|  |
| --- |
|  |
| SIP - sprawozdanie |
| Rozpoznawanie obrazów znaków drogowych przy użyciu sieci Kohonena. |

|  |
| --- |
| Wojciech Sobczak 209976, Bartosz Gasztych 210003  02.01.2018 r. |

**Spis treści**

[1. Problem 2](#_Toc502697381)

[2. Sieć Kohonena 2](#_Toc502697382)

[2.1. Opis 2](#_Toc502697383)

[2.2. Strategie wybierania zwycięzcy 2](#_Toc502697384)

[2.3. Funkcja uczenia 3](#_Toc502697385)

[3. Techniczna realizacja 3](#_Toc502697386)

[3.1. Technologia 3](#_Toc502697387)

[3.2. Interfejs 3](#_Toc502697388)

[3.3. Implementacja 4](#_Toc502697389)

[3.3.1. Topologie 4](#_Toc502697390)

[3.3.2. Funkcje uczące 4](#_Toc502697391)

[3.3.3. Funkcje redukcji 4](#_Toc502697392)

[3.3.4. Preprocessing 4](#_Toc502697393)

[4. Testy 6](#_Toc502697394)

[4.1. Przegląd skuteczności parametrów 6](#_Toc502697395)

[4.2. Test rozmazywania 7](#_Toc502697396)

[4.3. Test przygotowanych zniekształceń 7](#_Toc502697397)

[5. Wnioski 8](#_Toc502697398)

**Spis rysunków**

[Rysunek 1 Interfejs użytkownika 4](#_Toc502697411)

[Rysunek 2 Obraz poddany preprocessingowi 5](#_Toc502697412)

**Spis tabel**

[Tabela 1 Wyniki przeglądu najbardziej pasujących parametrów 6](#_Toc502697485)

[Tabela 2 Test efektywności po rozmyciu 7](#_Toc502697486)

[Tabela 3 Test przygotowanych zniekształceń 8](#_Toc502697487)

**Spis równań**

[Równanie 1 Reguła Grossberga 3](#_Toc502697506)

# Problem

Zadaniem tego projektu było stworzenie programu, który używając sieci neuronowej Kohonena, wykonywał zadanie rozpoznawanie pewnego zbioru obrazów znaków drogowych. Nasz projekt obejmował zbiór:

* 19 znaków nakazu,
* 42 znaki ostrzegawcze,
* 37 znaków zakazu,

pobranych ze strony szkoleniowej dla kierowców, [www.dlakierowcy.info](http://www.dlakierowcy.info).

# Sieć Kohonena

## Opis

Sieć Kohonena to typ sieci neuronowej, która uczona jest w trybie uczenia nienadzorowanego. Oznacza to, że proces uczenia pozbawiony jest korekt/kar za niepoprawną klasyfikację, a sama sieć kształtuje się tylko poznawaniem zbioru uczącego. Sieć ta posiada tylko jedną warstwę neuronów, których zadaniem, w procesie uczenia, jest modyfikowanie wag poszczególnych wejść i generowanie wedle tychże wejść, odpowiednio dużej wartości wyjściowej. Wyżej wspomniane wagi, generowane są losowo, gdy sieć zaczyna się uczyć, a ich ilość zależy od specyfiki obiektów, którymi tę sieć uczymy (w przypadku obrazów, mogą to być piksele danego obrazu).

## Strategie wybierania zwycięzcy

Podczas uczenia, suma ważona wejść tworzy pewną wartość. Neuron, który uzyska największą zostaje sklasyfikowany jako zwycięzca. Przy klasyfikacji, przypisany będzie do niego jakiś obiekt, lecz przy uczeniu, musimy jakoś taki neuron nagrodzić, podkreślić jego wyjątkowość dla danego obiektu uczącego. Tutaj wyróżniamy 2 strategie:

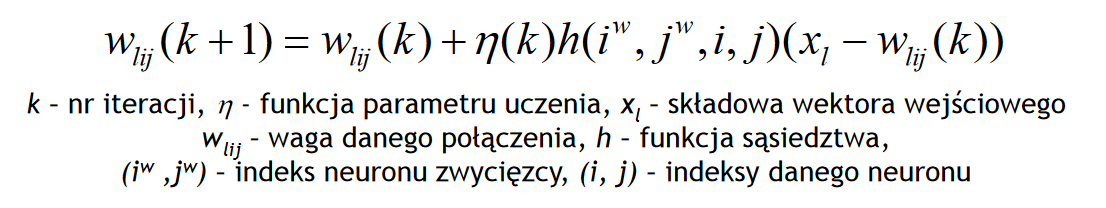
* WTA – winner takes all
* WTM – winner takes most

WTA, polega na tym, że gdy znajdziemy neuron, który wykazał się największą sumą ważoną, ustalamy jego wartość dla obiektu na „1” czyli, największą, a całej reszcie przypisujemy „0”.

WTM zaś, jest strategią mniej faworyzującą, która neuronowi zwycięzcy przypisuje wartość „1”, lecz reszcie neuronów przypisuje wartość zgodnie z wybraną przez programistę funkcją, np. kapelusza meksykańskiego.

## Funkcja uczenia

Wraz z czasem uczenia się sieci, następuje spadek jej efektywności, który to moduluje się tak zwaną „stałą uczenia”, która to jest dowolną funkcją malejącą, według tak zwanej „Reguły Grossberga”, którą opisuje następujący wzór:



Równanie 1 Reguła Grossberga

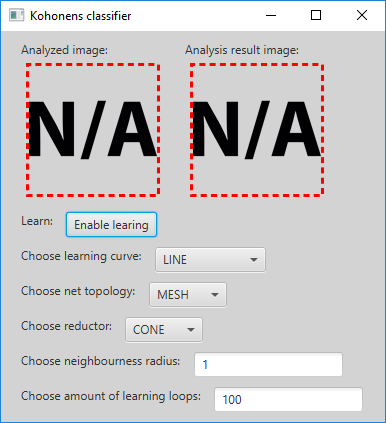
# Techniczna realizacja

## Technologia

Program do realizacji założeń projektu został napisany w języku JAVA, a został on wybrany, ze względu na to, że grupa projektowa ma w nim największe doświadczenie, więc problemem było tylko napisanie programu, a nie walka z zawiłościami języka programowania.

## Interfejs

Interfejs graficzny programu wygląda następująco:



Rysunek 1 Interfejs użytkownika

I pozwala na nauczenie sieci, według parametrów dostępnych użytkownikowi, a mianowicie:

* Funkcja ucząca
* Topologia sieci
* Funkcja redukcji – niestety dostępna tylko jedna, lecz chęci były wielkie
* Promień sąsiedztwa, do powyższej funkcji
* Ilość serii uczących

Po ustaleniu powyższych, można nacisnąć przycisk „Enable learning”, który rozpocznie procedurę uczenia, której proces będzie zasygnalizowany paskiem postępu. Po zakończeniu procesu uczenia się sieci, do okna programu można przeciągnąć dowolny obrazek, który pojawi się w polu „Analyzed image”, a po chwili w polu po prawej pojawi się rezultat, który sieć uznała za najbardziej odpowiedni.

## Implementacja

### Topologie

Zaimplementowane zostały dwie topologie sieci:

* Topologia siatki, z sąsiedztwem złożonym, która w praktyce była dwuwymiarową tablicą neuronów
* Topologia linii, która z kolei była tablicą jednowymiarową

Dla sieci został zaprojektowana klasa abstrakcyjna „Net”, której realizacjami powyższych technologii były klasy „MeshNet” oraz „LineNet”.

### Funkcje uczące

Funkcji uczących z kolei zaimplementowanych zostało trzy:

* Funkcja liniowa
* Funkcja kwadratowa
* Funkcja hiperboliczna

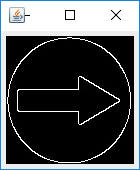
Gdzie to iteracja pętli uczenia, a to ilość wszystkich planowanych iteracji.

### Funkcje redukcji

Jako że wymienioną w wykładach dotyczących tego przedmiotu funkcją redukcji, była funkcja kapelusza meksykańskiego, która jest dosyć złożona, postanowiliśmy, ze względów optymalizacyjnych, uprościć ją funkcją stożka.

### Preprocessing

Obrazy uczące, oraz te które poddawane są klasyfikacji są poddawane preprocessingowi. Proces ten polega na wykryciu krawędzi w danym obrazie, oraz przedstawienie ich w postaci białych linii na czarnym tle (binaryzacja). Po przygotowaniu obraz prezentuje się następująco:



Rysunek 2 Obraz poddany preprocessingowi

# Testy

## Przegląd skuteczności parametrów

Pierwszym oraz najważniejszym testem, było wyznaczenie najbardziej optymalnych parametrów uczenia sieci, które dadzą w efekcie najlepiej przystosowaną sieć, która zostanie użyta do dalszych testów. Dla parametrów:

* Funkcje uczące:
  + Liniowa, Kwadratowa, Hiperboliczna
* Promieni sąsiedztwa:
  + 1, 2, 3, 4, 5, 6
* Topologii:
  + Siatki, Linii
* Ilości iteracji:
  + 10, 50, 100, 200, 500, 1000, 2000, 5000

Przeprowadzono test wszystkich możliwych kombinacji, których dokładne wyniki zostaną załączone do dokumentu w postaci arkusza Excel. Do prezentacji w raporcie wybraliśmy najbardziej obrazowe wyniki, czyli 10 najlepszych, oraz 10 najgorszych.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Efektywność** | **Topologia** | **Funkcja ucząca** | **Promień sąsiedztwa** | **Loops** |
| 0,78571427 | Siatka | Kwadratowa | 3 | 200 |
| 0,7653061 | Siatka | Kwadratowa | 1 | 500 |
| 0,75510204 | Siatka | Liniowa | 2 | 100 |
| 0,75510204 | Siatka | Kwadratowa | 2 | 500 |
| 0,75510204 | Siatka | Liniowa | 2 | 1000 |
| 0,75510204 | Siatka | Hiperboliczna | 2 | 2000 |
| 0,75510204 | Siatka | Hiperboliczna | 1 | 5000 |
| 0,74489796 | Siatka | Liniowa | 1 | 100 |
| 0,74489796 | Siatka | Liniowa | 1 | 500 |
| 0,74489796 | Siatka | Liniowa | 5 | 500 |
| … | … | … | … | … |
| 0,14285715 | Linia | Hiperboliczna | 5 | 10 |
| 0,14285715 | Linia | Hiperboliczna | 6 | 5000 |
| 0,13265306 | Linia | Hiperboliczna | 3 | 200 |
| 0,12244898 | Linia | Hiperboliczna | 6 | 2000 |
| 0,12244898 | Linia | Kwadratowa | 6 | 5000 |
| 0,1122449 | Linia | Kwadratowa | 6 | 2000 |
| 0,1122449 | Linia | Liniowa | 5 | 5000 |
| 0,1122449 | Linia | Liniowa | 6 | 5000 |
| 0,10204082 | Linia | Hiperboliczna | 5 | 5000 |
| 0,091836736 | Linia | Kwadratowa | 5 | 5000 |

Tabela 1 Wyniki przeglądu najbardziej pasujących parametrów

Przytłaczająco większą skuteczność miały kombinacja z topologią siatki. Na przełomie tabeli, gdzie znajdują się posortowane po efektywności kombinacje, znajduje się zmiana topologii z siatki na linię, a różnica w efektywności to 21%.  
Podczas tego testu została także obalona intuicja, podpowiadająca że ilość pętli uczących znacząco wpłynie na efektywność, bo jak widać na tabeli wyników, pierwsze miejsce piastuje kombinacja z 200 iteracjami, a dopiero na 7, z 5000.

## Test rozmazywania

Drugim testem, którym poddana została nasza sieć, był test rozmazania. Polegał on na nałożenie macierzy rozmycia Gaussa, o rozmiarach od 1 do 10, i sprawdzenia efektywności rozpoznawania znaków. Wyniki prezentują się następująco.

|  |  |
| --- | --- |
| **Rozmiar** | **Efektywność** |
| 1 | 0.6020408 |
| 2 | 0.030612245 |
| 3 | 0.020408163 |
| 4 | 0.010204081 |
| 5 | 0.0 |
| 6 | 0.0 |
| 7 | 0.020408163 |
| 8 | 0.010204081 |
| 9 | 0.010204081 |
| 10 | 0.010204081 |

Tabela 2 Test efektywności po rozmyciu

Wyniki są zaskakująco niskie, lecz wynikają najpewniej z algorytmu który przygotowuje obrazy przez uczeniem i klasyfikacją, a jest to już wcześniej wspomniane wykrywanie krawędzi, które to z rozmyciem zacierają się w znacznym stopniu.

## Test przygotowanych zniekształceń

Dla znaków nakazu (ponieważ było ich najmniej), zostało przygotowanych po 4 ich kopie z różnymi artefaktami, a były to:

* Umieszone w losowym miejscu koło, które udawać miało kulę śniegu
* Nałożony na obraz „spray” z programu „Windows Paint”, który miał symulować stan znaku po śnieżycy
* Również „spray’em”, jednak teraz nałożony losowy kształt, lub napis, który miał symulować wandali
* Oraz znów „spray’em”, tylko białym i dosyć niedokładnie, dodany jakiś element który mógł sprawić błąd interpretacji znaku, np. Dodatkowy grot przy nakazie skrętu w prawo, który również miał symulować wandali

Wyniki prezentują się następująco:

|  |  |  |  |
| --- | --- | --- | --- |
| **Kula śniegu** | **Losowy tekst** | **Śnieżyca** | **Element rozpraszający** |
| 0.8947368 | 0.15789473 | 0.0 | 0.7894737 |

Tabela 3 Test przygotowanych zniekształceń

Jak widać najłatwiej klasyfikatorowi udało się sobie poradzić z kulami śniegu i elementami rozpraszającymi, które miały zmienić znaczenie znaku. Natomiast uzyskane przez nas rozwiązanie jest bardzo mało odporne na osiedlowych wandali oraz na duże opady śniegu.

# Wnioski

W testach najbardziej pasujących do realnych warunków, czyli testy w których na obrazie następują zaburzenia, widać wyraźnie że nasza sieć może być, a jeżeli chcielibyśmy jej w jakiś sposób użyć, musi być jeszcze lepsza. Testy w których obraz jest zaburzony, ludzkim okiem jesteśmy w stanie w sposób oczywisty wskazać jaki widzimy znak (np. śnieżyca), jednak nasza sieć sobie nie radzi. Możliwym sposobem naprawienia tej wady, może być bardziej zaawansowany preprocessing, którego zmiana, znacząco wpłynie np. na test rozmazywania, bo krawędzie właśnie, w tym teście, szybko zanikły. Dodatkowo, można spróbować zwiększyć wymiarowość sieci, oraz zastosować bardziej zaawansowane funkcje promujące neurony.