Sprawozdanie 2 - Sieci Kohonena

Jan Pogłód

April 2024

1 Wstęp

Celem sprawozdania było opisanie implementacji i rezultatów sieci Kohonena dla problemu klasteryzacji danych. Problem klasteryzacji jest dziedziną nienadzorowanego uczenia maszynowego, w której dzielimy dane na grupy w taki sposób, aby obiekty w tych samych grupach były bardziej podobne do siebie niż do obiektów w innych grupach. Problem uczenia maszynowego nienadzorowanego polega na przetwarzaniu danych, w którym algorytm uczący nie ma dostępu do etykiet danych. W przeciwieństwie do uczenia nadzorowanego (regresji, klasteryzacji), gdzie korzystamy z implementacji sieci neuronowej Multi-Layer-Perceptron, tym razem do rozwiązania problemu omówimy działanie sieci Kohonena.

2 Dane do przeprowadzania eksperymentów

W naszej pracy będziemy dążyć do skutecznego pogrupowania danych "hexagon" oraz "cube". Pierwszy z tych zbiorów jest zbiorem na przestrzeni dwuwymiarowej, ze zmiennymi x oraz y, i reprezentuje wierzchołki sześciokąta. Drugi zbiór - "cube" jest z przestrzeni o trzech wymiarach x,y,z i reprezentuje wierzchołki sześcianu.

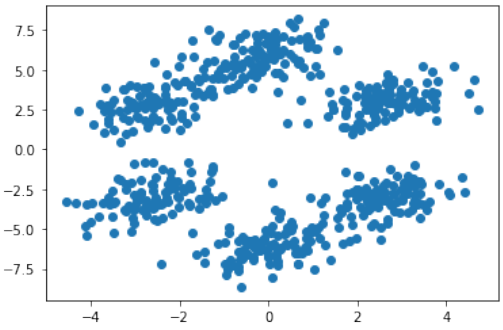


Figure 1: Dane "hexagon" bez podziału na grupy

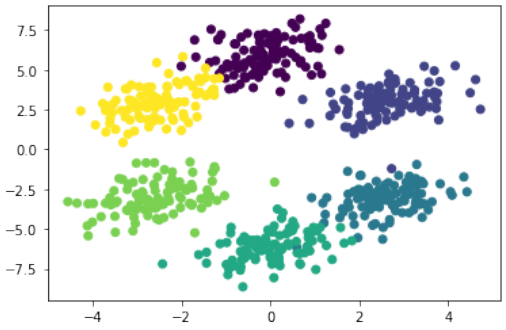


Figure 2: Dane "hexagon" z podziałem na grupy, niedostępne w procesie uczenia

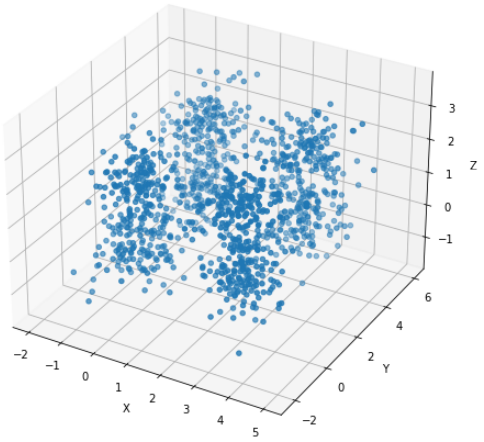


Figure 3: Dane "Cube" bez podziału na grupy

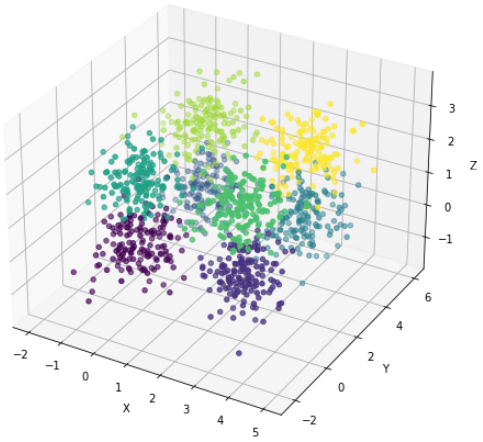
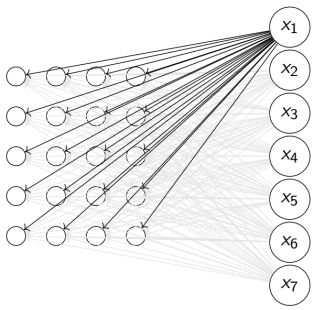


Figure 4: Dane "Cube" z podziałem na grupy, niedostępne w procesie uczenia

W przeciwieństwie do powszechnych algorytmów uczenia maszynowego oraz sieci neuronowych minimalizujących funkcję kosztu, która zawiera dane treningowe i odpowiadające im wartości przewidujące (na przykład znana wartość kursu waluty dla obserwacji czy znana klasa ryzyka kredytowego), w tym przypadku proces uczenia nie będzie miał dostępu do takich danych. Proces uczenia przeprowadzać będziemy na samych punktach w przestrzeni, a więc uczyć będziemy zbiory "Figura 1" i "Figura 3". W tym celu przedstawimy implementację sieci Kohonena.

3 Implementacja sieci Kohonena

Sieć Kohonena implementujemy na danej siatce prostokątnej MxN. Siatka zbudowana jest z neuronów i działa dla zbioru wektorów x_1,x_2, ...,x_n, na których podstawie grupujemy wektory dla danej, najlepiej rozłożonej grupy. Za pomocą manipulowania wagami sieć Kohonena wybiera neuron, który jest najbliżej tej grupy (wzorca) i "przesuwa" go w kierunku tego wzorca. Następnie z pewnym osłabieniem przesuwani są sąsiedzi tego wzorca, aby wzmocnić siłę (odległość) grupy. Przykładową architekturę sieci Kohonena na siatce MxN przedstawiono poniżej.



Wprowadzmy podstawowe oznaczenia do implementacji. Niech $w_{i,j}^k$ oznacza wagę pomiędzy wejściem *k=1,...,n*, a neuronem na pozycji *(i,j)*, *i=1,..,M, j=1,...,N*. Ponadto jako metrykę w przestrzeni danych wprowadzimy odległość euklidesową.

$$d(\mathbf{w}, \mathbf{x}) = \sqrt{\sum_{i=1}^{n}(w_{i,j} - x_i)^2} \tag{1}$$

Wprowadźmy również funkcję, która liczyć będzie wagę sąsiedztwa względem odległości euklidesowej między dwoma danymi punktami n1 i n2. Punkty te reprezentują odpowiednio zwycięski neuron (obliczony za pomocą odległości euklidesowej pomiędzy wagą $w_{i,j}$ oraz wektorem x jak w (1) oraz kolejny neuron o indeksie (i,j) z danej siatki MxN. Rozważać będziemy dwie funkcje sąsiedztwa.

1) Funkcja Gaussa:

$$\theta_1(n_1, n_2, t) = f(d(n_1, n_2), t) = e^{-(d(n_1,n_2)t)^2}$$

1) Minus druga pochodna funkcji Gaussa:

$$\theta_2(n_1, n_2, t) = f(d(n_1, n_2), t) = (2 - 4(xt)^2)e^{-(d(n_1,n_2)t)^2}$$

Jeżeli założymy, że (i*,j*) to zwycięski neuron, obliczony za pomocą formuły

$$(i*, j*) = \arg\min_{i,j} d(w_{i,j}, x)$$

to wówczas otrzymujemy następującą metodę na otrzymywanie aktualizacji wag, w kolejnych iteracjach sieci Kohonena. t=1,...,λ.

$$w_{i,j} \leftarrow \quad w_{i,j} \quad + \quad \theta((i^*, j^*), (i, j), t) \cdot \alpha(t) \cdot (x - w_{i,j}) \tag{2}$$

W powyższym wzorze wyrażenie $\alpha(t)$ jest funkcją wygaszania w czasie. Funkcja ta określa w jaki sposób maleje wpływ sąsiedztwa neuronów podczas treningu sieci. W początkowej fazie treningu, sąsiedztwo neuronów jest szerokie, co pozwala na szybszą eksplorację przestrzeni wejściowej. W miarę postępu treningu, sąsiedztwo neuronów stopniowo zmniejsza się, aby umożliwić lepszą stabilizację przypisywania danego wektoru do grupy. W celu wygaszania funkcji sąsiedztwa w czasie stosować będziemy następującą funkcję.

$$\alpha(t) = \eta \exp\left(-\frac{t}{\lambda}\right) \qquad \eta \in (0, 1)$$

Przy testach mierzyć będziemy wpływy różnych parametrów takich jak liczbę epoch, długość szerokości sąsiedztwa wymiar siatki oraz rodzaj funkcji sąsiedztwa. Szerokość sąsiedzwa modyfikować będziemy za pomocą parametru $\psi \in$ [0.1,1].

$$\theta(n_1, n_2, t) = f(\psi d(n_1, n_2), t) = e^{-(\psi d(n_1,n_2)t)^2}$$

Ponadto zwizualizujemy proces uczenia sieci za pomocą zmiany miary sumarycznej odległości punktów od siebie w klastrach pomiędzy kolejnymi epokami treningu sieci Kohonena.

4 Testowanie rezultatów implementacji sieci Kohonena

4.1 Zbiór hexagon

W przypadku zbioru "hexagon" rozważaliśmy siatkę 2x3, 3x2 oraz 5x5. Ponadto liczbę epoch ustawiliśmy na 100 i 1000 epok, szerokość sąsiedztwa na 1, 0.5, 0.3 i 0.9, a krok uczenia na $\eta = 0.1$. Sieć neuronowa bez problemu pogrupowała wektory według pasujących grup i zmniejszała sumaryczną odległość punktów w klastrach.

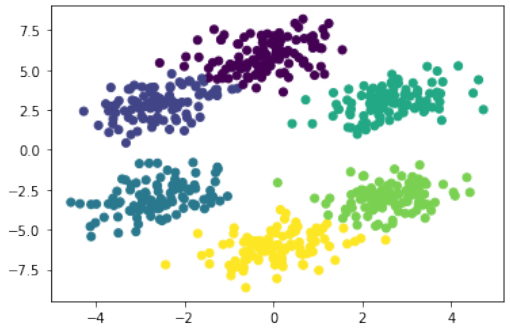


Figure 5: Rezultat wytrenowanej sieci - Suma odległości punktów w klastrach = **1.69**

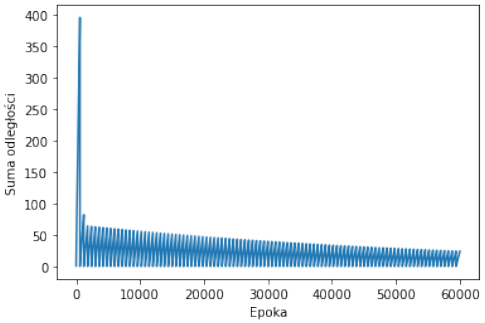


Figure 6: Sumaryczna odległość punktów w klastrach w zależności od epoch dla kolejnych neuronów

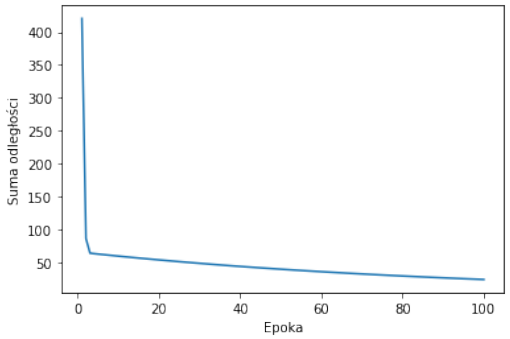


Figure 7: Sumaryczna odległość punktów w klastrach w zależności od 100 epok

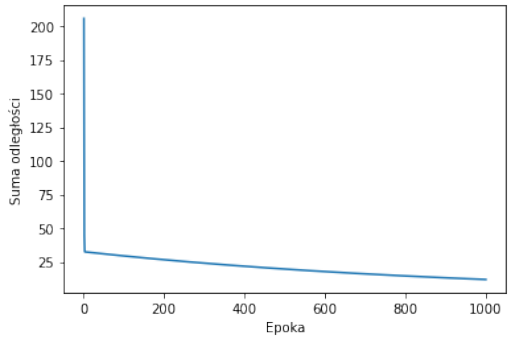


Figure 8: Sumaryczna odległość punktów w klastrach w zależności od 1000 epok

Poniżej zamieściliśmy końcowe rezultaty czterech z badanych sieci. Możemy zauważyć, że funkcja Gaussa θ_1 jako funkcja sąsiedztwa okazała się być lepsza dla tego zbioru. Ponadto Najlepszy wynik otrzymaliśmy dla siatki 2x3, $\psi = 0.3$, a więc najmniejszych z badanych szerokości siąsiedztwa oraz liczby epok równej 1000. Do uzyskania porównania na kilku zbiorach użyliśmy normalizacji min-max.

Liczba epoch	Siatka	Szerokość sąsiedztwa	Funkcja sąsiedztwa	Sum. odległość punktów w klastrach
100	2x3	$\psi = 1$	θ_1	$\rightarrow$ **3.30**
100	5x5	$\psi = 0.9$	θ_2	$\rightarrow$ **12.93**
1000	2x3	$\psi = 0.3$	θ_2	$\rightarrow$ **1.69**
1000	3x2	$\psi = 0.5$	θ_1	$\rightarrow$ **3.70**

4.2 Zbiór Cube

Zbiór "Cube" rozważaliśmy na siatce 3x3, 5x5 oraz 4x3. Ponadto liczbę epoch ustawiliśmy na 1000 epok, szerokość sąsiedztwa podobnie jak w poprzednim problemie na 1, 0.5, 0.3 i 0.9, a krok uczenia na $\eta = 0.01$. W tym przypadku problem grupowania, okazał się być dużo trudniejszy, ale także sieć Kohonena znalazła w danych odpowiednie wzorce.

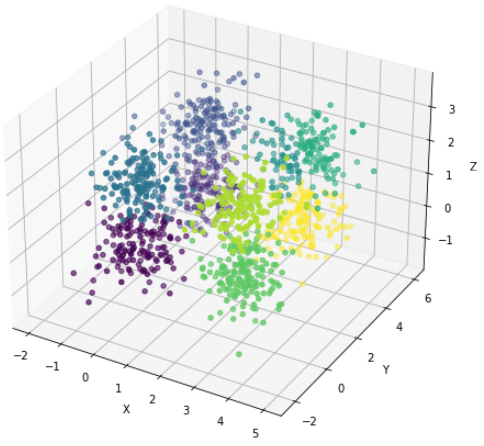


Figure 9: Rezultat wytrenowanej sieci - Suma odległości punktów w klastrach = **3.96**

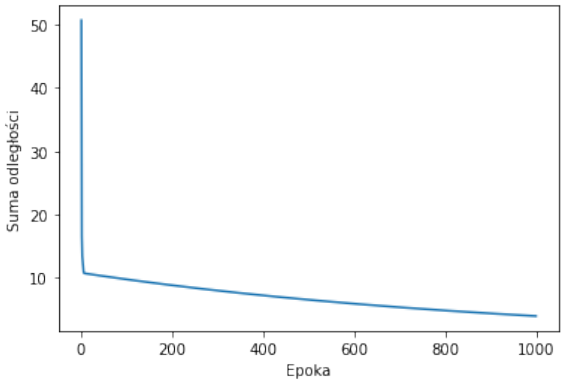


Figure 10: Sumaryczna odległość punktów w klastrach w zależności od epoch dla kolejnych neuronów

Ponownie zamieściliśmy końcowe rezultaty czterech z badanych sieci. Możemy zauważyć, że funkcja Gaussa θ_1 jako funkcja sąsiedztwa również okazała się być lepsza dla tego zbioru. Ponadto Najlepszy wynik otrzymaliśmy dla siatki 3x3, $\psi = 1$, a więc tym razem największej z badanych szerokości siąsiedztwa oraz liczby epok równej 1000.

Liczba epoch	Siatka	Szerokość sąsiedztwa	Funkcja sąsiedztwa	Sum. odległość punktów w klastrach
1000	3x3	$\psi = 1$	θ_1	$\rightarrow$ **3.96**
1000	5x5	$\psi = 0.5$	θ_2	$\rightarrow$ **4.27**
1000	4x3	$\psi = 0.3$	θ_2	$\rightarrow$ **4.86**
100	3x3	$\psi = 0.9$	θ_1	$\rightarrow$ **5.98**

4.3 Zbiór Moon

Jako trzeci z badanych zbiorów rozważyliśmy zbiór "Moon", który dzieli przestrzeń dwuwymiarową na dwa księżyce. Ponownie sieć Kohonena pogrupowała dane prawidłowo. Zbiór ten jest dobrym

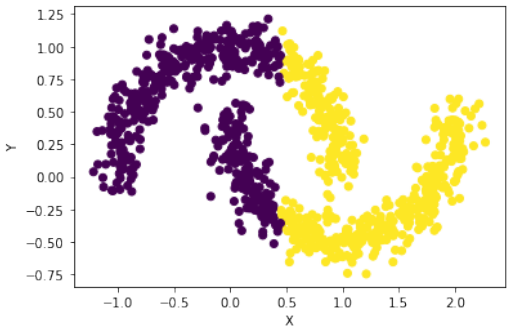


Figure 11: Rezultat wytrenowanej sieci - Suma odległości punktów w klastrach = **3.96**

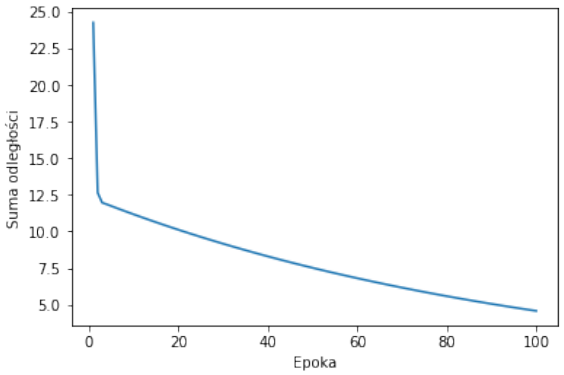


Figure 12: Sumaryczna odległość punktów w klastrach w zależności od epoch dla kolejnych neuronów

Liczba epoch	Siatka	Szerokość sąsiedztwa	Funkcja sąsiedztwa	Sum. odległość punktów w klastrach
1000	3x3	$\psi = 1$	θ_1	$\rightarrow$ **8.12**
1000	2x1	$\psi = 0.5$	θ_2	$\rightarrow$ **2.42**

przykładem, ponieważ mimo iż dostaliśmy małe sumy odległości punktów w klastrach i minimalizacje tych odległości w kolejnych epokach, to widzimy, że klasteryzacja nie przebiegła po oddzielonych księżycach lecz po najbliższych sobie punktach w metryce euklidesowej.

4.4 Sześciokątna siatka w sieci Kohonena

W przypadku sieci Kohonena zmodyfikować możemy równiez topologię siatki neuronów. W powyższych przykładach używaliśmy siatki w topologi prostokątnej, w której neurony ułożone są jeden obok drugiego oraz jeden pod drugim. Możemy przetestować jak zmiana topologi na sześciokątną wpłynie na zachowanie się sieci. Topologię sześciokątną możemy zaimplementować przesuwając odpowiednie neurony, aż do osięgnięcia oczekiwanego wzorca siatki.

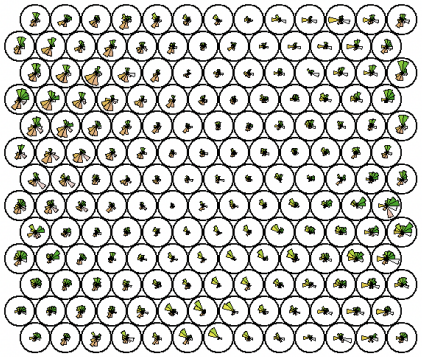


Figure 13: Wizualizacja siatki sześciokątnej z artykułu
https : //clarkdatalabs.github.io/soms/SOM_NBA

Aby zaimplementować tę nową topologię w naszej sieci Kohonena konieczna jest zmiana iterowania neruonów z (i,j) ∈ (M,N) w iterację (i,j) ∈ (σ, $2\sigma - 1$), gdzie σ oznacza liczbę sześciokątów w siatce oraz wyraża odpowiednie dla nich wierzchołki. Po aktualizacji siatki i przetestowania wyników dla zbioru hexagon dostaliśmy wyniki.

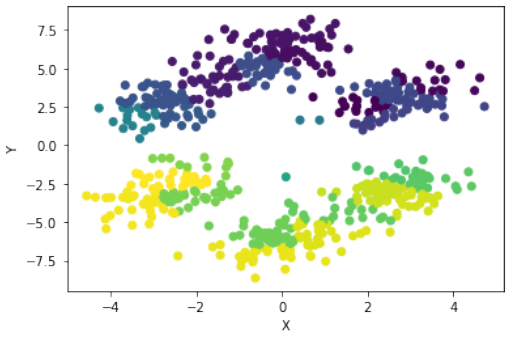


Figure 14: Rezultat wytrenowanej sieci - Suma odległości punktów w klastrach = **3.96**

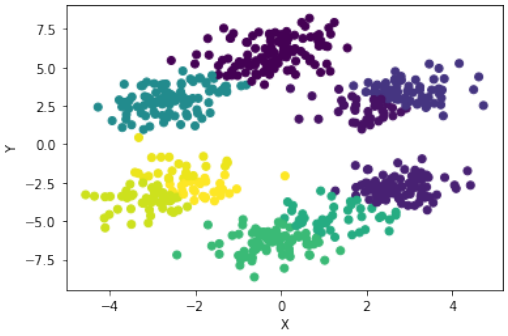


Figure 15: Sumaryczna odległość punktów w klastrach w zależności od epoch dla kolejnych neuronów

Liczba epoch	Liczba sześciokątów	Szerokość sąsiedztwa	Funkcja sąsiedztwa	$\sum$
1000	1	$\psi = 0.5$	θ_1	$\rightarrow$ **6.61**
1000	6	$\psi = 0.5$	θ_2	$\rightarrow$ **37.22**

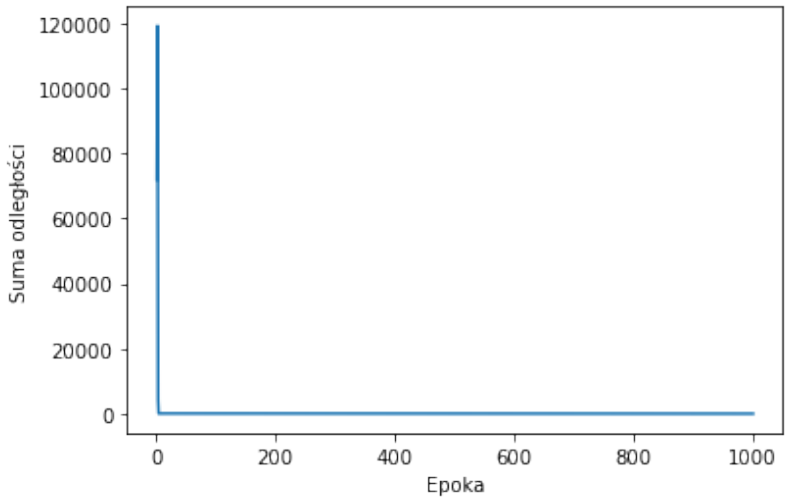


Figure 16: Suma odległości punktów w klastrach w zależności od epoch

5 Podsumowanie

Sieci Kohonena choć są jednym z wielu algorytmów do problemu klasteryzacji, to ich interesujący sposób działania i implementacja otwiera wiele możliwości do badań. Testowanie różnych parametrów, funkcji sąsiedztwia i topologii może przynieść korzystne rezultaty. Ponadto widzimy, że w przypadku prostych zbiorów jak hexagon, korzystanie z najprostrzej strukty sieci daje najlepsze wyniki, co potwierdza atuty tego algorytmu do klasteryzacji danych w dwu jak i wielo wymiarowych zbiorach.