**Klasyfikator**

Dokumentacja projektu

**Grupa projektowa:**  
Kamil Żak

Paweł Mitruś

Łukasz Napora

Mateusz Bukowski

Spis treści

[1. Cel dokumentu 3](#_Toc408586233)

[2. Środowisko programowania 3](#_Toc408586234)

[3. Opis problemu 3](#_Toc408586235)

[3.1. Reprezentacja i sposób klasyfikowania symboli 3](#_Toc408586236)

[3.2. Konstrukcja i symulacja automatu 4](#_Toc408586237)

[3.3. Optymalizacja 5](#_Toc408586238)

[4. Etap I 5](#_Toc408586239)

[4.1. Generowanie zbioru uczącego 5](#_Toc408586240)

[4.2. Tworzenie funkcji przejścia 5](#_Toc408586241)

[4.3. Symulacja działania automatu 5](#_Toc408586242)

[4.4. Funkcja błędu 5](#_Toc408586243)

[5. Etap II 5](#_Toc408586244)

[5.1. Generowanie zbioru uczącego 5](#_Toc408586245)

[5.2. Tworzenie funkcji przejścia 5](#_Toc408586246)

[5.3. Symulacja działania automatu 5](#_Toc408586247)

[5.4. Funkcja błędu 5](#_Toc408586248)

[6. Etap III 6](#_Toc408586249)

[6.1. Generowanie zbioru uczącego 6](#_Toc408586250)

[6.2. Tworzenie funkcji przejścia 6](#_Toc408586251)

[6.3. Symulacja działania automatu 6](#_Toc408586252)

[6.4. Funkcja błędu 6](#_Toc408586253)

[7. Format uruchomieniowy programu 6](#_Toc408586254)

# Cel dokumentu

Celem dokumentu jest przedstawienie użytkownikowi sposobu działania Klasyfikatora. Dokument skupia się na wyjaśnieniu kwestii teoretycznej, metody rozwiązywania poszczególnych problemów oraz implementacji.

# Środowisko programowania

Klasyfikator został zaimplementowany w środowisku MatLab – głównym powodem wyboru tego środowiska, były liczne operacje na macierzach. W ten sposób, z wiedzą że MatLab dysponuje optymalnymi algorytmami do wyżej wspomnianych działań, staraliśmy się podnieść wydajność programu.

# Opis problemu

Problemem jest implementacja automatu klasyfikującego symbole z zadanych zbiorów do odpowiadających im klas na podstawie przedstawionych cech symboli.

Poszczególne etapy: 1, 2 oraz 3 różnią się wzajemnie typem implementowanego automatu. Odpowiednio dla:

1. Automat deterministyczny
2. Automat niedeterministyczny
3. Automat rozmyty

## Reprezentacja i sposób klasyfikowania symboli

Do automatu wraz z każdym zbiorem wprowadzane są następujące wartości:

* Klasy: Zadany jest skończony zbiór klas , do których finalnie przypisywane są testowane symbole.
* Symbole: To reprezentanci każdej z klas. Zadany jest skończony zbiór symboli   
  . Po próbach rozpoznania symbolu są one przypisywane do odpowiednich klas.
* Cechy: To pewne określone właściwości każdego z symboli. Na ich podstawie decydujemy, czy dany symbol klasyfikuje się do danej klasy. Zadany jest skończony zbiór cech

Cechy

Symbole

Klasa

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Symbol | C1 | C2 | … | Cn |
| A1.1 | 2 | 14 | 5,1 | 22 |
| A1.2 | 0 | 12 | 4,8 | 30 |
| … | 1,9 | 9 | 7 | 25 |
| A1.n | 0,5 | 11 | 6 | 26 |
| … |  |  |  |  |
| An.1 |  |  |  |  |
| ... |  |  |  |  |
| An.n |  |  |  |  |

## Konstrukcja i symulacja automatu

Każdy z etapów możemy opisać poniższym, ogólnym algorytmem. Kwestie, pomiędzy którymi poszczególne etapy się różnią, a w zasadzie są kluczowymi, zostaną opisane w punktach 4, 5, 6.

* + 1. Generowanie (wczytywanie) zbioru uczącego – każdy ze zbiorów zawiera wartości dla symboli oraz cech. Są one dodawane losowo.
    2. Generowanie (wczytywanie) zbioru treningowego – każdy ze zbiorów jest losowany w ten sam sposób (niezależnie od etapu).
* Losujemy wartości z parametryzowanego przedziału.
* Losujemy wartości z rozkładu Gaussa, gdzie wartość oczekiwana oraz odchylenie są parametryzowane, którymi następnie zaburzamy wcześniej wylosowany zbiór.
* Kolejno, normalizujemy wartości zbioru do przedziału [0, 1].
* W zależności od liczby podanych przedziałów podziału każdej cechy, zamieniamy wartości na liczby porządkowe liczby całkowite (w zależności od tego w którym przedziale dla [0,1] znajdowała się poprzednia wartość).
  + 1. Generowanie automatu – opisane poniżej.
    2. Symulacja przejścia automatu – opisana poniżej.
    3. Optymalizacja – wykonywana jest za pomocą algorytmu PSO (3.3).
    4. Liczenie błędu – główna zasada polega na ilorazie symboli rozpoznanych do wszystkich symboli znajdujących się w zbiorze testowym.

## Optymalizacja

Optymalizacji dokonujemy algorytmem PSO (Particle Swarm Optimization, Optymalizacja Rojem Cząstek), który uprzednio zaimplementowaliśmy.

OPIS PSO

# Etap I

## Generowanie zbioru uczącego

* + 1. Zbiór cech losowany rozkładem jednostajnym
    2. Zbiór wylosowanych wartości zaburzany rozkładem Gaussa
    3. Zbiór cech jest normalizowany do przedziału [0,1]
    4. W zależności od ilości podanych przedziałów, dzielimy na tyle zadany przedział [0,1]
    5. Przypisujemy do każdej z cech wartość (porządkowa liczba całkowita) w zależności od tego w którym przedziale się znajduje.

## Tworzenie funkcji przejścia

Automat jest deterministyczny, więc losowany jest tylko jeden stan akceptujący dla każdego symbolu.

## Symulacja działania automatu

## Funkcja błędu

* + 1. Otrzymujemy wektor stanów po symulacji działania automatu
    2. Jeżeli symbol nie został rozpoznany, nie zwiększamy licznika rozpoznanych symboli, a w przypadku gdy został rozpoznany zwiększamy go o 1
    3. Obliczenia powtarzane są dla każdego symbolu
    4. Liczony jest procentowy błąd w klasyfikowaniu symboli

# Etap II

## Generowanie zbioru uczącego

* + 1. Zbiór cech losowany rozkładem jednostajnym
    2. Zbiór wylosowanych wartości zaburzany rozkładem Gaussa
    3. Zbiór cech jest normalizowany do przedziału [0,1]
    4. W zależności od ilości podanych przedziałów, dzielimy na tyle zadany przedział [0,1]
    5. Przypisujemy do każdej z cech wartość (porządkowa liczba całkowita) w zależności od tego w którym przedziale się znajduje.

## Tworzenie funkcji przejścia

Z racji że mamy do czynienia z automatem niedeterministycznym, losujemy kilka (od 0 do 3) stanów akceptujących dla każdego symbolu.

## Symulacja działania automatu

## Funkcja błędu

# Etap III

## Generowanie zbioru uczącego

* + 1. Zbiór cech losowany rozkładem Gaussa
    2. Zbiór cech jest normalizowany do przedziału [0,1]
    3. Każda z cech będzie rozszerzana do wektora o długości równej ilości przedziałów
    4. W wektorze umieszczamy wartości wg wzoru , gdzie to wartość cechy ze zbioru postaci w punkcie 2 oraz

Zasadnicza różnica polega na tym, że w wypadku słów rozmytych cecha nie jest reprezentowana przez jedną liczbę, a przez wektor. Zmienia to strukturę zbioru cech, z 2 wymiarowego na 3 wymiarowy.

## Tworzenie funkcji przejścia

Z racji że mamy do czynienia z automatem rozmytym losowane są wartości z przedziału [0 liczba.podzialow].

## Symulacja działania automatu

* + 1. Do automatu zostaje przekazana macierz cech danego symbolu.
    2. Zaczynamy od stanu początkowego stan.poczatkowy = [1 0 0 ... 0 ]
    3. Dla każdej z cech iterujemy po wektorze ją reprezentującym
    4. Przy użyciu funkcji:

Iterujemy przez wszystkie wartości w stan.poczatkowy i następnie z użyciem funkcji

tworzymy stan wejściowy dla kolejnej funkcji przejścia automatu.

* + 1. Zwracamy stworzony wektor stanów

W poprzednich etapach do symulacji automatu przekazywane były reprezentacje symbolu (wektor cech dla symbolu). W tym przypadku przekazujemy macierz cech (wektor różnych cech oraz wektory reprezentujące każdą z pojedynczych cech).

## Funkcja błędu

* + 1. Otrzymujemy wektor stanów po symulacji działania automatu
    2. Jeżeli symbol nie został rozpoznany, nie zwiększamy licznika rozpoznanych symboli, a w przypadku gdy został rozpoznany zwiększamy go o
    3. Obliczenia powtarzane są dla każdego symbolu
    4. Liczony jest procentowy błąd w klasyfikowaniu symboli

# Format uruchomieniowy programu