Sieci Kohonena raport

Laura Hoang, 313366

Maj 2023

1 Wprowadzenie

Sieci Kohonena, znane również jako Self-Organizing Map (SOM), są formą sieci neuronowych nienadzorowanych, które wykorzystują samoorganizujące się cechy, aby mapować i grupować dane wielowymiarowe. Ich głównym celem jest przedstawienie topologicznej reprezentacji danych, w której podobne elementy są blisko siebie na mapie, podczas gdy elementy różniące się są oddalone.

Proces uczenia sieci Kohonena składa się z dwóch etapów: inicjalizacji i adaptacji. W fazie inicjalizacji wagi są losowo przypisywane do neuronów na mapie, tworząc początkową strukturę. Następnie, w fazie adaptacji, sieć jest trenowana na zbiorze danych, a neurony są dostosowywane tak, aby lepiej odzwierciedlały charakterystyki danych wejściowych.

Rozwiązanie problemów za pomocą sieci Kohonena ma wiele zastosowań w różnych dziedzinach, m.in. w analizie danych, klasteryzacji danych, redukcji wymiarowości, przetwarzaniu obrazów i wielu innych. W ramach zajęć, zostały one wykorzystane do klasteryzacji zwykłych danych liczbowych dwu- i trójwymiarowych (na laboratorium KOH1), oraz klasyfikacji aktywności i obrazów (laboratorium KOH2).

W niniejszym raporcie opisane zostaną doświadczenia przeprowadzone w celu zbadania działania kilku zaimplementowanych wariantów sieci Kohonena, interpretacje otrzymanych wyników oraz wnioski na temat uzyskanych mapowań.

2 KOH1: Podstawowa sieć Kohonena

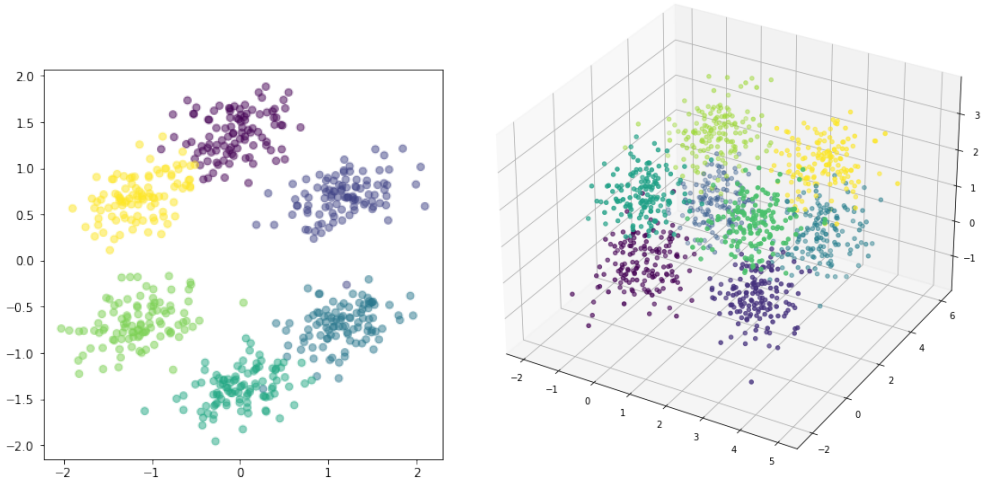
W SOM dane wejściowe są mapowane na siatkę neuronów (w tym przypadku jest ona dwuwymiarowa), gdzie każdy neuron reprezentuje wektor wag. W ramach tej części zaimplementowany został najprostszy wariant sieci Kohonena złożonej z neuronów w prostokątnej siatce o wymiarach $M \times N$, która działa dla zbioru wektorów, wszystkich o tej samej długości.

Zaimplementowane zostały następujące funkcje sąsiedztwa:

- funkcja gaussowska,
- funkcja gaussowska minus jej druga pochodna (znana jako "Mexican Hat").

W obu implementacjach dodana została możliwość zmiany szerokości sąsiedztwa z użyciem parametru t i została przetestowana dla kilku wartości z przedziału $[0.1, 1]$. Jako funkcji wygaszającą uczenie wraz z kolejnymi iteracjami należało użyć funkcji $\alpha(t) = e^{-t/\lambda}$.

Testowanie działania sieci zostało przeprowadzone na dwóch prostych zbiorach danych: **Hexagon** - dwuwymiarowych, skupionych w wierzchołkach sześciokąta oraz **Cube** - trójwymiarowych, skupionych na wierzchołkach sześcianu.



Rysunek 1: Wizualizacja zbioru **Hexagon** (lewo) i **Cube** (prawo) z uwzględnieniem oryginalnych klas.

Na obu zbiorach testowane były różne warianty modeli. Pod uwagę brane były odmienne wymiary siatek[1], wartości parametru szerokości t: dla pierwszego zbioru $t \in \{0.1, 0.2, 0.5, 1\}$, a dla drugiego $t \in \{0.4, 1\}$, oraz 2 funkcje sąsiedztwa: gaussowska i Mexican Hat. Uczenie zostało wykonane bez wartości etykiet na obu zbiorach, zostały one użyte tylko do porównania wyników.

Aby ocenić skuteczność modeli wykorzystana została metryka **v-measure**[2], której zaletą jest niezależność od wartości bezwzględnych etykiet, czyli permutacja wartości etykiet klas lub klastrów w żaden sposób nie zmieni wyniku.

Wyniki doświadczeń prezentują się następująco:

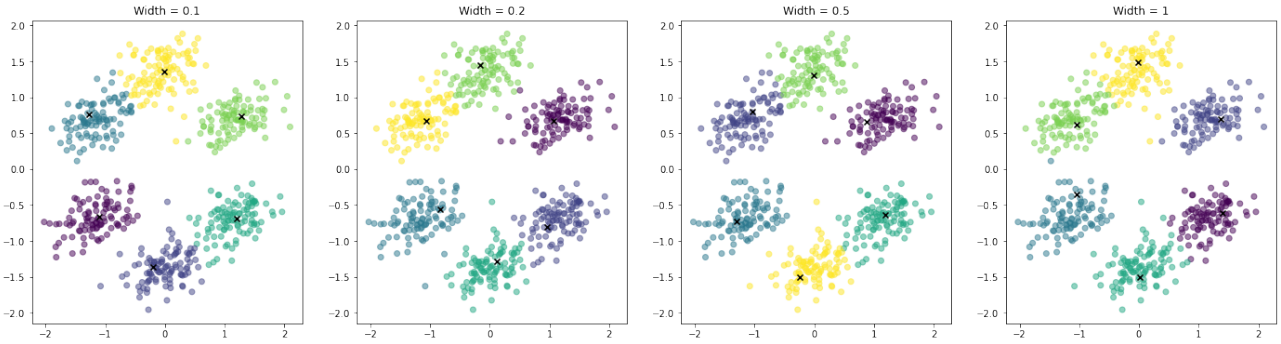
Wymiary siatki	**gaussian**				**mexican**			
	0.1	**0.2**	**0.5**	**1**	**0.1**	**0.2**	**0.5**	**1**
(2, 3)	0.941	0.94	0.95	0.946	0.792	0.875	0.883	0.931
(3, 3)	0.85	0.862	0.844	0.838	0.834	0.759	0.75	0.842
(4, 4)	0.763	0.766	0.759	0.759	0.728	0.761	0.751	0.768

Tabela 1: Wartości v-measure dla zbioru **Hexagon**.

Na zbiorze **Hexagon** zdecydowanie najlepiej radzi sobie metryka z funkcją gaussowską, na wymiarze siatki 2×3. Osiąga on wartość v-measure bardzo bliską 1, co świadczy o dobrym działaniu modelu. Jest to sensowny wynik, gdyż wówczas liczba neuronów odpowiada faktycznej liczbie klas ze zbioru. Natomiast, porównując gaussian z mexican, w większości wypadków pierwszy wariant osiąga lepsze wyniki.

[1] W przesłanej wersji kodu testowane było tylko 4×4 oraz 3×3. Jednak, jak się okazało, lepsze wyniki wychodziły dla innych wymiarów. Oprócz tego, w nowszej wersji kodu nie jest już wykorzystana funkcja grupująca (ograniczająca) końcowo otrzymane punkty.

[2] W przesłanym kodzie nie była jeszcze używana.



Rysunek 2: Wizualizacja najlepszych klasteryzacji dla zbioru **Hexagon**, model gaussian na 2×3.

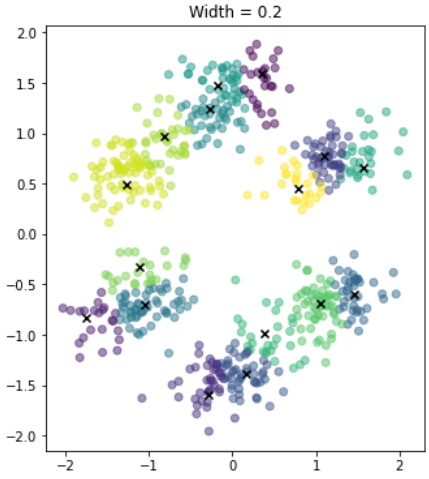
Oprócz tego można zaobserwować, iż Mexican Hat lepiej sobie radzi dla $t = 1$. Być może wynika to z faktu, iż przy doborze t mniejszym niż 1, wykres funkcji mexican "rozciąga się"w poziomie, co wiąże się z tym, iż spadek tej funkcji sięga dalszych punktów i niepotrzebnie wypycha je dalej.

Wymiary siatki	**gaussian**		**mexican**	
	0.4	**1**	**0.4**	**1**
(2, 3)	0.793	0.827	0.742	0.814
(3, 3)	0.866	0.882	0.853	0.806

Tabela 2: Wartości v-measure dla zbioru **Cube**.

W przypadku zbioru **Cube** nie dostrzega się ogromnych różnic między gaussian i mexican. Jedyne co ciekawe, to w tym wypadku obie funkcje lepiej zadziałały dla $t = 1$ i dla siatki 3×3. Zatem, w przeciwieństwie do poprzedniego zbioru, lepsza dokładność została osiągnięta na liczbie neuronów nieodpowiadających oryginalnemu podziałowi. Jednakże różnica wyników nie jest na tyle duża, aby mówić o zdecydowanej lepszości któregoś wariantu.

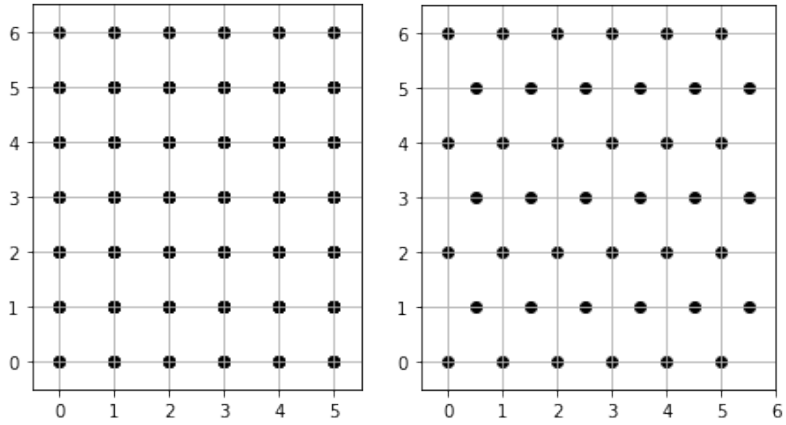
Klastry w odwzorowaniu znalezionym przez sieć nie zawsze pokrywały się w liczbą klastrów w faktycznych danych. Wynikało to z doboru parametrów modelu: liczba neuronów w siatce, rozmiar sąsiedztwa i współczynniki uczenia. Przy doborze wymiarów siatki które dają liczbę neuronów różną od liczby klastrów, otrzymane jest tyle klas, ile neuronów w siatce. Jednak podczas mapowania neurony te zbliżają się do skupisk punktów ze zbioru danych, niekiedy kumulując się blisko siebie, poniekąd jakby miały "połączyć się"w jeden neuron, który odpowiadałby danemu klastrowi. Natomiast, oczywiście, znalezione klastry pokrywają się z identyfikatorami wierzchołków.



Rysunek 3: Odzwierciedlenie liczby neuronów do otrzymanej liczby klas; zbiór **Hexagon**, siatka 4×4.

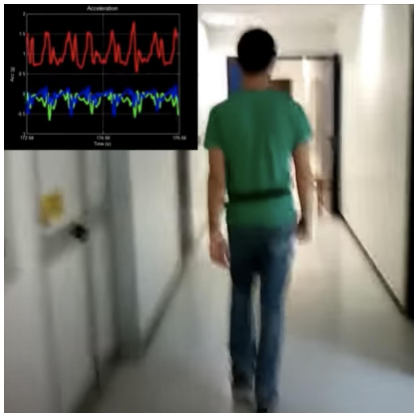
3 KOH2: Sieć Kohonena na siatce sześciokątnej

W ramach tej części, należało dodać do modelu nową topologię: siatkę sześciokątną. Ta topologia jest modyfikacją poprzedniej. Kolejne wiersze są przesunięte o połowę odległości między sąsiadującymi neuronami w poprzednim wierszu, co tworzy efekt sześciokątnej struktury. Każdy neuron jest połączony z kilkoma sąsiednimi neuronami, a odległości między neuronami są równe dla wszystkich sąsiadów.



Rysunek 4: Porównanie schematu siatki prostokątnej (lewo) i sześciokątnej (prawo). Neurony są połączone między sobą (na schemacie nie zostało uwzględnione), a sąsiedztwo można samodzielnie określić.

W celu porównania działania obu topologii należało przetestować je na zbiorach **Human Activity Recognition Using Smartphones** oraz **MNIST**. Zbiór **HAR** opisuje pewne czynności wykonywane przez człowieka, które zostały zarejestrowane za pomocą smartfona. Sześć badanych czynności: chodzenie, wchodzenie po schodach, schodzenie ze schodów, siedzenie, stanie i leżenie, są klasami w przeprowadzonych doświadczeniach.



Rysunek 5: Zdjęcie z przeprowadzanego badania nad **Human Activity Recognition**. [1]

Natomiast **MNIST** to znany zbiór danych zawierający ogromną ilość odręcznie zapisanych cyfr. Jest on powszechnie używany do uczenia różnych systemów przetwarzania obrazu, jak również szeroko wykorzystywany do szkolenia i testowania modeli z zakresu uczenia maszynowego. W przypadku przeprowadzonych doświadczeń w ramach tego zadania, klasami są cyfry od 0 do 9.

Rysunek 6: Przykładowe obrazy z zestawu danych testowych **MNIST**. [2]

Testy dla obu zbiorów zostały dokonane na 3 różnych wymiarach siatek, 2 topologiach (kwadratowej i sześciokątnej) oraz 2 metrykach (funkcji gaussowskiej i Mexican Hat). W przypadku tej części, doświadczenia zostały przeprowadzone jedynie na szerokości siatek $t = 1$.

W celu oceny skuteczności modeli, jak w przypadku poprzedniego laboratorium, znowu została wykorzystana metryka **v-measure**. Ponieważ są to zbiory o dużej liczbie obserwacji, liczba epoch do każdego doświadczenia została zredukowana do 5.

Wyniki doświadczeń prezentują się następująco:

Wymiary siatki	**rectangle**		**hexagon**	
	gaussian	**mexican**	**gaussian**	**mexican**
(2, 3)	0.482	0.59	0.517	0.431
(4, 4)	0.532	0.542	0.486	0.46
(6, 6)	0.5	0.479	0.508	0.485
Średnie wyniki:	**0.504**	**0.537**	**0.503**	**0.459**

Tabela 3: Wartości v-measure dla zbioru **HAR**.

Wymiary siatki	**rectangle**		**hexagon**	
	gaussian	**mexican**	**gaussian**	**mexican**
(3, 3)	0.294	0.262	0.319	0.253
(4, 4)	0.369	0.287	0.348	0.278
(6, 6)	0.444	0.373	0.432	0.366
Średnie wyniki:	**0.369**	**0.307**	**0.366**	**0.299**

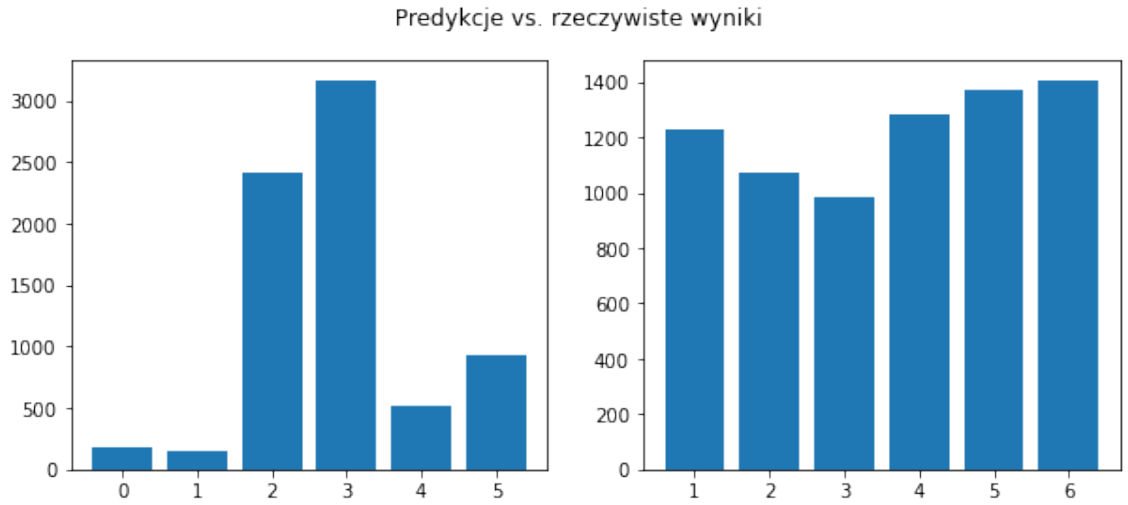
Tabela 4: Wartości v-measure dla zbioru **MNIST**.

Wartości metryki pokazują, że klastrowanie nie do końca odpowiada podziałowi na klasy (w szczególności dla drugiego zbioru). Być może wynika to ze zbyt małej liczby iteracji.

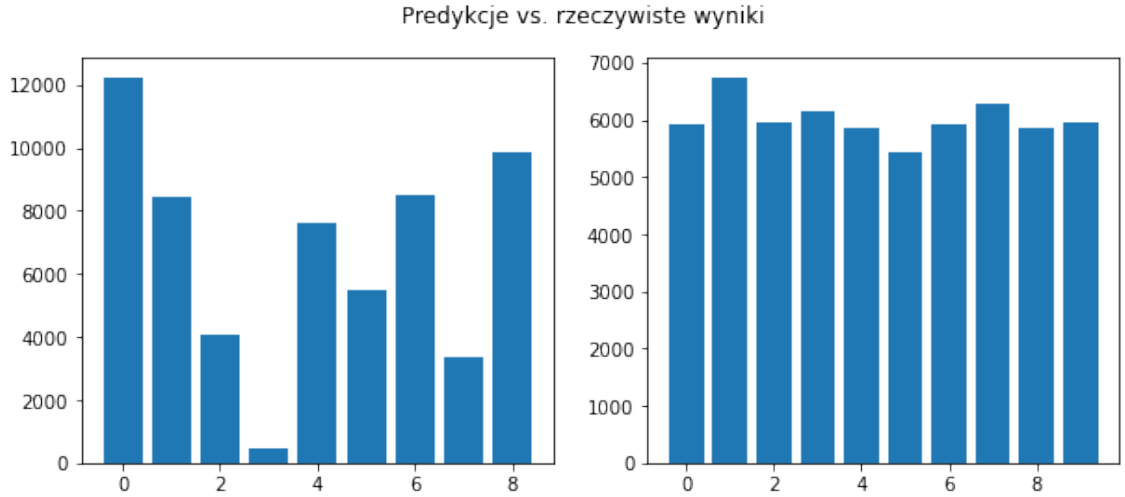
Porównując wszystkie warianty modeli, dla pierwszego zbioru najlepiej wypada siatka prostokątna z Mexican Hat, w szczególności dla wymiaru 2×3, co ma sens, gdyż zgadza się to z rzeczywistą liczbą klas w zbiorze. Wyniki dla Mexican Hat wypadły zdecydowanie lepiej dla tego przypadku, niż dla siatki sześciokątnej.

Z kolei zbiór **MNIST** najlepiej wypadł dla obu wariantów siatek z funkcją gaussowską, gdzie najwyższy wynik osiągnął model o wymiarach 6×6 z siatką prostokątną. Jednak, warto podkreślić, iż wyniki dla obu siatek w każdym wariancie są bardzo zbliżone do siebie, zatem nie można mówić o większej skuteczności którejś z nich (przynajmniej w tym wypadku).

Poniżej przedstawione są przykładowe wyniki klastrowania porównane z rzeczywistymi licznościami poszczególnych klas:



Rysunek 7: Wizualizacja dla zbioru **HAR**; model rectangle-mexican 2×3.



Rysunek 8: Wizualizacja dla zbioru **MNIST**; model hexagon-gaussian 3×3.

Niestety, wizualizacje potwierdziły, że klastry w obydwóch przypadkach nie do końca odpowiadały rzeczywistemu podziałowi na klasy. Przy zbiorze MNIST jedna klasa jest praktycznie pusta, a dla zbioru Human Activity Recognition Using Smartphones widać, że przy mapowaniu niektóre klasy trafiły prawie w całości do dwóch klastrów.

4 Podsumowanie

Na podstawie wyników można wyciągnąć pewne wnioski:

- Z laboratorium KOH1, w obydwóch przypadkach funkcja gaussowska osiągnęła lepsze wyniki.
- Mexican Hat lepiej sobie radzi dla domyślnej szerokości sąsiedztwa $t = 1$, a w przypadku funkcji gaussowskiej szerokość nie miała znaczenia (różnice były nieznaczące).
- Mapowanie zazwyczaj zachodzi lepiej dla liczby neuronów w siatce odpowiadającej rzeczywistej liczbie klas.

- Z laboratorium KOH2, w obydwóch przypadkach siatka prostokątna osiągnęła lepsze wyniki. Jednak nie są to ogromne różnice w wartości v-measure, zatem nie ma wyraźnych podstaw aby mówić, że jest to lepszy wariant.
- Zaskakująco, dla zbioru **HAR** lepiej poradziła sobie funkcja mexican, ale tylko w przypadku siatki prostokątnej.
- Słabe wyniki dla zbioru **MNIST** najprawdopodobniej wynikają ze zbyt małej liczby wykonanych iteracji oraz dużej złożoności danych spowodowanej liczbą wymiarów.

Eksperymenty wykazały, że sieć Kohonena jest skutecznym narzędziem do mapowania prostszych danych i odwzorowywania struktury przestrzennej. Znalezione klastry często pokrywały się z podziałem na oryginalne klasy. Sieć Kohonena była w stanie wykryć położenie skupisk punktów, umieszczając odpowiednie neorony w bliskiej odległości na mapie.

Jednym z ograniczeń sieci Kohonena jest jej zdolność do radzenia sobie z dużymi i złożonymi danymi, co można było zaobserwować na przeprowadzonych doświadczeniach. Stąd, przykładowo, zastosowanie sieci Kohonena na zbiorze **MNIST** nie osiągnęło tak dobrych wyników jak dla zbiorów **Hexagon** i **Cube**. W przypadku zbioru **MNIST**, mimo że klastry odpowiadały klasom cyfr, niektóre cyfry o podobnych kształtach mogły zostać pomieszane lub oddzielone w wyniku skomplikowanej struktury danych. Ponadto, w przypadku zbioru **HAR**, klastry mogły nie zawsze idealnie odzwierciedlać oczekiwany podział na odpowiednie czynności, co sugeruje, że bardziej złożone dane mogą wymagać bardziej zaawansowanych technik.

Sieć Kohonena ma wiele praktycznych zastosowań w dziedzinach takich. Może być wykorzystana do grupowania danych, wyodrębniania istotnych cech czy kompresji danych. Istnieje wiele możliwości ich rozwoju. Można eksperymentować z różnymi funkcjami sąsiedztwa, topologiami siatki i strategiami uczenia, aby dostosować sieć do konkretnych problemów. Ponadto, być może, zastosowanie pewnych bardziej zaawansowanych technik jest w stanie je ulepszyć.

5 Źródła

- `https://pages.mini.pw.edu.pl/~karwowskij/mioad/wyklad2.pdf`
- `https://paperswithcode.com/dataset/har`
- `https://en.wikipedia.org/wiki/MNIST_database`
- `https://www.youtube.com/watch?v=RgxRo7dmPFU`
- `https://www.youtube.com/watch?v=_Q-aV4BgO00`
- `https://www.youtube.com/watch?v=b3nG4c2NECI`
- `https://pl.wikipedia.org/wiki/Sie%C4%87_Kohonena`