Metody Inteligencji Obliczeniowej w Analizie Danych
Sprawozdanie KOH

Filip Langiewicz
nr indeksu 327293

Politechnika Warszawska
Wydział Matematyki i Nauk Informacyjnych
13 maja 2025

Spis treści

1. Wstęp

Celem laboratoriów było zaimplementowanie i przetestowanie działania sieci Kohonena, będącej nienadzorowaną siecią neuronową wykorzystywaną do grupowania i odwzorowywania danych wielowymiarowych. W ramach zadania opracowano sieć Kohonena działającą na prostokątnej oraz sześciokątnej siatce neuronów, z możliwością wykorzystania różnych funkcji sąsiedztwa, takich jak funkcja gaussowska oraz jej minus druga pochodna. Dodatkowo wprowadzono mechanizm stopniowego zmniejszania współczynnika uczenia zgodnie z funkcją wykładniczą.

Zaimplementowaną sieć zastosowano najpierw do analizy prostych zbiorów danych 2D i 3D (`hexagon` i `cube`), aby zweryfikować poprawność działania algorytmu i sposób odwzorowania struktur klas. Następnie przeprowadzono eksperymenty na bardziej złożonych zbiorach, takich jak `MNIST`, w celu zbadania zdolności sieci Kohonena do wyodrębnienia ukrytych struktur klas bez wykorzystania etykiet podczas procesu uczenia. Analiza wyników obejmowała ocenę spójności klastrów, rozkładu danych w neuronach oraz zgodności przyporządkowania z rzeczywistymi etykietami.

2. KOH1: Podstawowa sieć Kohonena

2.1. Opis tematu

Celem zadania było zaimplementowanie podstawowej wersji sieci Kohonena (Self-Organizing Map, SOM), będącej jedną z metod nienadzorowanego uczenia maszynowego. Sieć ta składa się z neuronów ułożonych w regularną siatkę prostokątną o wymiarach $M \times N$, gdzie M i N są parametrami ustalanymi przez użytkownika. Każdy neuron reprezentuje wektor wagowy o tej samej liczbie wymiarów co dane wejściowe.

Proces uczenia polegał na stopniowym dopasowywaniu wag neuronów do struktury danych wejściowych, przy czym zmiany były propagowane także w obszarze sąsiedztwa wyznaczanym wokół najlepiej dopasowanego neuronu (Best Matching Unit, BMU). W zadaniu zaimplementowano dwie funkcje sąsiedztwa: klasyczną funkcję gaussowską oraz funkcję będącą przeciwną do drugiej pochodnej funkcji gaussowskiej. Parametry funkcji sąsiedztwa, takie jak szerokość oddziaływania, były regulowane za pomocą odpowiedniego mnożnika i testowane dla różnych wartości z zakresu $[0.1, 10]$.

Aby proces uczenia sieci stopniowo wygasał w czasie, zastosowano funkcję wygaszania współczynnika uczenia postaci

$$\alpha(t) = e^{-\frac{t}{\lambda}},$$

gdzie t oznacza numer iteracji, a λ jest stałą określającą tempo wygaszania.

Działanie stworzonej sieci zostało przetestowane na dwóch prostych zbiorach danych dostarczonych w ramach kursu: danych 2D skupionych wokół wierzchołków sześciokąta oraz danych 3D skupionych wokół wierzchołków sześcianu. W obu przypadkach interesowało nas, czy wytrenowana sieć potrafi odwzorować strukturę danych poprzez utworzenie klastrów odpowiadających faktycznym klasom oraz czy w pojedynczym neuronie dominują dane należące do jednej klasy. Dodatkowo analizowano rozmieszczenie neuronów w przestrzeni danych oraz liczebność elementów różnych klas przypisanych do poszczególnych neuronów.

2.2. Opis wykonanej pracy

W ramach zadania zaimplementowano własną wersję sieci Kohonena (Self-Organizing Map - SOM) w języku Python. Stworzona klasa `KohonenNetwork` umożliwia budowę dwuwymiarowej siatki neuronów oraz przeprowadzenie procesu trenowania na dostarczonych danych. Główne funkcjonalności sieci obejmują:

- inicjalizację wag neuronów losowymi wartościami,
- wybór funkcji sąsiedztwa (`gaussian` lub `mexican_hat`),
- adaptacyjne aktualizowanie wag neuronów w oparciu o odległość od jednostki najlepiej dopasowanej (BMU - Best Matching Unit),
- dekrementację współczynnika uczenia i promienia sąsiedztwa w trakcie treningu,
- przypisywanie danych wejściowych do neuronów oraz wyznaczanie przynależności klasowej.

Funkcje sąsiedztwa (`gaussian` oraz `mexican_hat`) zaimplementowano jako osobne funkcje, zwracające wartości odpowiednio w zależności od odległości od BMU i parametru σ.

Trening sieci polegał na:

1. permutowaniu zbioru danych,
2. znajdowaniu dla każdej próbki BMU,
3. aktualizacji wag neuronów w oparciu o wartości funkcji sąsiedztwa i odległość od BMU.

Po zakończeniu procesu uczenia każdej konfiguracji, dane zostały przypisane do neuronów. Następnie wyznaczono klasę przypisaną do każdej próbki.

Zaimplementowano również narzędzia do oceny jakości klasteryzacji:

- metryki zewnętrzne: *Adjusted Rand Index (ARI)* oraz *Normalized Mutual Information (NMI)*,
- metryki wewnętrzne: *Calinski-Harabasz Index*, *Davies-Bouldin Index*, *Silhouette Score*.

2.3. Eksperymenty i ich wyniki

Sieć została przetestowana na dwóch zestawach danych, dostarczonych w kursie:

- zbiór `hexagon` (dane 2D) - punkty skupione w wierzchołkach sześciokąta,
- zbiór `cube` (dane 3D) - punkty skupione w wierzchołkach sześcianu.

W ramach eksperymentów przetestowano różne konfiguracje sieci Kohonena, obejmujące zmiany podstawowych parametrów mających wpływ na proces uczenia i wyniki klasteryzacji:

- **Promień sąsiedztwa** σ:
  - Testowane wartości: 0.1, 0.5, 1.0, 10.0,
  - Parametr σ określa szerokość funkcji sąsiedztwa i wpływa na zakres aktualizacji wag neuronów wokół jednostki najlepiej dopasowanej (BMU).
- **Funkcja sąsiedztwa**:
  - `gaussian` (funkcja Gaussa),
  - `mexican_hat` (tzw. "meksykański kapelusz"),
  - Różne funkcje sąsiedztwa modelują sposób, w jaki wpływ rozprzestrzenia się od BMU do sąsiednich neuronów.
- **Rozmiary siatki neuronów (M, N)**:
  - 2×2 (4 neurony),
  - 2×3 (6 neuronów) - dla danych `hexagon`,
  - 2×4 (6 neuronów) - dla danych `cube`,
  - 3×3 (9 neuronów),
  - 10×10 (100 neuronów),
  - Rozmiar siatki wpływa na liczbę dostępnych klastrów i rozdzielczość odwzorowania przestrzeni danych.

Oprócz tego ustawiono stałe wartości hiperparametrów: liczbę epok na 10, współczynnik uczenia `learning_rate` na 0.1, zapewniający umiarkowane tempo zmian wag, a `decay_lambda` ustawiono na 1000, co kontrolowało tempo zmniejszania promienia sąsiedztwa. Włączono także `sigma_decay`, czyli automatyczne zmniejszanie promienia w czasie.

Każda konfiguracja została przetestowana dla wszystkich kombinacji powyższych parametrów, co umożliwiło kompleksową ocenę wpływu poszczególnych czynników na jakość klasteryzacji.

Dla każdej konfiguracji:

1. przeprowadzono proces trenowania,

2. dokonano przypisania klas do neuronów,
3. obliczono liczbę wykrytych klastrów,
4. wyznaczono wszystkie wskaźniki jakości klasteryzacji,
5. wygenerowano wykresy:
    - porównanie danych z prawdziwymi etykietami i etykietami przypisanymi przez SOM,
    - słupkowe zestawienia liczby próbek przypisanych do neuronów według klas rzeczywistych i predykcji.

Wyniki eksperymentów dla poszczególnych zbiorów zostały przedstawione w tabeli 2.1 i 2.2, gdzie w kolumnie Funkcja g oznacza funkcję sąsiedztwa `gaussian`, zaś m funkcję sąsiedztwa `mexican_hat`. Kolumna c-h oznacza wskaźnik calinski-harabasz, zaś d-b oznacza wskaźnik davies-bouldin. Tabele znajdują się na końcu rozdziału.

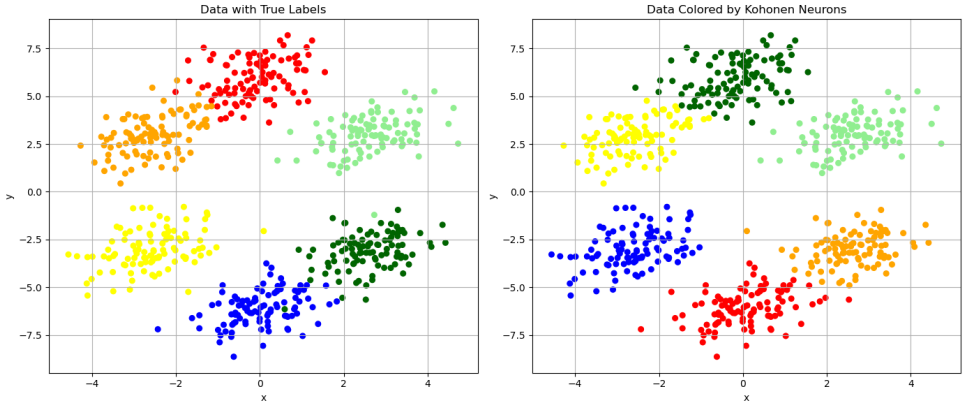
Najlepsze i najgorsze konfiguracje hiperparametrów sieci zostały dla poszczególnych zbiorów zostały opisane poniżej.

Dla zbioru `hexagon` najlepsza konfiguracja to: **funkcja: `gaussian`, M=2, N=3, sigma=0.1/0.5/1.0**. Wykryta liczba klastrów jest równa prawdziwej liczbie. Wysoki wskaźnik Calinskiego-Harabasza (1841.66), stosunkowo niski Davies-Bouldin (0.54). Dobre wyniki silhouette (0.60) oraz wysokie wartości ARI (0.94) i NMI (0.94). Najgorsza konfiguracja to natomiast: **funkcja: `mexican_hat`, sigma=1.0/10.0 dla różnych rozmiarów siatki**. Liczba klastrów spada do 1 lub 2. Pojawiają się wartości NaN w metrykach (Calinski-Harabasz, Davies-Bouldin, silhouette). Wartości ARI i NMI wynoszą 0.00, co wskazuje na całkowite niepowodzenie w klasteryzacji.

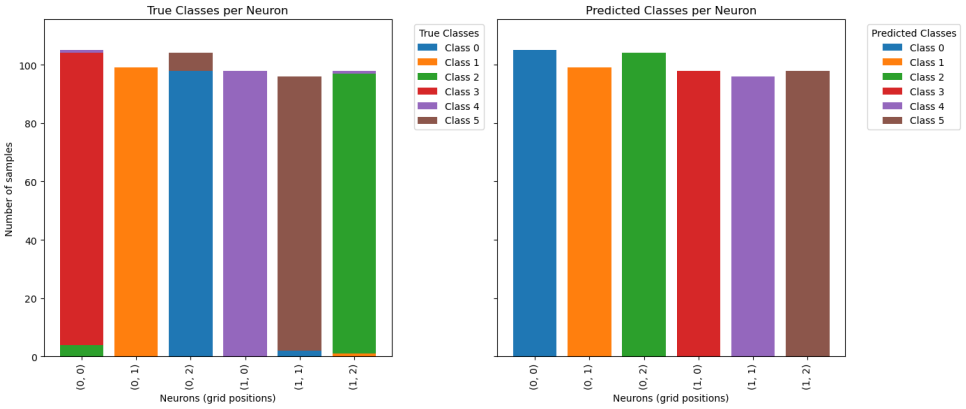
Wykresy 2.1 - 2.8 przedstawiają znaleziony podział na klastry w porównaniu do prawdziwych klas dla najlepszych oraz najgorszej konfiguracji hiperparametrów.

Dla zbioru `cube` najlepsze konfiguracje to: **funkcja: `gaussian`, M=N=3, sigma=0.5**. Wykryta liczba klastrów (9) jest zbliżona do rzeczywistej. Osiągnięto wysokie wartości wskaźnika Calinskiego-Harabasza (1161.43) oraz niskie Davies-Bouldin (1.02). Wynik silhouette był dobry (0.39), a wartość ARI (0.89) oraz NMI (0.90) wskazują na wysoką zgodność z prawdziwym podziałem. Najgorsza konfiguracja obejmuje: **funkcja: `mexican_hat`, sigma=1.0 dla małych siatek (np. 2×2, 2×4, 3×3)**. Skutkowała ona występowaniem wartości NaN dla metryk (Calinski-Harabasz, Davies-Bouldin, silhouette), a wartości ARI i NMI wynosiły 0.00, wskazując na całkowite niepowodzenie w klasteryzacji.

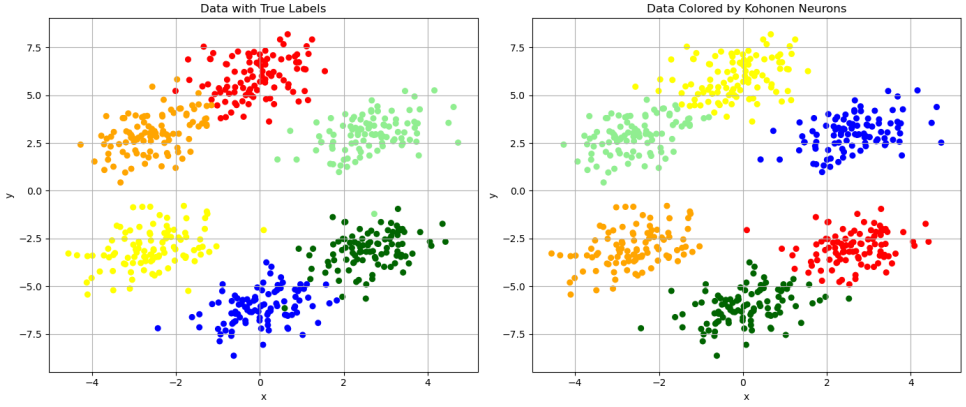
Wykresy 2.9 - 2.12 przedstawiają znaleziony podział na klastry w porównaniu do prawdziwych klas dla najlepszych oraz najgorszych konfiguracji hiperparametrów.



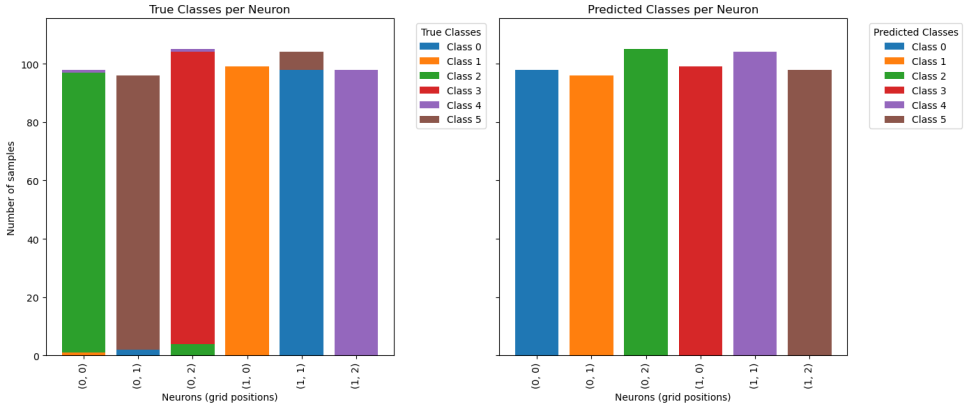
Rysunek 2.1: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: funkcja `gaussian`, M=2, N=3, sigma=0.1.



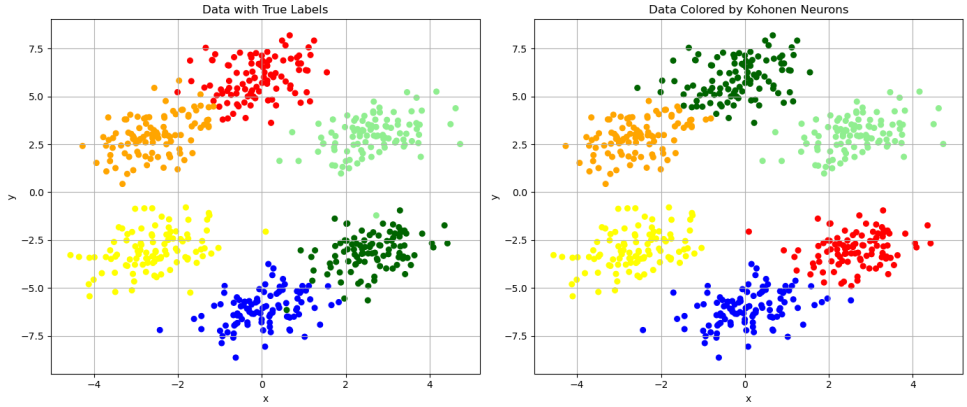
Rysunek 2.2: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: funkcja `gaussian`, M=2, N=3, sigma=0.1.



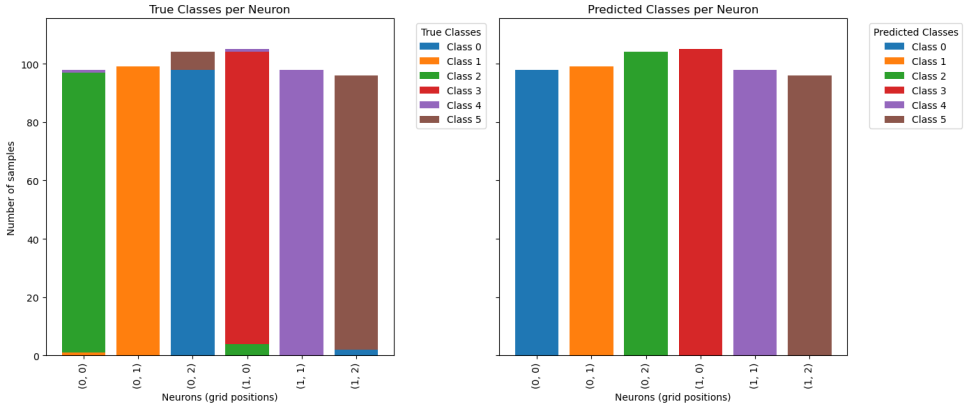
Rysunek 2.3: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: funkcja `gaussian`, M=2, N=3, sigma=0.5.



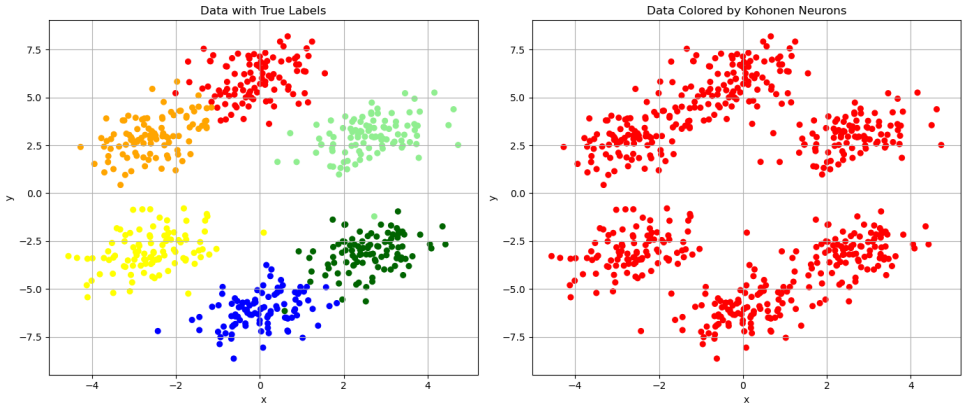
Rysunek 2.4: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: funkcja `gaussian`, M=2, N=3, sigma=0.5.



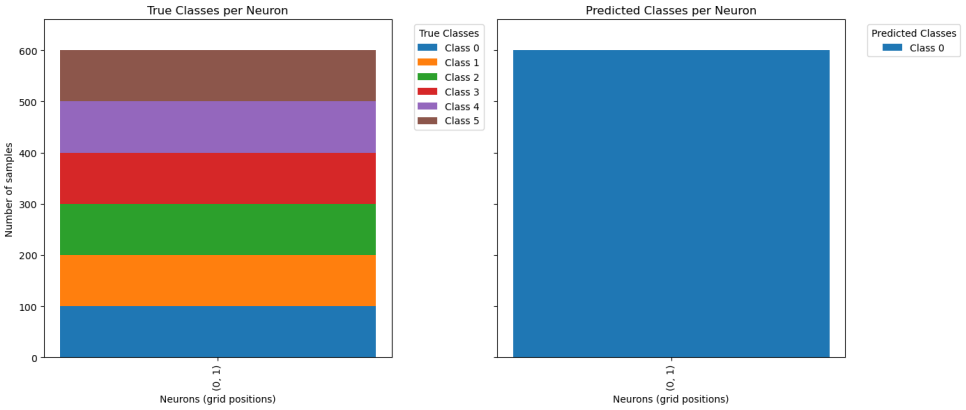
Rysunek 2.5: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: funkcja `gaussian`, M=2, N=3, sigma=1.



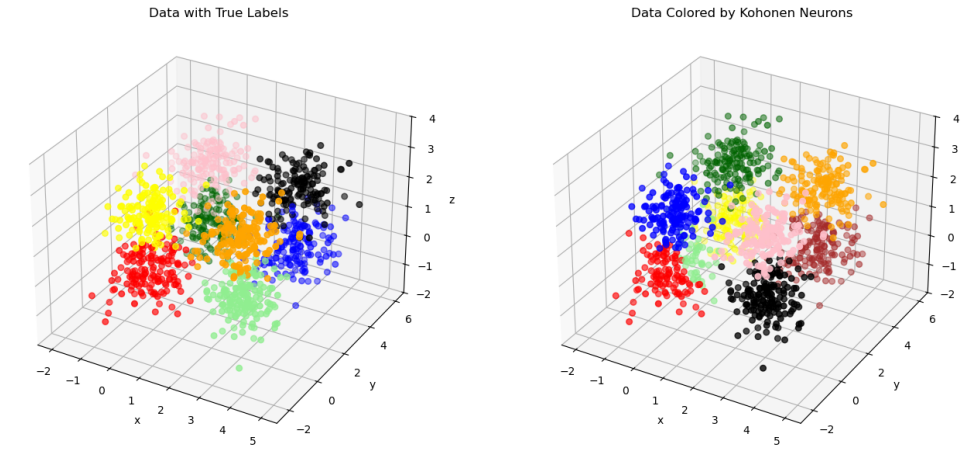
Rysunek 2.6: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: funkcja `gaussian`, M=2, N=3, sigma=1.



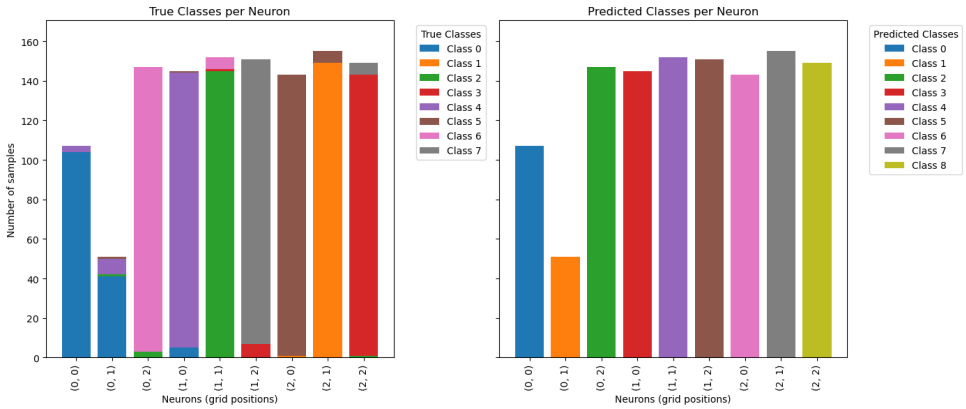
Rysunek 2.7: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: funkcja `mexican_hat`, M=2, N=3, sigma=1.



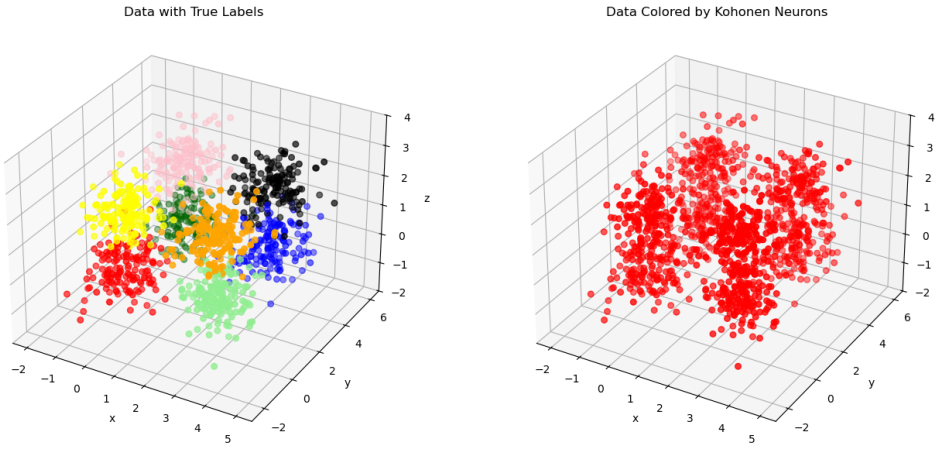
Rysunek 2.8: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: funkcja `mexican_hat`, M=2, N=3, sigma=1.



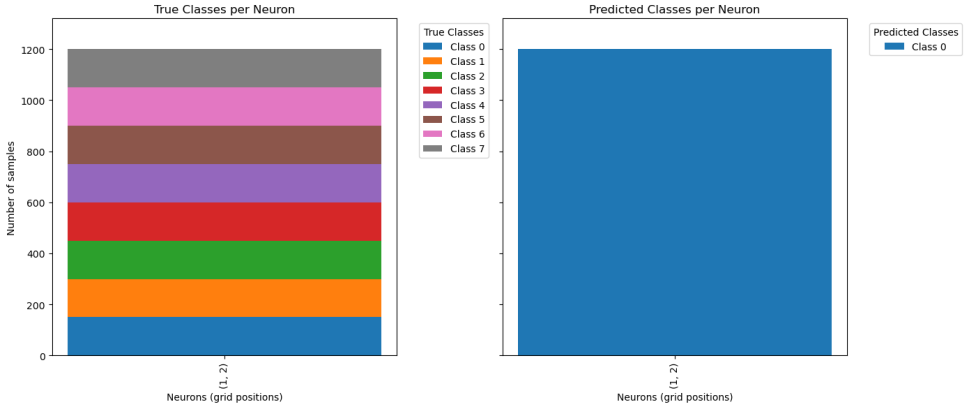
Rysunek 2.9: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: funkcja `gaussian`, M=3, N=3, sigma=0.5.



Rysunek 2.10: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: funkcja `gaussian`, M=3, N=3, sigma=0.5.



Rysunek 2.11: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: funkcja `mexican_hat`, M=2, N=4, sigma=1.



Rysunek 2.12: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: funkcja `mexican_hat`, M=2, N=4, sigma=1.

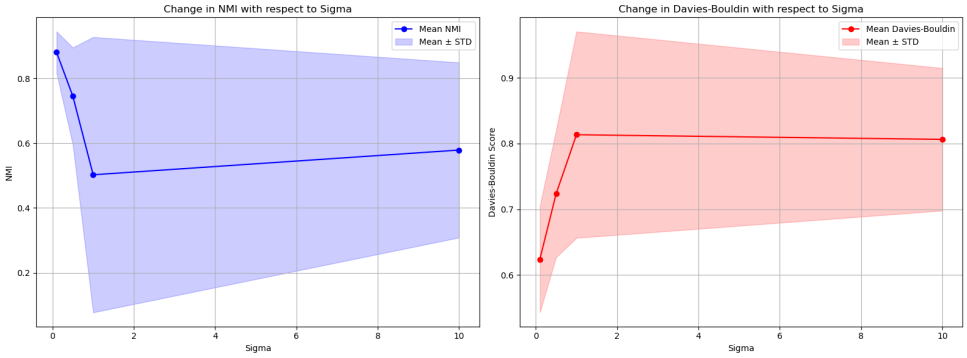
Funkcja	**M**	**N**	**sigma**	**Liczba klastrów**	**c-h**	**d-b**	**silhouette**	**ari**	**nmi**
g	2	2	0.1	4	985.13	0.76	0.49	0.67	0.82
g	2	2	0.5	4	985.13	0.76	0.49	0.67	0.82
g	2	2	1.0	4	953.08	0.82	0.45	0.65	0.78
g	2	2	10.0	4	582.75	0.87	0.29	0.32	0.52
g	2	3	0.1	6	1841.66	0.54	0.60	0.94	0.94
g	2	3	0.5	6	1841.66	0.54	0.60	0.94	0.94
g	2	3	1.0	6	1841.66	0.54	0.60	0.94	0.94
g	2	3	10.0	6	1047.89	0.74	0.44	0.73	0.81
g	3	3	0.1	9	1285.46	0.71	0.47	0.89	0.90
g	3	3	0.5	9	1296.15	0.76	0.49	0.88	0.90
g	3	3	1.0	9	1263.00	0.92	0.52	0.90	0.90
g	3	3	10.0	9	1239.68	0.91	0.51	0.90	0.90
g	10	10	0.1	12	937.00	0.60	0.42	0.89	0.90
g	10	10	0.5	14	992.50	0.75	0.34	0.78	0.83
g	10	10	1.0	45	909.35	0.88	0.25	0.26	0.65
g	10	10	10.0	82	592.29	0.90	0.16	0.19	0.60
m	2	2	0.1	4	809.76	0.56	0.52	0.54	0.75
m	2	2	0.5	2	1203.33	0.67	0.58	0.33	0.55
m	2	2	1.0	1	NaN	NaN	NaN	0.00	0.00
m	2	2	10.0	2	1203.33	0.67	0.58	0.33	0.55
m	2	3	0.1	6	1841.66	0.54	0.60	0.94	0.94
m	2	3	0.5	2	1203.33	0.67	0.58	0.33	0.55
m	2	3	1.0	1	NaN	NaN	NaN	0.00	0.00
m	2	3	10.0	2	1203.33	0.67	0.58	0.33	0.55
m	3	3	0.1	9	1256.09	0.64	0.48	0.89	0.91
m	3	3	0.5	3	879.79	0.77	0.48	0.45	0.66
m	3	3	1.0	1	NaN	NaN	NaN	0.00	0.00
m	3	3	10.0	1	NaN	NaN	NaN	0.00	0.00
m	10	10	0.1	12	945.01	0.64	0.42	0.88	0.89
m	10	10	0.5	28	952.84	0.87	0.30	0.44	0.71
m	10	10	1.0	25	843.45	0.90	0.27	0.57	0.76
m	10	10	10.0	3	747.82	0.88	0.45	0.52	0.69

Tabela 2.1: Wyniki eksperymentów dla zbioru `hexagon`.

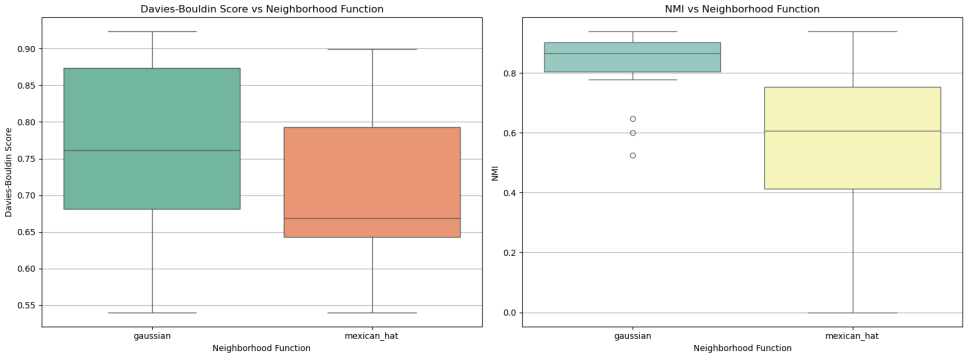
Funkcja	**M**	**N**	**sigma**	**Liczba klastrów**	**c-h**	**d-b**	**silhouette**	**ari**	**nmi**
g	2	2	0.1	4	780.37	0.85	0.40	0.41	0.70
g	2	2	0.5	4	1278.63	0.88	0.47	0.59	0.79
g	2	2	1.0	4	1278.63	0.88	0.47	0.59	0.79
g	2	2	10.0	4	442.67	1.38	0.08	0.24	0.45
g	2	4	0.1	8	833.56	1.13	0.39	0.68	0.83
g	2	4	0.5	8	1030.66	1.01	0.40	0.78	0.88
g	2	4	1.0	8	826.70	1.23	0.29	0.62	0.76
g	2	4	10.0	8	629.75	1.90	0.18	0.53	0.71
g	3	3	0.1	9	688.10	1.19	0.37	0.61	0.77
g	3	3	0.5	9	1161.43	1.02	0.39	0.89	0.90
g	3	3	1.0	9	787.36	1.15	0.29	0.68	0.79
g	3	3	10.0	9	528.49	1.41	0.18	0.54	0.70
g	10	10	0.1	49	134.65	0.85	0.19	0.60	0.74
g	10	10	0.5	55	219.70	0.91	0.21	0.71	0.77
g	10	10	1.0	74	284.55	1.07	0.14	0.32	0.67
g	10	10	10.0	95	208.90	1.58	0.01	0.32	0.65
m	2	2	0.1	4	780.36	0.85	0.40	0.41	0.70
m	2	2	0.5	2	1160.41	0.98	0.45	0.25	0.50
m	2	2	1.0	1	NaN	NaN	NaN	0.00	0.00
m	2	2	10.0	2	443.50	1.60	0.29	0.24	0.48
m	2	4	0.1	8	417.28	1.14	0.28	0.41	0.69
m	2	4	0.5	4	780.37	0.85	0.40	0.41	0.70
m	2	4	1.0	1	NaN	NaN	NaN	0.00	0.00
m	2	4	10.0	4	1278.63	0.88	0.47	0.59	0.79
m	3	3	0.1	9	746.41	1.11	0.39	0.65	0.81
m	3	3	0.5	3	1026.95	0.83	0.44	0.38	0.65
m	3	3	1.0	1	NaN	NaN	NaN	0.00	0.00
m	3	3	10.0	1	NaN	NaN	NaN	0.00	0.00
m	10	10	0.1	45	145.73	0.99	0.22	0.60	0.74
m	10	10	0.5	48	237.14	0.96	0.25	0.78	0.80
m	10	10	1.0	37	325.52	1.07	0.25	0.73	0.79
m	10	10	10.0	3	361.59	1.07	0.22	0.16	0.35

Tabela 2.2: Wyniki eksperymentów dla zbioru `cube`.

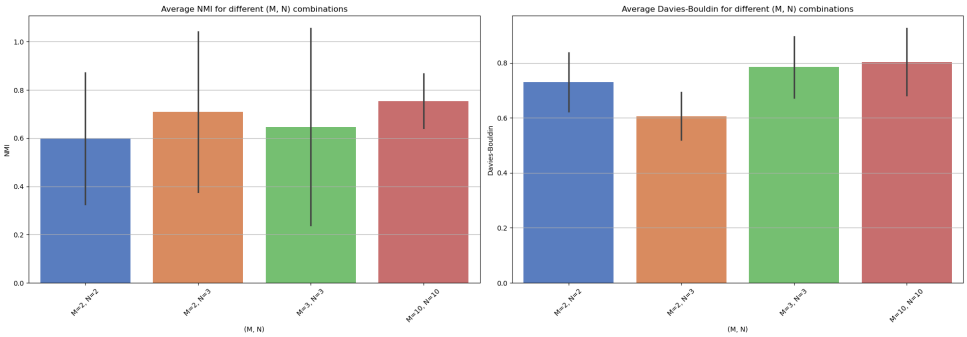
Oprócz standardowych eksperymentów porównano również zmiany metryk przy zmianie hiperparametrów. Otrzymane wyniki przedstawione są na wykresach 2.13 - 2.15 dla zbioru `hexagon` oraz 2.16 - 2.18 dla zbioru `cube`.



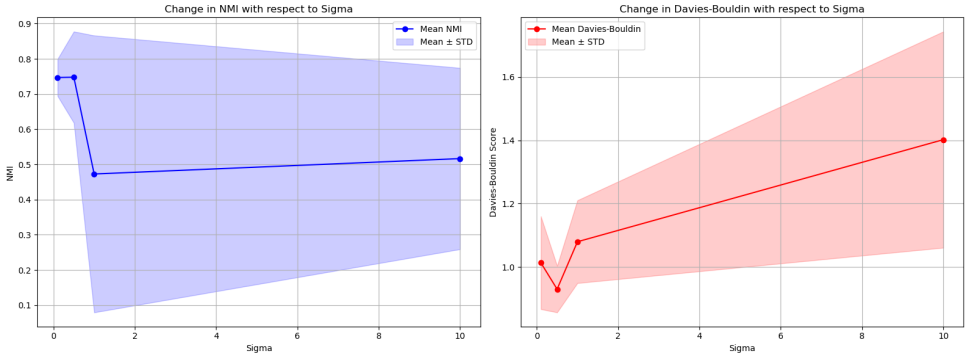
Rysunek 2.13: Porównanie wyników dla różnych wartości parametru sigma.



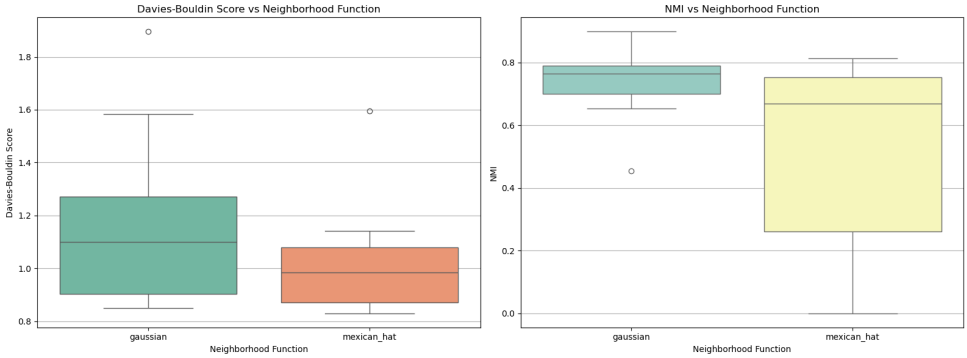
Rysunek 2.14: Porównanie wyników dla różnych funkcji sąsiedztwa.



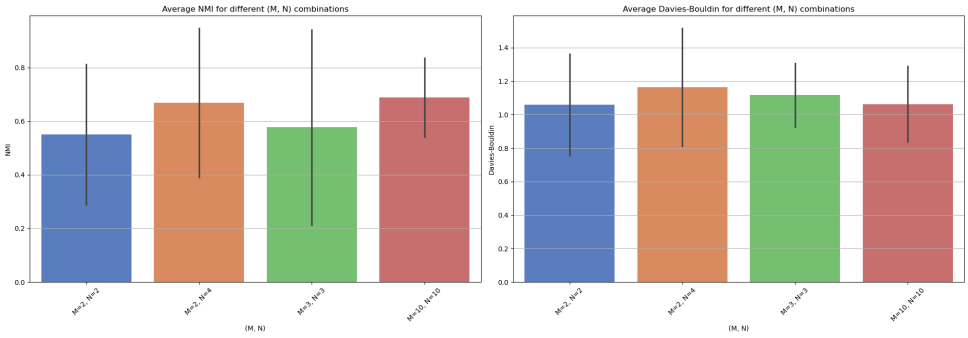
Rysunek 2.15: Porównanie wyników dla różnych siatek neuronów.



Rysunek 2.16: Porównanie wyników dla różnych wartości parametru sigma.



Rysunek 2.17: Porównanie wyników dla różnych funkcji sąsiedztwa.



Rysunek 2.18: Porównanie wyników dla różnych siatek neuronów.

2.4. Wnioski

W przeprowadzonych eksperymentach sieć Kohonena skutecznie klasteryzowała dane w dwóch różnych zbiorach (`hexagon` i `cube`), jednak wyniki wskazują na pewne rozbieżności między liczbą wykrytych klastrów a rzeczywistymi klasami w danych.

1. **Czy klastry w odwzorowaniu znalezionym przez sieć pokrywają się w liczbą klastrów w faktycznych danych?**

   W wielu przypadkach liczba wykrytych klastrów przez sieć pokrywała się z liczbą klas w danych, jednak nie zawsze. Dla mniejszych rozmiarów siatek neuronowych (np. 2×2) sieć często wykrywała tylko część rzeczywistych klas, co może być wynikiem zbyt małej liczby neuronów do reprezentacji wszystkich klas. Większe siatki neuronowe (np. 3×3) wykazywały lepszą zdolność do odwzorowywania liczby klas, ale pojawiał się również problem nadmiernej fragmentacji (np. przy siatkach 10×10).

2. **Czy w obrębie pojedynczego neuronu znajdują się dane z tylko jednej klasy?**

   W obrębie pojedynczego neuronu zazwyczaj znajdują się dane z różnych klas, co jest charakterystyczne dla sieci Kohonena, gdzie każdy neuron może przyciągać próbki z różnych obszarów przestrzeni danych. Jednak przy odpowiednio dobranych parametrach (np. mały promień sąsiedztwa) neuron może skupić dane z jednej klasy, ale jest to zależne od rozmiaru siatki i funkcji sąsiedztwa.

3. **Jakie są pozycje neuronów w danych?**

   Neurony w siatce Kohonena reprezentują różne regiony przestrzeni danych. W przypadku zbioru `hexagon` neurony są rozmieszczone w regularnym wzorze, co wpływa na sposób grupowania danych w sąsiednich neuronach. Dla zbioru `cube` rozkład neuronów w przestrzeni 3D jest bardziej zróżnicowany, co może skutkować różnym rozmieszczeniem danych w przestrzeni.

4. **Jakie są liczności elementów z poszczególnych z klas w pojedynczym neuronie?**

   Liczności elementów z poszczególnych klas w pojedynczym neuronie zależą od parametrów sieci oraz od tego, jak dobrze sieć udało się wyodrębnić klastry. W przypadku mniejszych siatek, takich jak 2×2, neurony często muszą reprezentować dane z wielu różnych klas, co może prowadzić do nierównych liczności w obrębie neuronów. W przypadku większych siatek neuronów (np. 10×10), liczność elementów z poszczególnych klas w neuronach może

być bardziej jednorodna, ponieważ większa liczba neuronów pozwala na bardziej precyzyjne przypisanie danych do pojedynczych jednostek.

Podsumowując, sieć Kohonena skutecznie identyfikuje grupy w danych, jednak wyniki mogą się różnić w zależności od ustawień parametrów takich jak rozmiar siatki, funkcja sąsiedztwa oraz promień sąsiedztwa. Optymalizacja tych parametrów jest kluczowa dla uzyskania lepszych rezultatów w klasteryzacji, szczególnie w przypadkach wymagających precyzyjnego odwzorowania liczby klas. Na podstawie dodatkowej analizy można stwierdzić, że wybór małego parametru sigma może dać lepsze wyniki. Lepsze wyniki uzyskano również przy wyborze funkcji sąsiedztwa `gaussian`, co pozwala stwierdzić, że może być ona lepszym wyborem. Należy pamiętać, że wielkości hiperparametrów powinny być dobrane pod konkretne zadanie.

3. KOH2: Sieć Kohonena na siatce sześciokątnej

3.1. Opis tematu

Celem zadania było rozszerzenie implementacji sieci Kohonena o możliwość rozmieszczenia neuronów w topologii siatki sześciokątnej. Następnie należało przeprowadzić eksperymenty z wykorzystaniem obu wariantów topologii (prostokątnej i sześciokątnej) oraz dwóch funkcji sąsiedztwa (`gaussian` i `mexican_hat`).

Testy przeprowadzono na wektorach danych pochodzących ze zbiorów:

- MNIST (obrazy cyfr ręcznie pisanych),
- Human Activity Recognition Using Smartphones (dane akcelerometryczne dotyczące aktywności człowieka).

W sieci Kohonena wykorzystano dane bez etykiet, a następnie, na potrzeby analizy wyników, odniesiono uzyskane mapowania do rzeczywistych etykiet klas, aby ocenić jakość znalezionych klastrów.

3.2. Opis wykonanej pracy

W ramach zadania zaimplementowano rozszerzoną wersję sieci Kohonena (Self-Organizing Map – SOM) umożliwiającą wybór topologii siatki neuronów: prostokątnej (rectangular) lub sześciokątnej (hexagon). Rozszerzenie to pozwala na bardziej elastyczne dostosowanie struktury sieci do różnych typów danych i analizowania wpływu układu neuronów na jakość klasteryzacji.

Wprowadzono następujące zmiany i funkcjonalności dla wyboru topologii siatki. W konstruktorze klasy KohonenNetwork (`__init__`) wprowadzono parametr topology, który przyjmuje wartości 'rectangular' (domyślnie) lub 'hexagon'.

Dla topologii prostokątnej neurony są rozmieszczone w regularnej siatce, w której każdy neuron ma współrzędne (i, j).

Dla topologii sześciokątnej neurony w co drugiej linii są przesunięte o 0.5 jednostki w poziomie, co odwzorowuje układ heksagonalny — zapisano to poprzez dodanie $0.5 * (i \bmod 2)$ do współrzędnej poziomej neuronów.

Implementacja uwzględnia automatyczną generację współrzędnych neuronów w zależności od wybranej topologii, co umożliwia ich odpowiednie rozmieszczenie na siatce.

Odległości między neuronami obliczane są zgodnie z rzeczywistym układem współrzędnych (prostokątnym lub sześciokątnym), co poprawnie odzwierciedla wpływ sąsiedztwa w obu przypadkach.

Dzięki wprowadzeniu wyboru topologii oraz odpowiedniemu modelowaniu odległości neuronów w przestrzeni, sieć Kohonena może lepiej odwzorować naturalną strukturę danych i tworzyć bardziej spójne klastry w zależności od typu analizowanych zbiorów.

3.3. Eksperymenty i ich wyniki

Sieć Kohonena została przetestowana na dwóch rzeczywistych zbiorach danych:

- `MNIST` – zbiór obrazów cyfr (dane 784-wymiarowe),
- `Activity Recognition` – zbiór sygnałów akcelerometru z przypisanymi aktywnościami (dane 561-wymiarowe).

W ramach eksperymentów przebadano wpływ podstawowych parametrów sieci Kohonena na proces uczenia i jakość klasteryzacji:

- **Promień sąsiedztwa** σ:
  - Testowane wartości: 0.1, 0.5, 1.0, 10.0,
  - Parametr ten kontroluje szerokość funkcji sąsiedztwa, determinując zakres aktualizacji wag neuronów wokół najlepszego dopasowania (BMU).
- **Funkcja sąsiedztwa**:
  - `gaussian` (funkcja Gaussa),
  - `mexican_hat` ("meksykański kapelusz"),
  - Funkcja sąsiedztwa określa, jak wpływ rozprzestrzenia się od BMU do sąsiednich neuronów.
- **Topologia siatki**:

  - `rectangular` (prostokątna),
  - `hexagon` (sześciokątna),
  - Topologia definiuje sposób połączeń pomiędzy neuronami.

- **Rozmiary siatki neuronów (M, N)**:
  - 2×5 (10 neuronów) dla zbioru `MNIST`,
  - 2×3 (6 neuronów) dla zbioru `Activity Recognition`.

Oprócz tego ustawiono stałe wartości hiperparametrów: liczbę epok na 20, współczynnik uczenia `learning_rate` na 0.1, zapewniający umiarkowane tempo zmian wag, a `decay_lambda` ustawiono na 1000, co kontrolowało tempo zmniejszania promienia sąsiedztwa. Włączono także `sigma_decay`, czyli automatyczne zmniejszanie promienia w czasie.

Każda kombinacja parametrów została przetestowana, a dla każdej konfiguracji:

1. przeprowadzono proces trenowania sieci,
2. przypisano klasy do neuronów,
3. określono liczbę wykrytych klastrów,
4. wyznaczono wszystkie wskaźniki jakości klasteryzacji,
5. wygenerowano wykresy:
   - skumulowane wykresy słupkowe liczby próbek na neuron według rzeczywistych i przewidzianych klas,
   - mapę neuronów SOM z oznaczeniem rzeczywistych i przewidzianych klas.

Wyniki eksperymentów zostały przedstawione w tabelach 3.1 i 3.2. W tabelach funkcja g oznacza `gaussian`, a m oznacza `mexican_hat`; c-h oznacza wskaźnik Calinskiego-Harabasza, a d-b oznacza wskaźnik Daviesa-Bouldina. Topologia rect oznacza siatkę prostokątną, zaś hex siatkę sześciokątną.

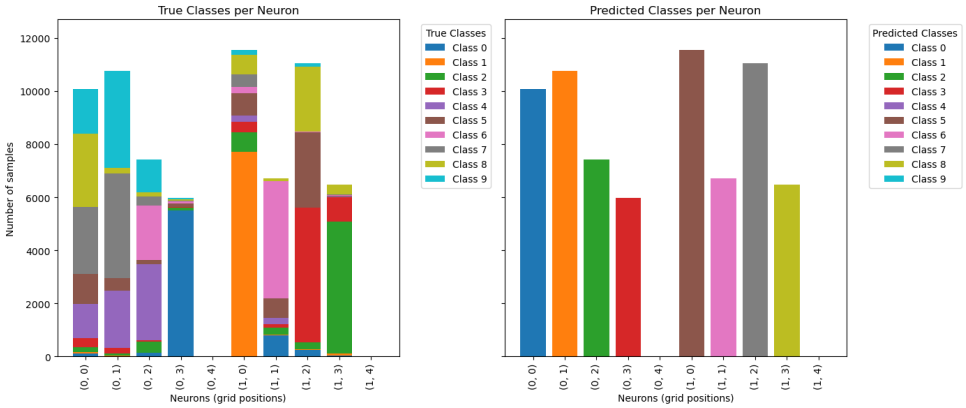
Funkcja	Topologia	sigma	Liczba klastrów	c-h	d-b	silhouette	ari	nmi
g	rect	0.10	1	NaN	NaN	NaN	0.00	0.00
g	rect	0.50	3	4302.68	3.37	0.06	0.17	0.29
g	rect	1.00	10	2276.50	2.69	0.06	0.36	0.47
g	rect	10.00	10	2345.43	3.47	0.05	0.37	0.46
g	hex	0.10	1	NaN	NaN	NaN	0.00	0.00
g	hex	0.50	2	4758.67	3.75	0.09	0.07	0.15
g	hex	1.00	7	2783.41	3.30	0.06	0.28	0.38
g	hex	10.00	10	2092.24	3.79	0.02	0.26	0.37
m	rect	0.10	1	NaN	NaN	NaN	0.00	0.00
m	rect	0.50	1	NaN	NaN	NaN	0.00	0.00
m	rect	1.00	1	NaN	NaN	NaN	0.00	0.00
m	rect	10.00	2	68.46	1.92	0.17	0.00	0.00
m	hex	0.10	1	NaN	NaN	NaN	0.00	0.00
m	hex	0.50	1	NaN	NaN	NaN	0.00	0.00
m	hex	1.00	1	NaN	NaN	NaN	0.00	0.00
m	hex	10.00	2	2113.77	1.97	0.15	0.00	0.06

Tabela 3.1: Wyniki eksperymentów dla zbioru danych `MNIST`.

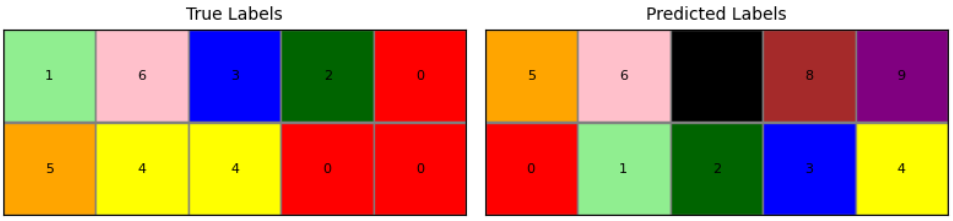
Najlepsze i najgorsze konfiguracje hiperparametrów sieci zostały dla poszczególnych zbiorów opisane poniżej.

Dla zbioru `MNIST` najlepsza konfiguracja to: **funkcja: `gaussian`, topologia: `rect`, sigma=1.0/10.0**. W obu przypadkach wykryto 10 klastrów, co odpowiada rzeczywistej liczbie klas. Wysokie wartości wskaźnika Calinskiego-Harabasza (2276.50 oraz 2345.43) oraz rozsądne wartości wskaźnika Daviesa-Bouldina (odpowiednio 2.69 oraz 3.47) świadczą o dobrej jakości klasteryzacji. Wartości silhouette (0.06 i 0.05) są niskie, ale wartości ARI (0.36 i 0.37) oraz NMI (0.47 i 0.46) wskazują na umiarkowaną zgodność z rzeczywistym podziałem danych. Najgorsza konfiguracja to natomiast: **funkcja: `mexican_hat`, dowolna topologia, sigma=0.1/0.5/1.0**. We wszystkich przypadkach liczba klastrów spada do 1, a metryki (Calinski-Harabasz, Davies-Bouldin, silhouette) przyjmują wartości NaN, co świadczy o całkowitym braku klasteryzacji. Wartości ARI i NMI wynoszą 0.00.

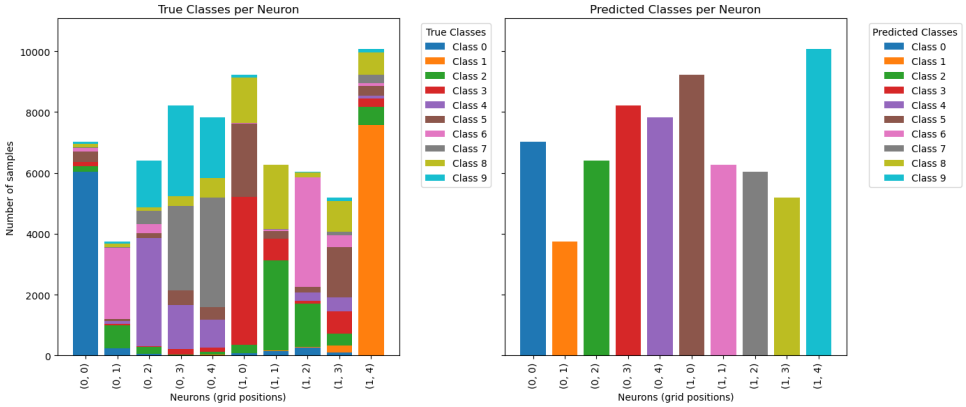
Wykresy 3.1 - 3.6 przedstawiają znaleziony podział na klastry w porównaniu do prawdziwych klas dla najlepszych oraz najgorszych konfiguracji hiperparametrów.



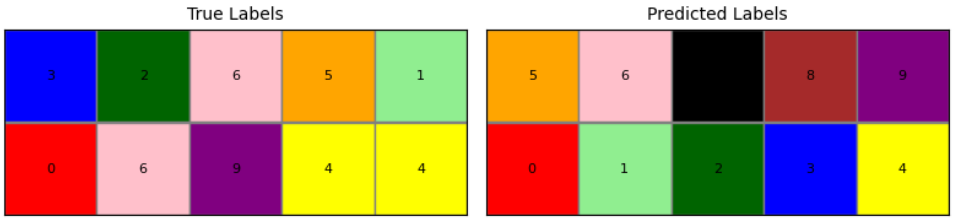
Rysunek 3.1: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: topologia rectangle, funkcja `gaussian`, topologia rect, sigma=1.0.



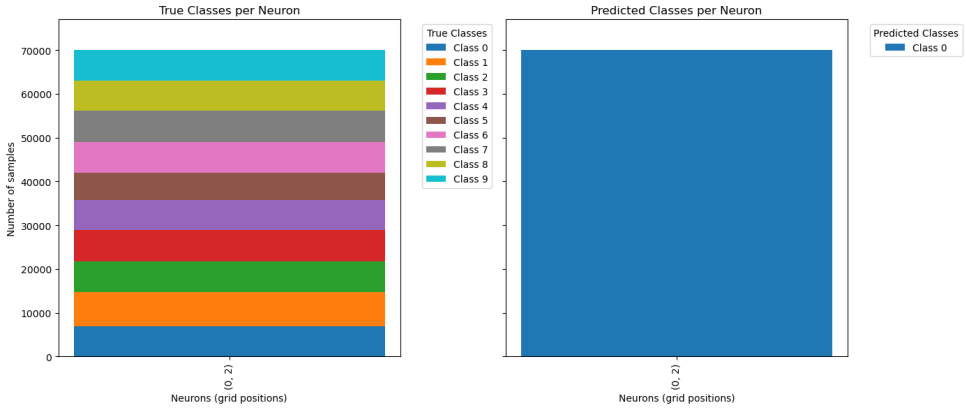
Rysunek 3.2: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: topologia rectangle, funkcja `gaussian`, topologia rect, sigma=1.0.



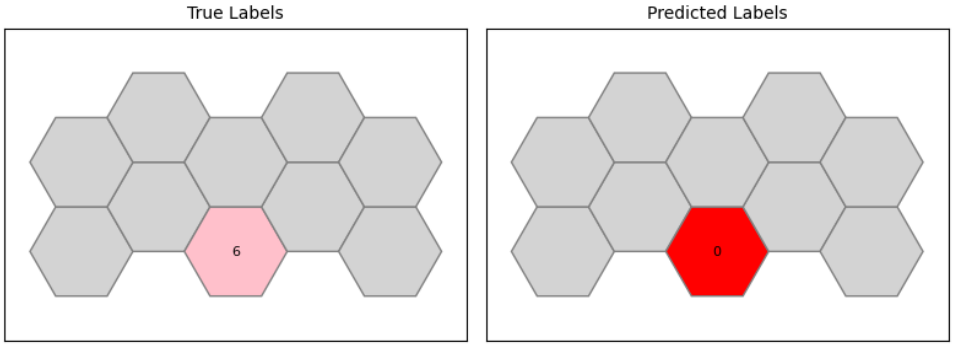
Rysunek 3.3: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: topologia rectangle, funkcja `gaussian`, sigma=10.



Rysunek 3.4: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: topologia rectangle, funkcja `gaussian`, sigma=10.



Rysunek 3.5: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: topologia hexagon, funkcja `mexican_hat`, sigma=1.0.



Rysunek 3.6: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: topologia hexagon, funkcja `mexican_hat`, sigma=1.0.

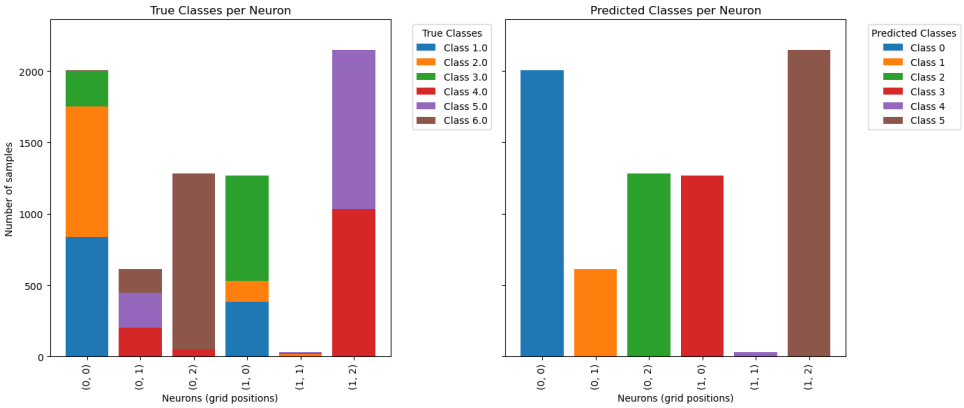
Dla zbioru `activity` najlepsza konfiguracja to: **funkcja: `gaussian`, topologia: `rect`, sigma=1.0**. Otrzymano 6 klastrów, co jest rozsądnym przybliżeniem rzeczywistej liczby. Wysoka wartość Calinskiego-Harabasza (2932.05), umiarkowany Davies-Bouldin (2.14) oraz wartość silhouette (0.16) wskazują na umiarkowanie dobre wyniki. Wysoka wartość ARI (0.48) oraz NMI (0.61) świadczą o dobrej jakości klasteryzacji. Najgorsza konfiguracja to: **funkcja: `mexican_hat`,**

topologia: dowolna, sigma=1.0. W tym przypadku liczba klastrów wynosi 1, a wszystkie metryki jakościowe (Calinski-Harabasza, Davies-Bouldin, silhouette) są NaN, przy wartościach ARI i NMI równych 0.00, co oznacza całkowite niepowodzenie w klasteryzacji.

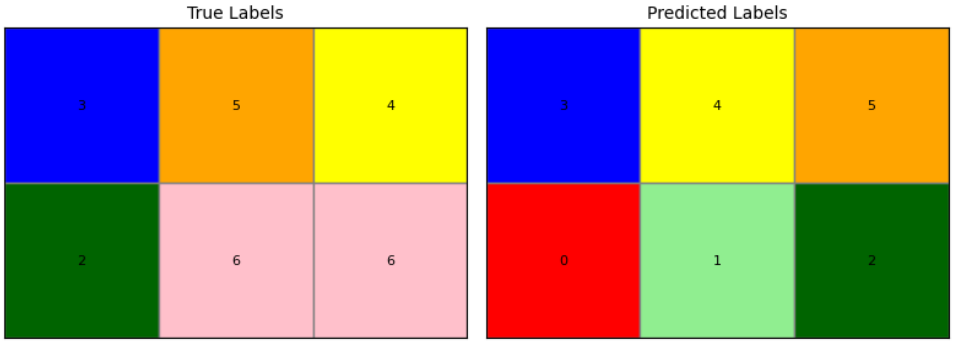
Funkcja	**Topologia**	**sigma**	**Liczba klastrów**	**c-h**	**d-b**	**silhouette**	**ari**	**nmi**
g	rect	0.10	2	220.77	0.87	0.41	0.00	0.01
g	rect	0.50	4	3888.40	1.68	0.29	0.33	0.53
g	rect	1.00	6	2932.05	2.14	0.16	0.48	0.61
g	rect	10.00	6	2671.71	2.46	0.09	0.31	0.47
g	hex	0.10	2	262.18	0.90	0.39	0.00	0.02
g	hex	0.50	3	5315.72	1.13	0.44	0.33	0.54
g	hex	1.00	6	2804.69	2.88	0.10	0.31	0.47
g	hex	10.00	6	2605.37	2.76	0.11	0.30	0.47
m	rect	0.10	2	125.31	0.76	0.46	0.00	0.01
m	rect	0.50	2	137.41	0.78	0.45	0.00	0.01
m	rect	1.00	1	NaN	NaN	NaN	0.00	0.00
m	rect	10.00	2	9817.65	0.84	0.49	0.33	0.55
m	hex	0.10	3	40.26	2.50	0.49	0.00	0.00
m	hex	0.50	2	64.04	0.67	0.51	0.00	0.00
m	hex	1.00	1	NaN	NaN	NaN	0.00	0.00
m	hex	10.00	2	270.26	0.90	0.38	0.00	0.02

Tabela 3.2: Wyniki eksperymentów dla zbioru danych `activity`.

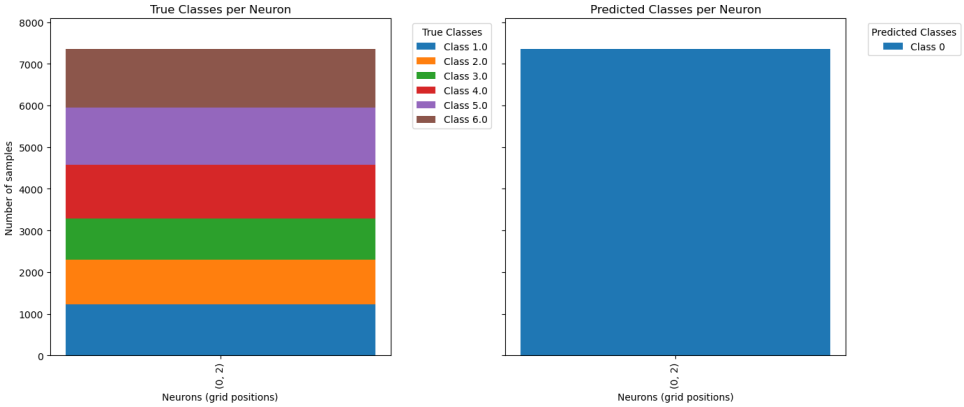
Wykresy 3.7 - 3.10 przedstawiają znaleziony podział na klastry w porównaniu do prawdziwych klas dla najlepszych oraz najgorszych konfiguracji hiperparametrów.



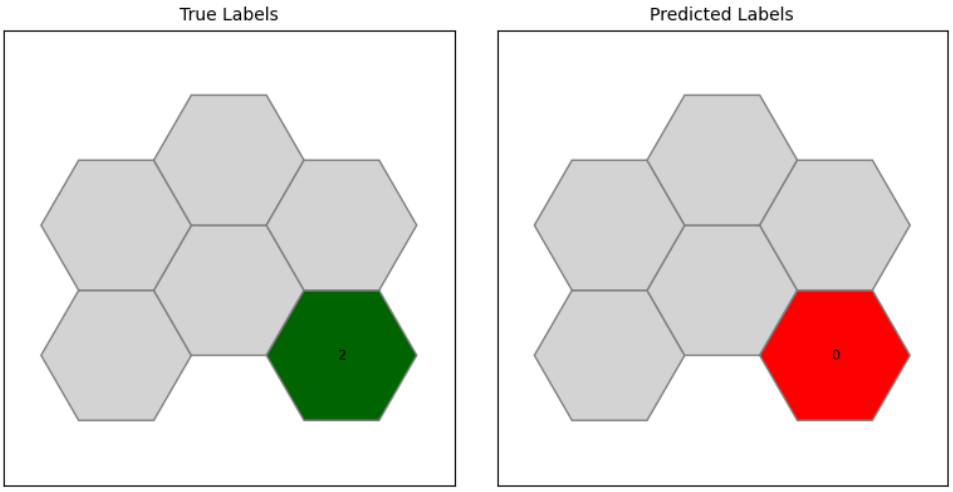
Rysunek 3.7: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: topologia rectangle, funkcja `gaussian`, topologia rect, sigma=1.0.



Rysunek 3.8: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: topologia rectangle, funkcja `gaussian`, topologia rect, sigma=1.0.



Rysunek 3.9: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: topologia hexagon, funkcja `mexican_hat`, sigma=1.0.



Rysunek 3.10: Porównanie danych z przyporządkowanymi i prawdziwymi klasami dla konfiguracji: topologia hexagon, funkcja `mexican_hat`, sigma=1.0.

3.4. Wnioski

Na podstawie przeprowadzonych eksperymentów można wyciągnąć kilka ogólnych wniosków dotyczących jakości klasteryzacji oraz wpływu wybranych parametrów sieci Kohonena na proces uczenia.

Jakość klasteryzacji

Zastosowanie odpowiednich parametrów, takich jak funkcja sąsiedztwa oraz wartość promienia sąsiedztwa, ma istotny wpływ na jakość klasteryzacji. Najlepsze wyniki uzyskano w konfiguracjach z funkcją `gaussian` oraz odpowiednią wartością sigma (najczęściej 1.0 lub 10.0). Dobrze dobrane parametry pozwalają na znalezienie klastrów, które odpowiadają rzeczywistym klasom danych, w tym przypadku dla zbioru `MNIST` oraz `Activity Recognition`.

W przypadku funkcji `mexican_hat`, klasteryzacja nie powiodła się, a wszystkie metryki jakościowe wskazywały na brak skutecznej klasyfikacji. To potwierdza, że dobór funkcji sąsiedztwa ma kluczowe znaczenie dla jakości wyników.

Wpływ siatki neuronów

Topologia siatki (prostokątna vs sześciokątna) ma zauważalny wpływ na proces klasteryzacji, ale nie jest to decydujący czynnik. Choć siatka prostokątna wykazywała lepszą wydajność w niektórych przypadkach, różnice te nie były wystarczająco duże, aby jednoznacznie stwierdzić, że jedna topologia jest zawsze lepsza od drugiej.

W przypadku zbioru `MNIST` oraz `Activity Recognition`, zarówno topologia prostokątna, jak i sześciokątna dawały porównywalne wyniki klasteryzacji, co sugeruje, że inne parametry, takie jak funkcja sąsiedztwa czy promień sąsiedztwa, mają większy wpływ na jakość klasteryzacji niż sama topologia siatki.

Rola promienia sąsiedztwa

Wartość promienia sąsiedztwa σ również odgrywa istotną rolę w procesie klasteryzacji. Wyższe wartości tego parametru (takie jak 1.0) umożliwiły lepsze znalezienie większej liczby klastrów, co okazało się bardziej odpowiednie w kontekście rzeczywistej liczby klas w zbiorach danych. Natomiast zbyt małe wartości σ prowadziły do zbyt małej liczby klastrów, co miało miejsce w przypadku funkcji `mexican_hat`.

Podsumowanie jakości klasteryzacji

Pod względem zgodności znalezionych klastrów z rzeczywistym podziałem na klasy, wyniki eksperymentów były zróżnicowane. W niektórych przypadkach klasteryzacja wykazała się dobrą zgodnością, jednak w innych wynik była mniej satysfakcjonujący. Analizując wykresy słupkowe, zauważono, że w większości przypadków pojedynczy neuron w sieci Kohonena łączył w sobie dane z różnych klas, co nie jest idealnym rozwiązaniem, ponieważ wskazuje na zbyt szeroką generalizację. Taka sytuacja, w której jeden neuron reprezentuje wiele klas, utrudnia dokładne odwzorowanie struktury danych i jest efektem niedoskonałego procesu klasteryzacji.

Wielu wyników klasteryzacji nie można uznać za idealne, co może wynikać z niewłaściwego doboru hiperparametrów, szczególnie tych związanych z funkcją sąsiedztwa, promieniem sąsiedztwa oraz topologią siatki neuronów. Ważnym czynnikiem może być również zbyt mała przestrzeń hiperparametrów do przeszukania, co ogranicza możliwości optymalizacji procesu klasteryzacji. Istnieje prawdopodobieństwo, że rozszerzenie zakresu hiperparametrów, np. poprzez wprowadze-

nie dodatkowych wartości, mogłoby poprawić jakość uzyskanych wyników i bardziej precyzyjnie odwzorować podział na klasy w danych.

W skrócie, chociaż w niektórych przypadkach udało się uzyskać dobre wyniki, to jednak w większości eksperymentów klasteryzacja nie była w pełni zgodna z rzeczywistymi klasami, a jakość wyników zależała w dużym stopniu od staranności w doborze parametrów oraz ograniczeń przestrzeni przeszukiwania.