każdego wymiaru.
- Jądro estymacji, wybierane jako argument znakowy z listy dostępnych jąder wymienionych w tabeli **??**
- Siła modyfikacji parametru wygładzania. W przypadku podania siły 0 obliczenia związane z modyfikacją nie zostają wykonane.

Oprócz wspomnianych parametrów do konstruktora sieci neuronowej przekazujemy również zbiór danych w postaci macierzy oraz wektor etykiet klas. Po inicjalizacji klasa reprezentująca sieć udostępnia metodę classify zwracającą etykiety klas umożliwiającą klasyfikację nowych danych. Funkcja classify jest zwektoryzowana, można przekazywać do niej całe wektory elementów do zaklasyfikowania, w takim wypadku zostanie zwrócony wektor etykiet klas do których przyporządkowano odpowiednie elementy. Wektoryzacja funkcji znacznie przyśpiesza proces klasyfikacji dla większej ilości elementów. Funkcja classify na każdym z elementów wywołuję funkcję get_class_score, a następnie wybiera etykietę klasy dla której zwrócony został najwyższy wynik. Funkcja get_class_score oblicza wartość estymatora gęstości prawdopodobieństwa stworzonego dla danej klasy w badanym punkcie. Wykorzystanie klasy do implementacji algorytmu umożliwia zmniejszenie ilości argumentów przekazywanych pomiędzy funkcjami, co optymalizuje ilość wykorzystywanej pamięci.

Listing 1: Kod funkcji classify i get_class_score

```
get_class_score = function(x, class_index) {
  kernel_density=function(cls, mod){
    univariate=kernel((x - cls)/(smooth[,class_index] * mod))
    return(colProds(univariate)/mod)
  }
  kernels = mapply(kernel_density,
                   data.frame(test_set[[class_index]]),
                   modification[[class_index]])
  m=ncol(test_set[[class_index]])
  return(rowSums2(to_mat(kernels))/(prod(smooth[,class_index]) * m))
}

classify = function(x) {
  x <- t(to_mat(x))
  class_scores =sapply(seq_along(test_set),
                       Curry(get_class_score, x = x))
  return(names(test_set)[max.col(to_mat(class_scores))])
}
```

Klasa PNN posiada również implementację algorytmu modyfikacji parametrów wygładzania. Jeżeli podana zostanie siła modyfikacji większa od zera, wykonane zostaną obliczenia współczynników modyfikacji dla każdego z elementów zbioru trenującego, a następnie zostaną one użyte do budowania estymatorów jądrowych poszczególnych klas.

Poza wspomnianymi funkcjonalnościami do wykorzystania z klasą PNN zaimplementowano funkcję pozwalającą na wyznaczenie parametrów wygłądzania metodą plug-in dla dowolnego jądra wymienionego w tabeli **??**. Wykorzystano trzystopniowy algorytm plug-in, do estymacji pochodnych funkcji gęstości wykorzystano jądro Cauchy'ego. Parametry wygładzania obliczane mogą zostać dla całego zbioru danych, bądź dla każdej z klas z osobna.

Dodatkowo w celu ułatwienia prezentacji wyników zaimplementowano funkcję umożliwiającą rysowanie wykresów estymatorów jądrowych wykorzystywanych w klasie PNN. Funkcja wyświetla trójwymiarowy przekrój estymatora dla dwóch zmiennych, a pozostałe zostają ustawioną na średnią wartość w obrębie klasy.


Rysunek 2: Przykład wykresu przekroju estymatora jądrowego

## 3.3 Implementacja FPA

Flower pollination algorithm zaimplementowano w postaci funkcji przyjmującej funkcję do optymalizacji, wymiarowość optymalizowanej zmiennej, rozmiar populacji i maksymalną ilość iteracji które należy wykonać. Do generacji liczb losowych z rozkładu jednostajnego wykorzystano funkcje wbudowane, natomiast do generacji lotów Levy'ego wykorzystano algorytm Mantegna zaproponowany w [?].

Listing 2: Algorytm Mantegana do generacli lotów Levy'ego

```
Levy <- function(d, l, scale) {
    sigma <- (gamma(1 + l) * sin(pi * l/2)
                /(gamma((1 + l)/2) * l * 2^((l - 1)/2)))^(1/l)
    u <- rnorm(d, sd = sigma)
    v <- rnorm(d)
    step <- u/(abs(v))^(1/l)
    return(scale * step)
}
```

Generacja odbywa się poprzez obliczenie ilorazu zmiennych pochodzących z dwóch rozkładów normalnych, u i v. Rozkład u ma odchylenie standardowe określone w parametrze sigma, natomiast rozkład v ma odchylenie standardowe równe 1.

## 3.4 Integracja algorytmów

Do oceny jakości klasyfikacji w niniejszej pracy wykorzystano dziesięciokrotną walidację krzyżową. W procesie tym zbiór danych dzielony jest na dziesięć równych części, a następnie dziesięciokrotnie przeprowadzana jest klasyfikacja w której jako zbiór treningowy wykorzystuje się dziewięć z dziesięciu części, i sprawdza się jakość klasyfikacji pozostałej jednej dziewiątej. Jakość klasyfikacji mierzona jest jako iloczyn ilości poprawnie zaklasyfikowanych elementów i liczności zbioru klasyfikowanego.

FPA wymaga sformułowania funkcji celu, zaimplementowano wersję algorytmu służącą do maksymalizacji wartości funkcji. Algorytm optymalizacyjny użyty został do automatycznego dobierania wartości parametru wygładzania tak aby osiągnąć najlepszą jakość klasyfikacji. Stworzono więc funkcję której jedynym parametrem wejściowym jest macierz wygładzania, a parametrem wyjściowym jest wynik 10-cio krotnej walidacji krzyżowej procesu klasyfikacji zbioru danych. Optymalizacji można poddać wszystkie możliwości doboru zmienności parametru wygładzania.

FPA jest algorytmem optymalizacji globalnej przeznaczonym do pracy z funkcjami gdzie przestrzeń rozwiązań jest nieograniczona. W przypadku optymalizacji doboru parametrów wygładzania przestrzeń rozwiązań ograniczona jest od dołu przez zero, i teoretycznie nie jest ograniczona z góry. Bardzo wysokie wartości parametru wygładzania bardzo rzadko mają jednak zastosowanie praktycznie, dlatego też można stosować również ograniczenia odgórne. Należało więc zmodyfikować algorytm FPA w taki sposób aby umożliwić podanie wartości granicznych w obrębie których ma zostać przeprowadzona optymalizacja. Aby to osiągnąć rozszerzono sygnaturę funkcji o dwa dodatkowe parametry, ograniczono zakres losowania populacji początkowej tak aby wszystkie wyniki znalazły się wewnątrz ograniczonego przedziału. Kolejnym krokiem modyfikacji było dopasowanie procesu losowania nowych położeń w taki sposób aby zwracane były tylko nowe lokalizacje w obszarze optymalizowanym. Osiągnięto to poprzez powtarzanie losowania tak długo jak otrzymany wynik nie spełnia założonych kryteriów. Powyższa modyfikacja znacznie poprawiła szybkość zbieżności algorytmu, odpowiednie dobranie granic optymalizacji pozwala znacznie przyśpieszyć proces.

Ostatnią modyfikacją algorytmu w celu lepszej integracji z zadaniem optymalizacji parametrów wygładzania jest dodanie możliwości wprowadzenia znanego dobrego rozwiązania. Jeżeli chcemy optymalizować funkcję, ale mamy rozwiązanie pochodzące z innego algorytmu możemy podać to rozwiązanie do funkcji FPA, co zapewni że jedno z rozwiązań z populacji początkowej znajdzie się w tym punkcie. W przypadku optymalizacji parametrów wygładzania, jako znane dobre rozwiązanie można wykorzystać na przykład wynik metody plug-in. Taka modyfikacja gwarantuje otrzymanie wyniku nie gorszego niż podany, jednakże może negatywnie wpłynąć na zdolności optymalizacyjne algorytmu. Wpływ modyfikacji zostanie zbadany w kolejnym rozdziale.

# 4 Analiza użyteczności

Do oceny użyteczności algorytmu wykorzystana zostanie dziesięciokrotna walidacja krzyżowa na 4 standardowych zbiorach danych:

Tabela 1: Zbiory danych

| Nazwa | Ilość elementów | Wymiarowość |
|---|---|---|
| Seeds | 210 | 8 |
| Cardiotocography (CTG) | 2126 | 35 |
| Breast Cancer Wisconsin | 569 | 32 |
| Iris | 150 | 5 |

Ocenę będzie stanowić stosunek próbek poprawnie zaklasyfikowanych do liczności zbioru danych.

## 4.1 Badanie wpływu doboru parametrów

### 4.1.1 Wybór jądra estymacji

Istnieje wiele jąder które można wykorzystać do estymacji jądrowej i klasyfikacji przy użyciu PNN, nie można jednoznacznie stwierdzić które jest najlepsze. Bardzo często wybór jądra zależy od zbioru danych, często jest też uwarunkowywany łatwością przeprowadzania obliczeń z wykorzystaniem danego jądra. W tabeli 2 przedstawiono wyniki klasyfikacji osiągnięte przez sieć przy użyciu wszystkich jąder z tabeli **??**. Do wykonania klasyfikacji wykorzystano stały parametr wygładzania wynoszący 1 dla wszystkich klas i wymiarów.

Tabela 2: Wybór jądra, h=1

| | Normalne | Trójkątne | Cauchyego | Epanechinkowa | Jednostajne | Dwuwagowe |
|---|---|---|---|---|---|---|
| Seeds | 0.9095238 | 0.8904762 | 0.9 | 0.8857143 | 0.8952381 | 0.8857143 |
| CTG | 0.6622265 | 0.3283063 | 0.9388542 | 0.3344406 | 0.3494707 | 0.3551023 |
| Breast Cancer | 0.4920739 | 0.4974311 | 0.9402569 | 0.5184211 | 0.5007832 | 0.4955514 |
| Iris | 0.9266667 | 0.96 | 0.9533333 | 0.9666667 | 0.9133333 | 0.9666667 |

Wyniki klasyfikacji w zbiorach o niskiej wymiarowości są zbliżone dla wszystkich jąder, natomiast w zbiorach o wyższej wymiarowości zauważalna jest znaczna

różnica pomiędzy jądrami Cauchiego i Normalnym a pozostałymi. Te dwa jądra odróżniają się od pozostałych faktem nieskończonej szerokości. Wszystkie pozostałe jądra dla wartości o module większym od jedności przyjmują wartość zero. Dokładniejsza analiza wyników gęstości prawdopodobieństwa dla poszczególnych klas pokazuje że w przypadku jąder o skończonym czasie trwania większość badanych elementów uzyskuje gęstość prawdopodobieństwa równą zero. Zjawosko to wynika z klątwy wysokiej wymiarowości ( ang. curse of dimentinality). W przestrzeniach o dużej liczbie wymiarów uzyskanie dobrego pokrycia dostępnego miejsca wymaga ogromnej ilości elementów. W przypadku jąder o skończonej szerokości, niskie pokrycie skutkuje powstawaniem przerw w estymatorze. Jeżeli klasyfikowany element trafi na tego rodzaju przerwę, istnieje duże prawdopodobieństwo że nie zostanie poprawnie zaklasyfikowany. Jądra o nieskończonej szerokości, jak na przykład jądro cauchiego radzą sobie z tą sytuacją znacznie lepiej, ponieważ pokrywają one w teorii całą przestrzeń rozwiązań za każdym razem, dlatego też teoretycznie zawsze powinniśmy uzyskać niezerową wartość gęstości prawdopodobieństwa, na podstawie której można dokonać klasyfikacji. W praktyce ograniczona dokładność obliczeń komputerowych powoduje że nawet okna Cauchiego i normalne mają ograniczoną szerokość, jednakże jest ona nadal znacznie większa od szerokości okien skończonych. Próba rozwiązania problemu skończonej szerokości zostanie przeprowadzona w kolejnych podrozdziałach przy pomocy odpowiedniego dostrojenia parametru wygładzania.

W prawie wszystkich przypadkach najlepiej sprawdza się okno Cauchiego. Teoretycznie najlepsze okno Epanechinkowa wypadło najlepiej tylko w jednym przypadku. Powyższy wynik jest pierwszą sugestią że stosowanie metod optymalnych asymptotycznie do estymacji rozkładu prawdopodobieństwa może nie być najlepszym rozwiązaniem do zastosowania w PNN.

### 4.1.2 Stopnie swobody parametru wygładzania

Parametr wygładzania można dobierać z różną dokładnością. Można dobierać tylko jeden globalny parametr, jak również osobny parametr dla każdego wymiaru zbioru danych czy dla każdej klasy osobno.

Stosowanie doboru osobnego parametru wygładzania dla każdego z wymiarów jest uzasadnione zróżnicowaniem zmienności danych. Bardzo często zdarza się że w

jednym zbiorze danych znajdą się kolumny o odchyleniu standardowym różnych rzędów wielkości. W takiej sytuacji, gdyby dobrano wyłącznie jeden parametr wygładzania jeden z wymiarów musiałby być wy-estymowany niepoprawnie ze względu na zbyt duże lub zbyt małe wygładzenie. Przykład takiej sytuacji wykryto badając zbiór seeds wykorzystując okno Epanechnikowa, przy zastosowaniu parametru wygładzania 2 dla wszystkich wymiarów osiągnięto wynik 0.9047619, jednakże część wymiarów przy takim ustawieniu została za bardzo wygładzona i część informacji została utracona. Ustawienie wygładzania jednego z wymiarów na 0.5 przyniosło poprawę wyniku do 0.9142857. Metoda plug-in umożliwia dobór parametrów wygładzania dla każdego z wymiarów.

Osobny parametr wygładzania dla każdej z klas również może być przydatny, można zaistnieć sytuacja w której w jednym zbiorze danych znajdą się klasy bardzo szerokie, charakteryzujące się dużą zmiennością wielu atrybutów jak i klasy bardzo specyficzne, o bardzo niskiej zmienności. W sytuacji kiedy używalibyśmy tylko jednego parametru wygładzania niezależnie od klasy, mogło by się zdarzyć tak że poprawne wygładzenie klasy szerokiej spowodowało by że dwie zbliżone do siebie klasy wąskie zostały by ze sobą połączone. Jednakże dobór osobnych parametrów wygładzania dla każdej z klas niesie ze sobą pewne problemy. Przede wszystkim, wartość parametru wygładzania wpływa na wartość estymowanej funkcji gęstości prawdopodobieństwa. Wysokie wygładzanie prowadzi do niskich wartości gęstości prawdopodobieństwa ponieważ zawsze zachowana musi zostać wartość całki gęstości prawdopodobieństwa. Taki wpływ może powodować błędną klasyfikację, elementy mogą być rzadziej klasyfikowane do klasy bardziej wygładzonej. Kolejnym problemem jest możliwość pojawienia się zerowego odchylenia standardowego. Może zdarzyć się tak że w obrębie klasy dana kolumna będzie miała stałą wartość, a w większości algorytmów wyznaczania parametru wygładzania wygładzanie jest odwrotnie proporcjonalne do odchylenia standardowego. Taka sytuacja prowadzi do wyznaczenia ogromnych bądź nieskończonych wartości parametru wygładzania, co zwykle uniemożliwia klasyfikację.


Rysunek 3: Estymator jądrowy, wygładzanie równe jeden dla każdego wymiaru i klasy


Rysunek 4: Estymator jądrowy, wygładzania 1.5 dla jednej z klas


Rysunek 5: Estymator jądrowy, wygładzanie 2 dla jednego z wymiarów


Rysunek 6: Estymator jądrowy, wygładzanie 2 dla jednego z wymiarów i 1.5 dla jednej z klas

Należy również pamiętać że większa ilość parametrów wygładzania to również większa złożoność obliczeniowa. Algorytmy wyznaczania parametrów wygładzania mają zwykle wysokie wymagania obliczeniowe, metoda plug-in ma złożoność kwadratową. W przypadku optymalizacji przy użyciu FPA większa ilość parametrów do optymalizowania znacznie spowalnia proces optymalizacji.

Tabela 3: Wpływ ilości stopni swobody parametru wygładzania na proces optymalizacji FPA

| Ilość iteracji / Wariant | 10 | 20 | 100 |
|---|---|---|---|
| Jeden parametr | 0.9047 | 0.9047 | 0.9047 |
| Różne dla wymiarów | 0.9523 | 0.9571 | 0.9619 |
| Różne dla klas | 0.895 | 0.9047 | 0.9142 |
| Różne dla wymiarów i klas | 0.9238 | 0.94761 | 0.9571 |

Tabela 3 pokazuje że optymalizacja dla parametru wygładzania zmiennego dla każdego wymiaru daje najlepsze rezultaty najszybciej. Można podejrzewać, że optymalizacja dla każdego wymiaru i klasy dała by lepsze rezultaty, ale wymagało by do bardzo dużej ilości iteracji. Z powyższych przyczyn do dalszych rozważań wybrano wariant wyboru parametru wygładzania dla każdego wymiaru, lecz taki sam dla wszystkich klas.

### 4.1.3 Siła modyfikacji parametru wygładzania

Literatura najczęściej podaje że dobrą wartością siły modyfikacji parametru wygładzania jest 0.5, w niniejszym podrozdziale zbadana zostanie zależność jakości klasyfikacji od wartości parametru siły modyfikacji. Ponadto podjęta zostanie decyzja czy algorytm FPA należy uruchamiać z już ustawionym wygładzaniem, czy wystarczy znaleźć optymalne parametry bez wygładzania.

Pierwszym eksperymentem jaki przeprowadzono jest porównanie wyników uzyskanych przez klasyfikator z parametrem wygładzania wybranym przez algorytm FPA w poprzednim podrozdziale. Zastosowanie parametru modyfikacji spowodowało spadek wyniku z 0.9619 do 0.9285. Można więc wnioskować, że stosowanie parametru wygładzania po zastosowaniu FPA nie jest dobrym rozwiązaniem. Drugim przetestowanym podejściem byłą optymalizacja parametrów wygładzania z modyfikacją ustawioną na 0.5. Wadą takiego rozwiązania jest

znacznie dłuższy czas wykonywania, ze względu na konieczność obliczenia parametrów modyfikacji dla każdego wektora rozwiązań. Rozwiązanie to również nie okazało się skuteczne, wynik jaki udało się osiągnąć w 100 iteracjach to tylko 0.9095.

Tabela 4: Wpływ siły modyfikacji na jakość klasyfikacji

| Wariant \ Siła wygładzania | 0 | 0.5 | 1 |
|---|---|---|---|
| FPA bez modyfikacji, Epanechnikow, seeds | 0.9619 | 0.9047 | 0.9047 |
| plug-in, Cauchy, seeds | 0.9142 | 0.9142 | 0.9142 |
| h=1, Cauchy, seeds | 0.9 | 0.8952 | 0.8952 |
| plug-in, Cauchy, breast cancer | 0.9825 | 0.9825 | 0.9821 |

Wyniki zaprezentowane w tabeli 6 wskazują że stosowanie modyfikacji wygładzania ma neutralny lub negatywny wpływ na proces klasyfikacji. Oczywiście powyższy wniosek nie musi być prawdziwy dla innych zbiorów danych, jednakże w niniejszej pracy modyfikacja parametru wygładzania nie zostanie wykorzystana.

### 4.1.4 Strojenie parametrów FPA

Algorytm Flower Pollination posiada parametry wejściowe, które należy odpowiednio dobrać aby uzyskać najlepszą efektywność. Efektywność algorytmu wyraża się przez szybkość z jaką wynik zbiega do optymalnego, jak również zdolność algorytmu do odnajdywania ekstremów globalnych. W przypadku FPA w klasycznej implementacji zaproponowanej w [?] do inicjalizacji algorytmu należy wybrać jedynie liczność populacji. Wybór ten ma bardzo duży wpływ na zbieżność algorytmu, czym większa populacja tym szybciej algorytm znajduje coraz lepsze rozwiązania, jednakże większa ilość elementów to również więcej obliczeń do wykonania. Należy również pamiętać, że duża liczba elementów populacji daje większą szanse na znalezienie globalnego celu optymalizacji, niska liczba elementów może prowadzić do zatrzymania algorytmu w lokalnym ekstremum. Wysoka liczność populacji jest również pożądana w celu zwiększenie pokrycia badanej przestrzeni. Klątwa wymiarowości oznacza że przestrzenie o wysokich wymiarowościach bardzo trudno jest wypełnić, dlatego też zwłaszcza w przypadkach zbiorów o wysokiej wymiarowości istotny jest duży rozmiar populacji początkowej.

W celu doboru optymalnej wartości parametru liczności populacji przeprowadzono test w którym porównano jakość klasyfikacji i czas wykonywania algorytmu przy użyciu liczności 5 i 50 elementów.Test przeprowadzono wykorzystując optymalizację wyniku pięciokrotnej walidacji krzyżowej. Wykorzystując pięcioelementowa populacja po 100 generacjach osiągnęła wynik 0.9047, obliczenia zajęły 125 sekund. Populacja pięćdziesięciu elementów osiągnęła wynik 0.9619 a czas wykonywania wyniósł 1250 sekund. Różnica w czasie wykonywania jest zrozumiała, złożoność obliczeniowa FPA zależy liniowo od wielkości populacji. FPA wykorzystujące populację 50-ciu elementów osiągnęło wynik lepszy od przypadku pięcioelementowego już w pierwszej generacji, której obliczenie zajęło zaledwie 30 sekund.

Kolejnym elementem mającym wpływ na zbieżność algorytmu FPA jest ograniczenie domeny poszukiwań. Z definicji FPA jest algorytmem optymalizacji globalnej funkcji o nieskończonej dziedzinie, jednakże po zastosowaniu modyfikacji opisanej w podrozdziale 3.4 można wykorzystać go do optymalizacji funkcji ograniczonych. Funkcja jakości klasyfikacji w zależności od wartości parametrów wygładzania jest funkcją ograniczoną, parametr wygładzania nie powinien mieć wartości mniejszej od zera, a wartości wygładzania rzędu dziesiątek czy setek tysięcy w praktyce nigdy nie są wykorzystywane. Dlatego też można ograniczać dziedzinę poszukiwań optymalnego parametru wygładzania. Przeprowadzono test wpływu modyfikacji na zbieżność algorytmu. Usunięcie ograniczeń spowodowało ogromne pogorszenie jakości klasyfikacji i zbieżności algorytmu. W poprzednim teście wykorzystano ograniczenie zakresu parametru wygładzania do przedziału $(0, 10])$, po zmianie przedziału na $(0, 10^6])$ wynik algorytmu w pierwszej iteracji spadł z 0.93 do 0.38, a finalny wynik wyniósł zaledwie 0.9095.

Ostatnią możliwością modyfikacji jest wprowadzona możliwość podania punktu startowego do algorytmu FPA. W badanych zbiorach algorytm FPA niemalże natychmiast znajduje rozwiązanie lepsze od proponowanego przez metodę plug-in, dlatego też rozwiązanie to nie zostało wykorzystane przede wszystkim ze względu na wysoką złożoność obliczeniową metody plug-in. Jednakże, możliwość taka może być przydatna w innych zbiorach danych. Nie zaobserwowano żadnego wpływu stosowania tej modyfikacji w badanych zbiorach danych.

### 4.1.5 Wpływ normalizacji na klasyfikację

W pracy badane są zbiory danych z atrybutami o bardzo różnych charakterystykach zmienności co wymusza dopasowanie parametrów klasyfikacji mocno różniących się dla każdego wymiaru. Można analizować przykładowo dane ma temat wzrostu i na temat zarobków, obydwie zmienne będą miały zarówno różną wartość oczekiwaną jak i odchylenie standardowe. Przeskalowanie zbioru danych w taki sposób aby uniknąć tego typu różnic mógłby teoretycznie uprościć analizę danych i procedury wyznaczania parametrów do algorytmów, jak na przykład parametr wygładzania. Przeprowadzono test sprawdzający wpływ zastosowania normalizacji przed przystąpieniem do analizy danych na wyniki klasyfikacji.

Bardzo ważny problemem podczas wykorzystywania algorytmów optymalizacyjnych jest nadmierne dopasowanie danych. Zjawisko to wynika ze zbyt długiego treningu, z czasem optymalizowana struktura traci swoje możliwości generalizacji. Zbyt długi trening prowadzi do wyciągania wniosków z danych w sposób nie dający większego zrozumienia problemu, ale pozwalający na osiągnięcie lepszego wyniku. Aby rozwiązać ten problem należy tak dobrać ilość iteracji algorytmu aby uzyskać najbardziej zoptymalizowany wynik nie pozwalając na przetrenowanie. Aby to osiągnąć przeprowadzono testy wykorzystujące pięciokrotną walidację krzyżową. Należy przypomnieć, że algorytm FPA optymalizuje wynik pięciokrotnej walidacji krzyżowej, wykorzystane zostaną dwie walidacje krzyżowe, jedna w procesie tworzenia modelu z wykorzystaniem FPA, a druga do walidacji algorytmu tworzenia modeli. Przyczyną przetrenowania może być zarówno duża ilość iteracji jak i duża populacja. W celu doboru optymalnej ilości iteracji i wielkości populacji przeprowadzono porównanie, wyniki przedstawiono w tabeli 5.

Tabela 5: Wpływ ilości iteracji i wielkości populacji na jakość klasyfikacji dla zbioru seeds.

| Ilość iteracji / Liczność populacji | 10 | 20 | 50 |
|---|---|---|---|
| 5 | 0.9285 | 0.9378 | 0.9047 |
| 10 | 0.9476 | 0.939 | 0.941 |
| 20 | 0.9333 | 0.8952 | 0.8952 |

Najlepszym wynikiem jaki udało się uzyskać jest wynik 0.9476 dla przypadku dziesięciu iteracji i dziesięciu elementów populacji FPA. Takie ustawienia zostaną wykorzystane w niniejszej pracy, jednakże dla każdego zbioru danych wybór tych parametrów należy przeprowadzić osobno.

Tabela 6: Wpływ normalizacji na jakość klasyfikacji

| Wariant \ Siła wygładzania | 0 | 0.5 | 1 |
|---|---|---|---|
| FPA bez wygładzania, Epanechnikow, seeds | 0.9619 | 0.9047 | 0.9047 |
| plug-in, Cauchy, seeds | 0.9142 | 0.9142 | 0.9142 |
| h=1, Cauchy, seeds | 0.9 | 0.8952 | 0.8952 |
| plug-in, Cauchy, breast cancer | 0.9825 | 0.9825 | 0.9821 |

Można zaobserwować że że wszystkich badanych przypadkach neutralizacja ma wpływ pozytywny lub neutralny na wyniki klasyfikacj, dlatego też zostanie użyta w finalnym algorytmie. Dodatkowo, zastosowanie normalizacji umożliwia określenie mniejszego rejonu optymalizacji dla algorytmu FPA, który będzie uniwersalny niezależnie od zbioru danych. Ogranicza to liczbę parametrów wejściowych algorytmu, i poprawia czas zbieżności.

### 4.1.6 Wybór parametru wygładzania

W niniejszej pracy zbadane zostaną dwie metody automatycznego doboru parametru wygładzania, metoda plug-in, oraz optymalizacji przy użyciu FPA. Dobierane będą osobne parametry wygładzania dla każdego z wymiarów. Do analizy wykorzystane zostanie jedno okno o nieskończonej szerokości ( Cauchiego) i jedno jądro o skończonej szerokości (Epanechinkowa). Dokonano porównania jakości klasyfikacji dla przypadku wykorzystania stałego wygładzania, dla metody plug-in i dla algorytmu FPA uruchomionego na 100 generacji.

Tabela 7: Dobór parametrów wygładzania dla jądra Cauchiego

| | 1 | plug-in | FPA |
|---|---|---|---|
| Seeds | 0.9 | 0.9142 | 0.9714 |
| Cardiotocography (CTG) | 0.9388 | 0.9825 | 0.98824 |
| Breast Cancer Wisconsin | 0.9402 | 0.94558 | 0.9595 |
| Iris | 0.9533 | 0.9333 | 0.97333 |

W tabeli 7 można zauważyć że we wszystkich przypadkach algorytm FPA był w stanie osiągnąć lepsze wyniki niż metoda plug-in. Często już losowo wygenerowana populacja osiąga wynik znacznie lepszy od tego osiągniętego przez metodę plug-in. Dodatkowo w tabeli ??obserwujemy że w przypadku użycia jąder o skończonej szerokości algorytm FPA świetnie radzi sobie z doborem wygładzania pozwalającego na dostateczne wypełnienie przestrzeni, i w przeciwieństwie do algorytmu plug-in umożliwia użycie jąder tego typu. Wynik taki jest kolejnym argumentem potwierdzającym tezę postulującą że stosowanie estymatora gęstości o najmniejszym błędzie średnio kwadratowym i w ogólności dążenie do jak najlepszego odwzorowania funkcji gęstości prawdopodobieństwa nie jest najbardziej optymalne dla zadania klasyfikacji przy użyciu PNN. Przykład jąder o skończonej szerokości pokazuje że dokładne pokrycie pewnych obszarów przestrzeni przez estymator jest ważniejsze niż błąd średnio kwadratowy. Kolejnym argumentem za skutecznością FPA jest możliwość optymalizacji oceniającej istotność konkretnych wymiarów w procesie klasyfikacji. Proces optymalizacji prowadzi do znalezienia takiego rozwiązania gdzie kolumny wprowadzające nieistotne bądź szkodliwe informacje mogą zostać wygładzone znacznie bardziej od pozostałych. Można zauważyć że różnica jakości klasyfikacji jest większa w małych i nisko-wymiarowych zbiorach danych. Fakt ten wynika z mniejszej przestrzeni rozwiązań którą łatwiej jest przeszukać. W tych przypadkach zastosowanie większej populacji i większej liczby iteracji mogło by dać jeszcze lepsze rezultaty.

Jeżeli klasyfikacja pojedynczego elementu zostanie uznana za operację jednostkową to złożoność obliczeniowa algorytmu plug-in jest kwadratowa względem ilości elementów w zbiorze danych, natomiast złożoność obliczeniowa algorytmu FPA jest liniowa. Jednakże, w przypadku algorytmu FPA liniowy czas wykonywania przemnożony zostaje przez stałą wynoszącą iloraz ilości iteracji i wielkości populacji początkowej. Dlatego też w mały zbiorach danych algorytm FPA wykonuje się znacznie wolniej niż algorytm plug-in, jednakże w zbiorach znacznie większych, jeżeli ilość iteracji i wielkość populacji pozostanie na tym samym poziomie algorytm FPA będzie wykonywał się szybciej. Zestawienie czasów wykonania przedstawiono w tabeli 10.

Tabela 8: Czas wykonania algorytmów

| | 1 | plug-in | FPA |
|---|---|---|---|
| Seeds | 0.24 s | 1.12 s | 1250 s |
| Cardiotocography (CTG) | 19 s | 336 s | 89988 s |
| Breast Cancer Wisconsin | 1.95 s | 30.30 s | 8793 s |
| Iris | 0.18 s | 0.47 s | 906.91 s |

### 4.1.7 Finalny algorytm

Na podstawie wszystkich poprzednich obserwacji przygotowano algorytm służący do klasyfikacji który zostanie porównany z innymi dostępnymi metodami klasyfikacji w kolejnym podrozdziale. Założenia algorytmu:

- Wykorzystanie Probabilistycznej sieci neuronowej jako klasyfikatora
- Dobór parametru wygładzania przy pomocy Flower Pollination Algorithm
- Brak modyfikacji parametru wygładzania
- Okno Cauchi'ego
- Strojenie parametrów dla każdego wymiaru osobno, tak samo dla każdej z klas.
- Dokonano normalizacji zbioru danych przed podejściem do klasyfikacji

## 4.2 Porównanie z istniejącymi algorytmami

Do porównania z innymi algorytmami wykorzystana zostanie dziesięciokrotna walidacja krzyżowa. Porównane zostaną wyniki oraz czas wykonywania algorytmu dla wszystkich zbiorów danych wymienionych w tabeli 1. Do porównania wykorzystaną popularne metody klasyfikacji:

- SVM - maszyna wektorów nośnych, jej działanie zostało opisane w podrozdziale 1.2.3. Do porównania wyników wykorzystany zostanie pakiet e1071 i wchodząca w jego skład funkcja svm. Parametry modelu cost i gamma zostały dobrane przy użyciu funkcji svm::tune, następnie stworzono model i przeprowadzono klasyfikację.

- Gradient Boosting (XGBoost) - algorytm polegający na tworzeniu modelu poprzez składanie mniej dokładnych modeli, najczęściej drzew decyzyjnych. Biblioteka XGBoost implementuje algorytm parallel tree boosting, umożliwiając rozwiązywanie problemów klasyfikacyjnych i regresyjnych. Biblioteka ta jest obecnie bardzo popularna w świecie analizy danych. Parametry modelu zostały dobrane eksperymentalnie.
- KNN - algorytm k najbliższych sąsiadów opisany w podrozdziale 1.2.1. Współczynnik k

Tabela 9: Czas wykonania algorytmów

| | SVM | XGBoost | kNN | FPA PNN |
|---|---|---|---|---|
| Seeds | 0.9380 | 0.9333 | 0.9142 | |
| Cardiotocography (CTG) | 0.9901 | 0.9901 | 0.9891 | |
| Breast Cancer Wisconsin | 0.9771 | 0.9666 | 0.97189 | |
| Iris | 0.9466 | 0.94 | 0.96 | 0.9333 |

Tabela 10: Jakość klasyfikacji algorytmów

| | SVM | XGBoost | kNN | FPA PNN |
|---|---|---|---|---|
| Seeds | 24.28 | 2.92 | 14.82 | |
| Cardiotocography (CTG) | 3764 | 8.30 | 121 | |
| Breast Cancer Wisconsin | 188.4 | 3.16 | 20.43 | |
| Iris | 12.73 | 2.22 | 12.75 | |

# 5 Podsumowanie

asd

# 6 Kierunki dalszego rozwoju

Niniejsza praca stanowi jedynie badanie użyteczności algorytmu FPA do strojenia probabilistycznych sieci neuronowych, jednakże analizowana dziedzina jest niezwykle szeroka i istnieje jeszcze wiele elementów które należało by zbadać. Dużym problemem prezentowanego algorytmu jest jego czas wykonywania. Przede wszystkim, niniejszy algorytm zaimplementowano w języku R, który świetnie nadaje się do szybkiego prototypowania, jednakże jest znacznie wolniejszy od języków niższego poziomu. Implementacja zarówno FPA jak i PNN w kompilowanym języku jak na przykład C++ przyniosłaby znaczny zysk wydajnościowy. Wartym rozważenia było by również przeniesienie obliczeń na GPU. Należało by również przetestować proponowane modyfikacje algorytmu PNN zakładające zastąpienie umieszczania wszystkich elementów zbioru danych w strukturze sieci umieszczaniem jedynie reprezentantów klastrów. Zwiększenie szybkości wykonywania algorytmu pozwoliło by na wykorzystanie optymalizacji wygładzania dla każdej z klas z osobna, jak również na zwiększenie populacji i liczby iteracji. Dopracowanie samego algorytmu mogło by polegać na zbadaniu innych metod przeszukiwania przestrzeni w celach optymalizacyjnych. Można również przetestować możliwości strojenia parametru modyfikacji wygładzania przy użyciu FPA przy jednoczesnym strojeniu pozostałych parametrów. Zainteresowanie sieciami typu PNN jest obecnie bardzo duże, dlatego też powstaje wiele algorytmów mających na celu poprawę architektury sieci tego typu. Istnieją modyfikację wprowadzające dodatkowe wagi i/lub warstwy, warto było by przeprowadzić próbę wykorzystania algorytmu FPA do optymalizacji wszystkich parametrów takich zmodyfikowanych sieci.

# Bibliografia

[1] https://cve.mitre.org/about/faqs.html

[2] https://cve.mitre.org/about/terminology.html

[3] "CVE Overview for Prospective CNAs", Mitre Corporation, Version 1.0, 29 Wrzesień 2018

# Spis rysunków

1 Logo języka R . . . . . 21
2 Przykład wykresu przekroju estymatora jądrowego . . . . . 24
3 Estymator jądrowy, wygładzanie równe jeden dla każdego wymiaru i klasy . . . . . 32
4 Estymator jądrowy, wygładzania 1.5 dla jednej z klas . . . . . 33
5 Estymator jądrowy, wygładzanie 2 dla jednego z wymiarów . . . . . 34
6 Estymator jądrowy, wygładzanie 2 dla jednego z wymiarów i 1.5 dla jednej z klas . . . . . 35

# Spis tabel


1 Zbiory danych . . . . . 29
2 Wybór jądra, h=1 . . . . . 29
3 Wpływ ilości stopni swobody parametru wygładzania na proces optymalizacji FPA . . . . . 36
4 Wpływ siły modyfikacji na jakość klasyfikacji . . . . . 37
5 Wpływ ilości iteracji i wielkości populacji na jakość klasyfikacji dla zbioru seeds. . . . . 39
6 Wpływ normalizacji na jakość klasyfikacji . . . . . 40
7 Dobór parametrów wygładzania dla jądra Cauchiego . . . . . 40
8 Czas wykonania algorytmów . . . . . 42
9 Czas wykonania algorytmów . . . . . 43
10 Jakość klasyfikacji algorytmów . . . . . 43

# Spis listingów

1 Kod funkcji classify i get_class_score . . . . . 23
2 Algorytm Mantegana do generacli lotów Levy'ego . . . . . 25

# Dodatki