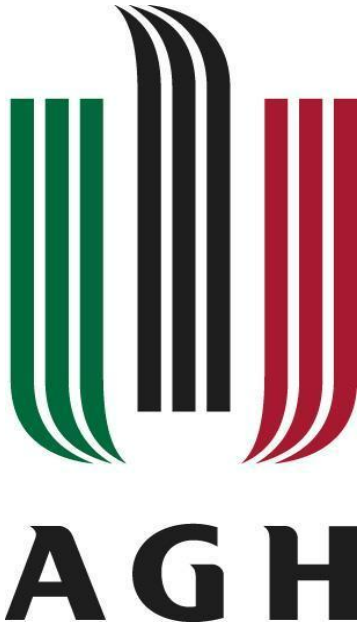
Akademia Górniczo – Hutnicza
im. Stanisława Staszica w Krakowie



Wydział Inżynierii Metali i Informatyki Przemysłowej
Kierunek: Informatyka Stosowana

Projekt z przedmiotu Podstawy Sztucznej Inteligencji

Autor: Agnieszka Bielak
rok III 2016/2017
semestr zimowy

Prowadzący przedmiot:
Dr inż. Dorota Wilk-Kołodziejczyk

Opis projektu:

Problemem którym się zajęłam, oparty o strukturę sieci Kohenena, jest odwzorowaniem kolorów z ich trójwymiarowych elementów - czerwonego, zielonego i niebieskiego, w dwóch wymiarach.

Opis Sieci Kohenena:

Zasady działania sieci Kohonena:

- Wejścia (tyle, iloma parametrami opisano obiekty) połączone są ze wszystkimi węzłami sieci
- Każdy węzeł przechowuje wektor wag o wymiarze identycznym z wektorami wejściowymi
- Każdy węzeł oblicza swój poziom aktywacji jako iloczyn skalarny wektora wag i wektora wejściowego (podobnie jak w zwykłym neuronie)
- Ten węzeł, który dla danego wektora wejściowego ma najwyższy poziom aktywacji, zostaje zwycięzcą i jest uaktywniony
- Wzmacniamy podobieństwo węzła-zwycięzcy do aktualnych danych wejściowych poprzez dodanie do wektora wag wektora wejściowego (z pewnym współczynnikiem uczenia)
- Każdy węzeł może być stowarzyszony z pewnymi innymi, sąsiednimi węzłami - wówczas te węzły również zostają zmodyfikowane, jednak w mniejszym stopniu.

Inicjalizacja wag sieci Kohonena jest losowa. Wektory wejściowe stanowią próbę uczącą, podobnie jak w przypadku zwykłych sieci rozpatrywaną w pętli podczas budowy mapy. Wykorzystanie utworzonej w ten sposób mapy polega na tym, że zbiór obiektów umieszczamy na wejściu sieci i obserwujemy, które węzły sieci się uaktywniają. Obiekty podobne powinny trafiać w podobne miejsca mapy.

Sieci Kohonena są szczególnym przypadkiem algorytmu realizującego uczenie się bez nadzoru. Ich głównym zadaniem jest organizacja wielowymiarowej informacji (np. obiektów opisanych 50 parametrami w taki sposób, żeby można ją było prezentować i analizować w przestrzeni o znacznie mniejszej liczbie wymiarów, czyli mapie (np. na dwuwymiarowym ekranie). Warunek: rzuty "podobnych" danych wejściowych powinny być bliskie również na mapie. Sieci Kohonena znane są też pod nazwami Self-Organizing Maps, Competitive Filters.

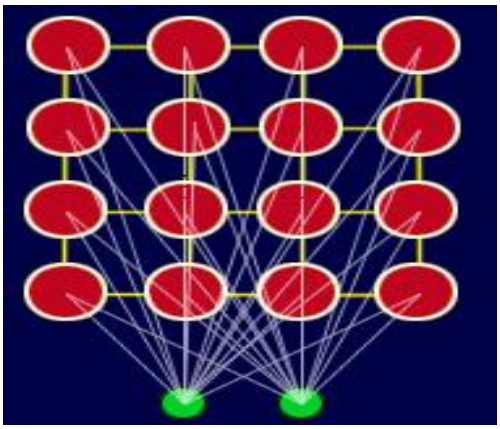
Topologia sieci Kohonena odpowiada topologii docelowej przestrzeni. Jeśli np. chcemy prezentować wynik na ekranie, rozsądnym modelem jest prostokątna siatka węzłów (im więcej, tym wyższą rozdzielczość będzie miała mapa):

Architektura Sieci:

Każda czarna kropka w takiej siatce reprezentuje jeden **neuron** (komórkę nerwową). Główną rolą neuronów jest reagowanie na bodźce. Co najważniejsze, każdy z nich jest przystosowany do tego, by reagować wyłącznie na bodźce z bardzo wąskiego zakresu. Na przykład, gdy mamy do czynienia z siecią reagującą na barwy, neuron odpowiedzialny za reakcję na barwę czerwoną reagowałby bardzo mocno na intensywną czerwień, nieco słabiej na kolor wiśniowy, za to na kolor pomarańczowy, brązowy czy czarny nie reagowałby wcale. Tam, gdzie „milczałby” neuron czerwieni, zareagowaliby jego sąsiedzi. Dzięki temu sieć jako całość może być wrażliwa na całe spektrum barw.

W tym miejscu kluczową rolę odgrywa mapa. Każdy neuron jest połączony z neuronami leżącymi w jego bezpośrednim otoczeniu. Takie otoczenie nie jest przypadkowe. Kontynuując przykład sieci rozpoznającej barwy, neurony uwrażliwione na kolor pomarańczowy albo brązowy leżałyby w bliskim sąsiedztwie neuronu odpowiedzialnego za reagowanie na barwę czerwoną. Z kolei neuronów reagujących na barwę niebieską bądź zieloną, należałoby się spodziewać znacznie dalej. W ten sposób prosta siatka neuronów staje się mapą badanej przestrzeni (w tym przypadku: przestrzeni barw).

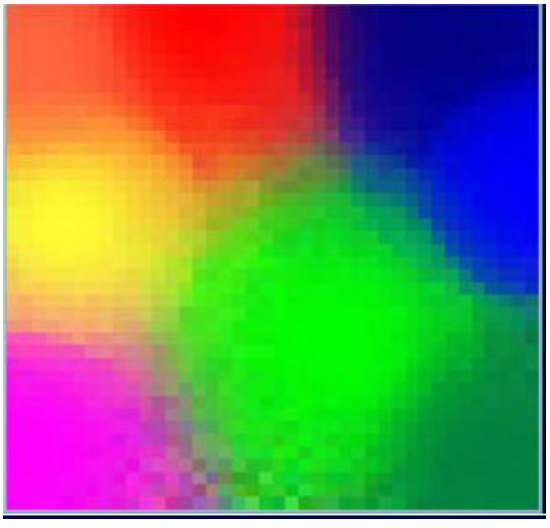
Mamy dwuwymiarową sieć SOM, która jest utworzona z siatki 2D węzłów, z których każdy jest połączony z warstwą wejściową. Poniżej przedstawiono małą sieć Kohonena 4x4, węzły są połączone z warstwą wejściową (kolor zielony):



Każdy węzeł ma specyficzną pozycję topologiczną (współrzędna x i y w siatce) i zawiera wektor wag w tym samym wymiarze co wektor wejściowy.

Linie łączące węzły na rysunku 2 są tam tylko do reprezentowania przylegania i nie oznacza połączenia jak w sieci neuronowej. Nie istnieją żadne powiązania pomiędzy węzłami w siatce.

SOM na poniższym rysunku ma domyślny rozmiar kraty 40 x 40. Każdy węzeł w sieci krystalicznej ma trzy wagi, po jednej dla każdego elementu wektora wejściowego: czerwony, zielony i niebieski. Każdy węzeł jest reprezentowany przez prostokątne komórki na ekranie.



Algorytm uczenia w krokach:

1. Inicjalizacja wag w węzłach. Są to małe wartości losowe. W projekcie wagi inicjowane są wartościami 0 <w <1.

```
public void Inicjalizacja()
{
    if (wejscia == null)
    {
        return;
    }
    for (int i = 0; i < wejscia.Count; i++)
    {
        wagi.Add(random.NextDouble()); //zwraca od 0 do 1
    }
}
```

2. losowy dobór bodźców, na które mają reagować poszczególne neurony. Wektor jest dobierany losowo z zestawu uczącego i jest prezentowany siatce.

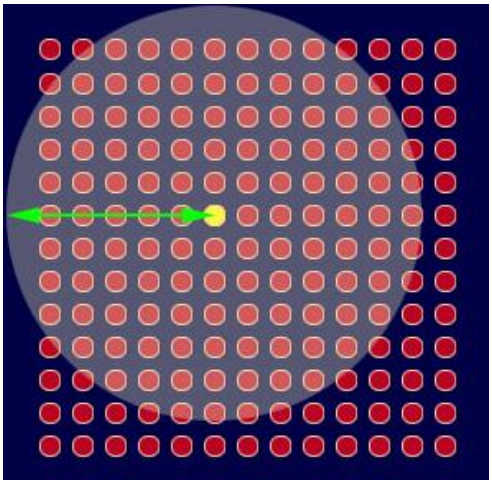
3. Każdy węzeł jest badany w celu obliczenia, która z wag jest najbardziej podobna do wektora wejściowego. Taki węzeł nazywany jest powszechnie najlepszy Matching Unit (BMU). Iterujemy po wszystkich węzłach i obliczamy odległość Euklidesową, pomiędzy każdym wektorem wag węzła, a bieżącym wektorem wejściowym. Odległość Euklidesową zapisujemy jako:

$$Dystans = \sqrt{\sum_{i=0}^{i=n} (V_i - W_i)^2}$$

gdzie V jest bieżącym wektorem wejściowym, a W jest wektorem wag węzła.

```
public double Dystans() //dystans pomiedzy naszym neuronem a szukaną wartością
{
    double suma = 0;

    for (int i = 0; i < wejscia.Count; i++)
    {
        suma += (wejscia[i].wyjscie - wagi[i]) * (wejscia[i].wyjscie - wagi[i]);
    }

    return Math.Sqrt(suma);
}
```

4. Promień sąsiadów BMU został obliczony. Początkowo wartość jest duża, lecz w każdym kolejnym kroku ulega zmniejszeniu. Węzły, które zostały znalezione w zasięgu promienia są uważane za sąsiadów BMU.



Powyżej zielona strzałka pokazuje promień.

Unikalną cechą algorytmu uczenia Kohonena jest to, że obszar zasięgu sąsiadów zmniejsza się w miarę upływu czasu. Wagi każdego sąsiedniego węzła (znalezione w kroku 4) są korygowane, aby uczynić je zbliżone do wektora wejściowego. Im bliżej węzeł jest BMU, tym bardziej jego wagi się zmieniają.

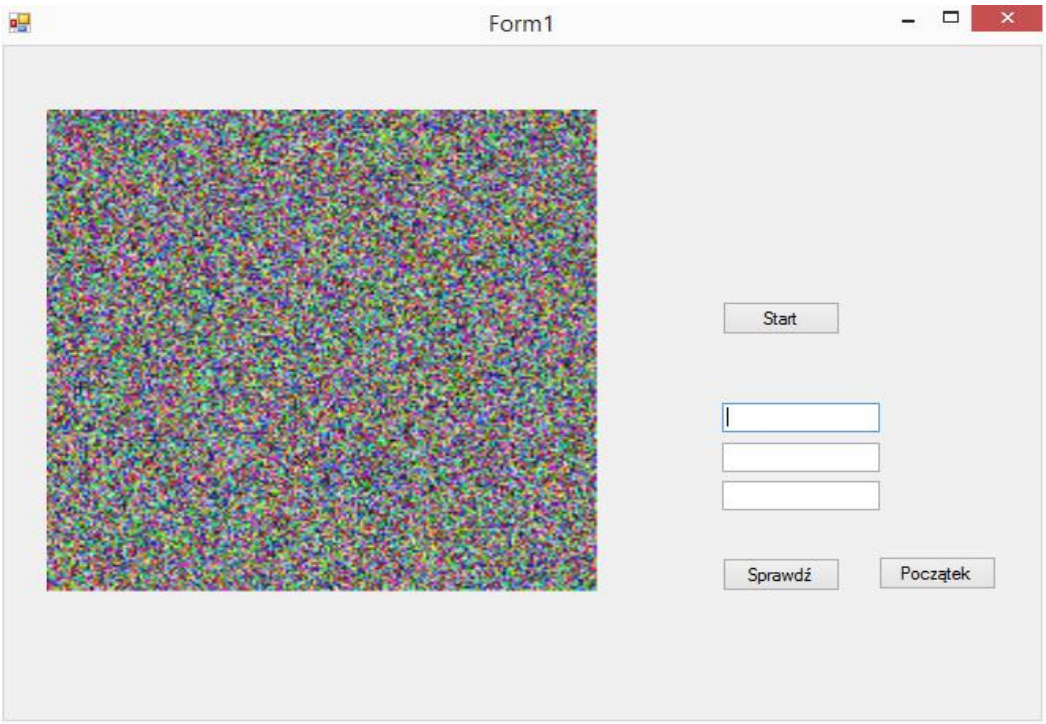
Chociaż kolor jest renderowany przez komputer przy użyciu wartości dla każdego składnika (czerwony, zielony i niebieski) od 0 do 255, wektory wejściowe zostały dostosowane tak, aby każdy składnik posiadał wartość z przedziału od 0 do 1. (Po to, aby dopasować zakres wartości użytych do wag węzłów).

Zestaw uczący składa się z losowych kolorów:

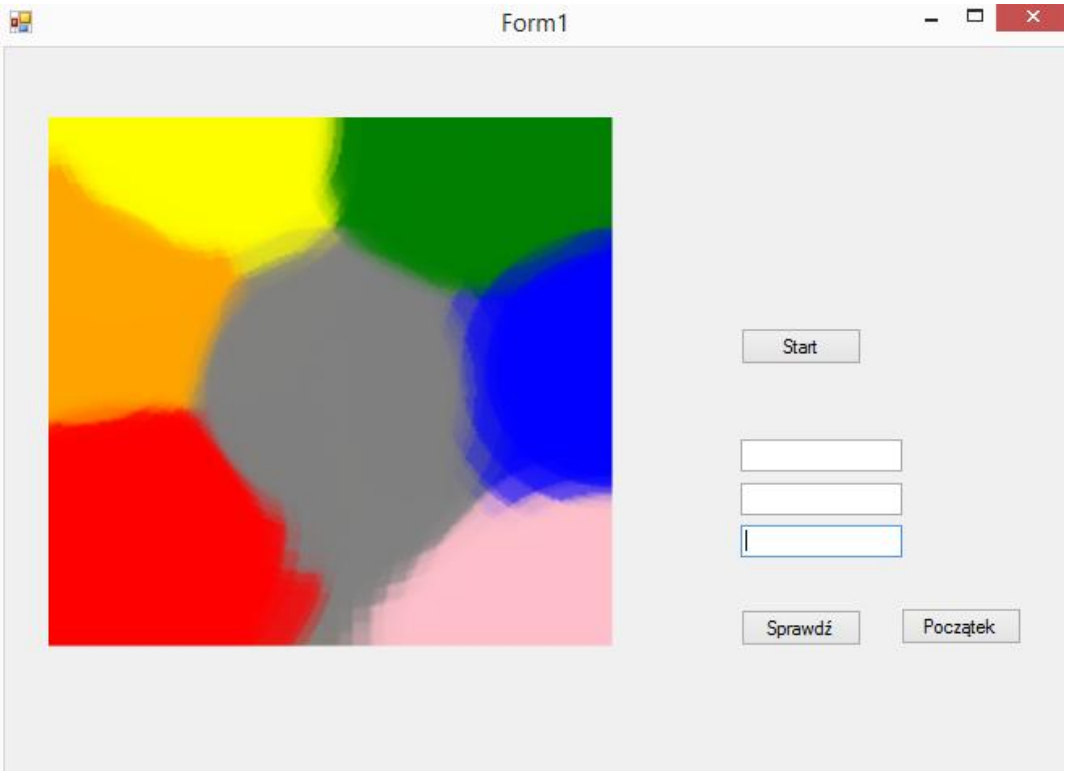
```
public Form1()
{
    siec = new Siec(200,200);
    InitializeComponent();
    pictureBox1.BackgroundImage = new Bitmap(
    List<Color> kolory = new List<Color>();
    kolory.Add(Color.Red);
    kolory.Add(Color.Yellow);
    kolory.Add(Color.Blue);
    kolory.Add(Color.Orange);
    kolory.Add(Color.Pink);
    kolory.Add(Color.Gray);
    kolory.Add(Color.Green);

    siec.Start(kolory, 1000);

}
```

Pokaz działania:

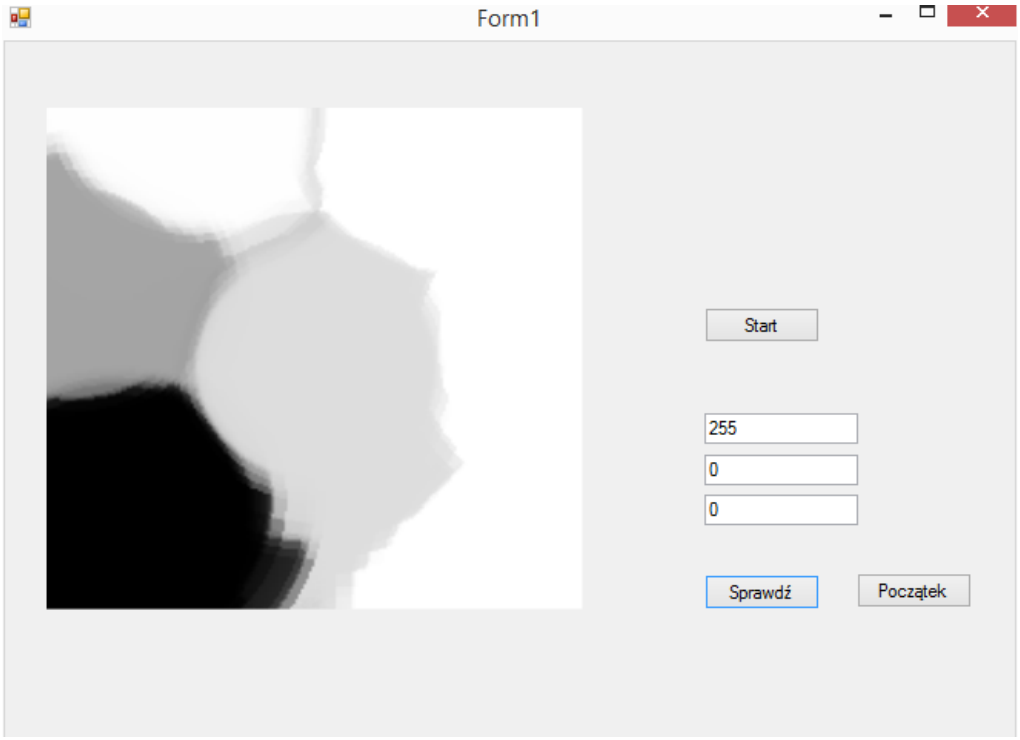
Ekran startowy aplikacji, na którym jest bitmapa kolorów.



Przyciśnięcie przcisku "Start" powoduje uruchomienie wszystkich funkcji odpowiedzialnych za naukę.



Sprawdzenie czy program rozpozna barwę czerwoną (RGB (255,0,0)). Tam gdzie był kolor czerwony, obszar został pokolorowany na czarno.



Sprawdzenie czy program rozpozna barwę niebieską (RGB (0,0,255)). Tam gdzie był kolor niebieski, obszar został pokolorowany na czarno.