Metody inteligencji obliczeniowej

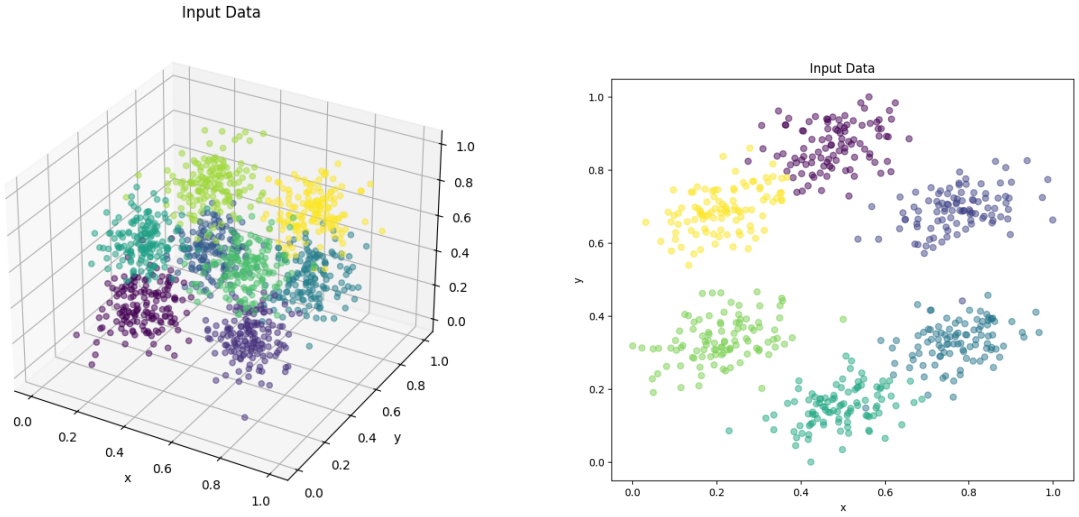
Sieci Kohonena – sprawozdanie

Jakub Kaproń 327282

maj 2025

1 Podstawowa sieć Kohonena

Celem pierwszego zadania było zaimplementowanie podstawowej sieci Kohonena (z siatką prostokątną) i zbadanie wpływu rodzaju funkcji sąsiedztwa i szerokości sąsiedztwa na jakość wyników. Do tego należało użyć dwóch dostarczonych zbiorów, *hexagon* oraz *cube*, gdzie punkty były skupione w narożnikach odpowiednio sześciokąta i sześcianu. Klasy puntków odpowiadały narożnikom, w którym się one znajdowały.



(a) *cube* (b) *hexagon*

Rysunek 1: Zbiory, na których przeprowadzano analizę.

Funkcje sąsiedztwa, które należało przebadać to gaussowska (f_1) oraz meksykański kapelusz (f_2), których wzory przedstawiają się następująco:

$$f_1(x) = \exp\left(-\frac{x^2}{2\sigma^2}\right),$$
$$f_2(x) = \left(1 - \frac{x^2}{\sigma^2}\right)\exp\left(-\frac{x^2}{2\sigma^2}\right),$$

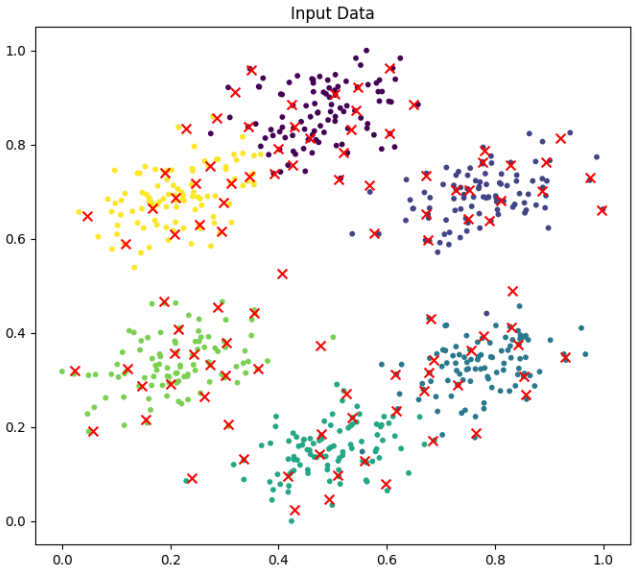
gdzie za x należy podstawić odległość między punktami, a σ^2 to parametr szerokości sąsiedztwa. Od razu nadmienię, że niestety, z nieidentyfikowanego przeze mnie powodu, uczenie sieci z meksykańskim kapeluszem nie działało – rozbiegały one do nieskończoności, gdzieś następował overflow.

1.1 hexagon

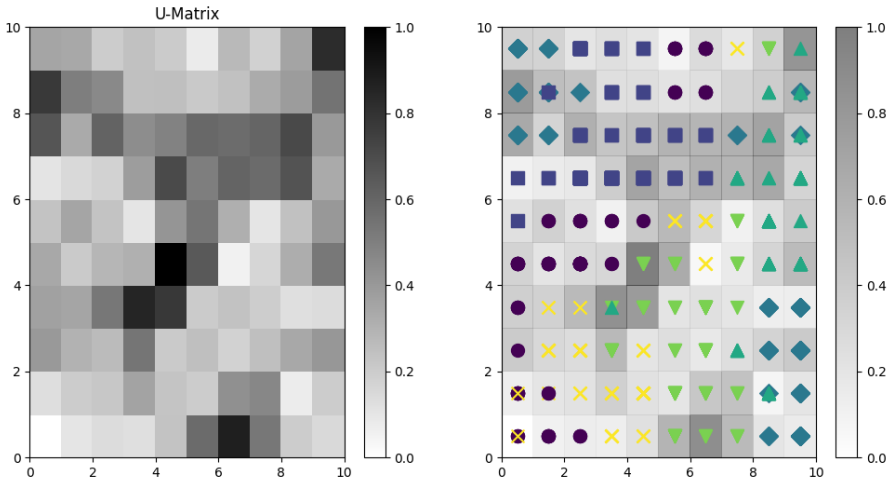
1.1.1 Wpływ szerokości sąsiedztwa

Tak jak zadano przebadałem, kilka wartości z przedziału $[0.1, 10]$, mianowicie $\sigma^2 \in \{0.1, 1, 2, 10\}$. Przyjęty rozmiar sieci to 10×10.

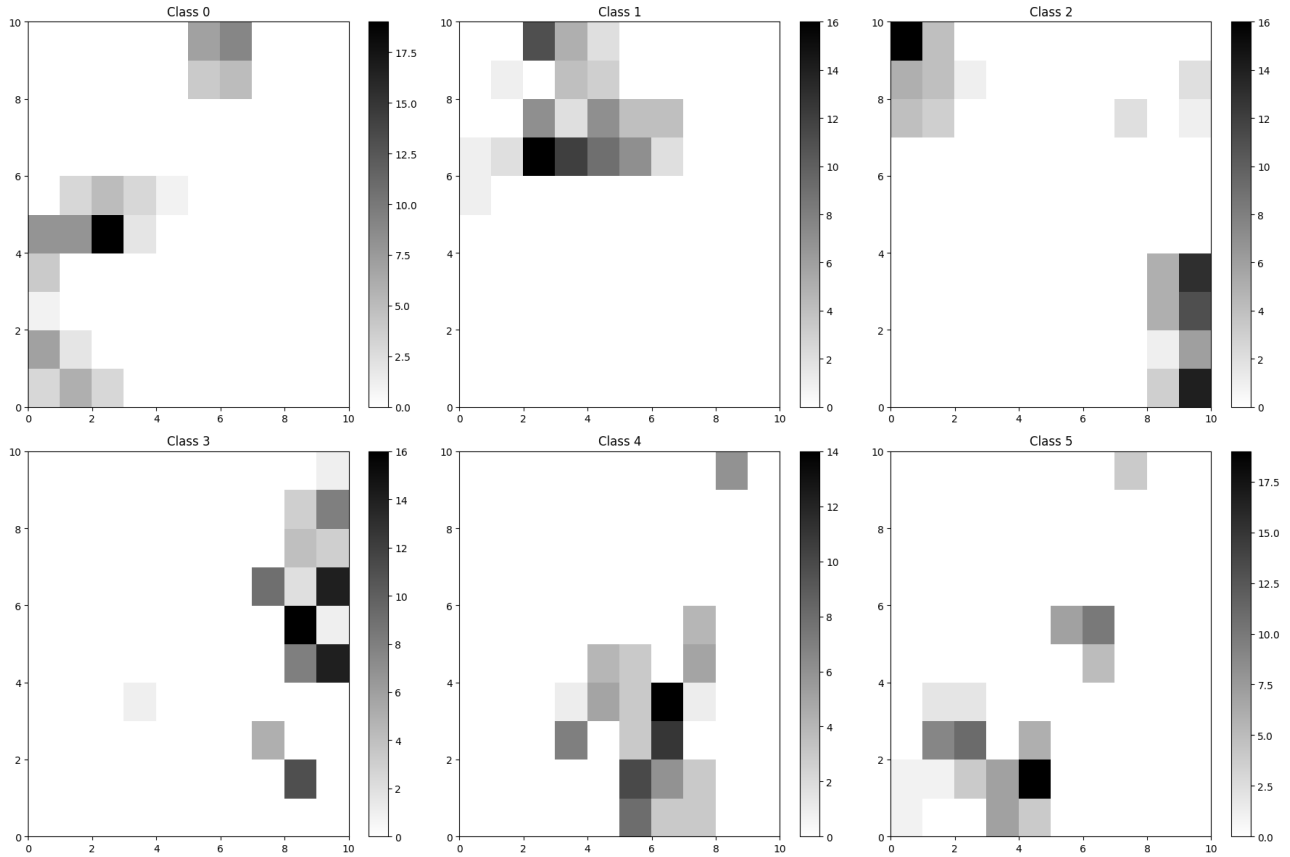
W przypadku $\sigma^2 = 0.1$, że neurony dość szeroko pokrywają dane i ciężko jest je pogrupować w klastry. Gdy zwiększymy szerokość do 1 lub 2, wtedy sieć wydaję się najlepiej układać. Tworzą bardziej zarysowane klastry, które jednocześnie dość dobrze odzwierciedlają prawdziwe klastry. Po 10-tce widać, że im bardziej byśmy zwiększali szerokość, tym bardziej neurony zaczęły by się zbliżać do środka. Dla tej wartości odwzorowanie już nie ma sensu.



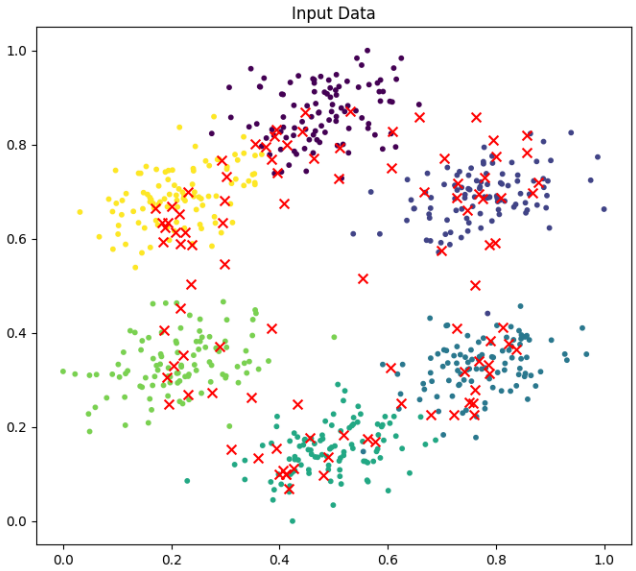
Rysunek 2: Położenie neuronów dla $\sigma^2 = 0.1$.



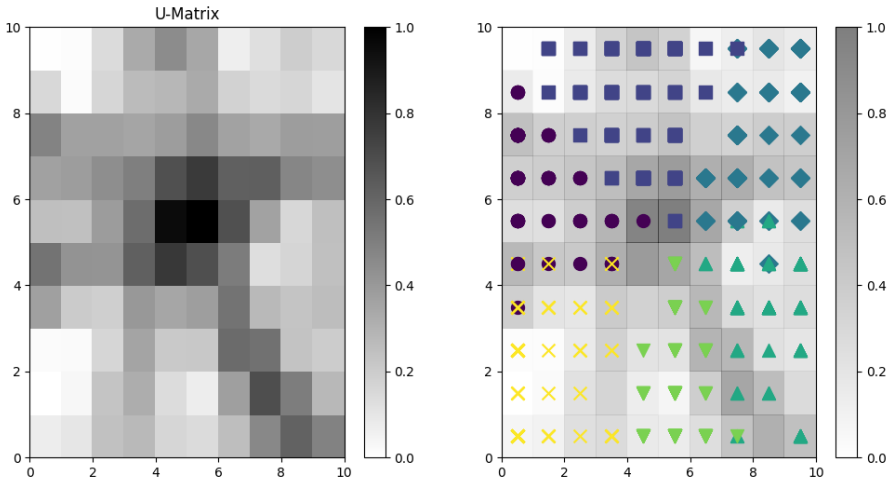
Rysunek 3: Wizualizacja pokazująca, które neurony odpowiadają więcej niż jednej klasie ($\sigma^2 = 0.1$).



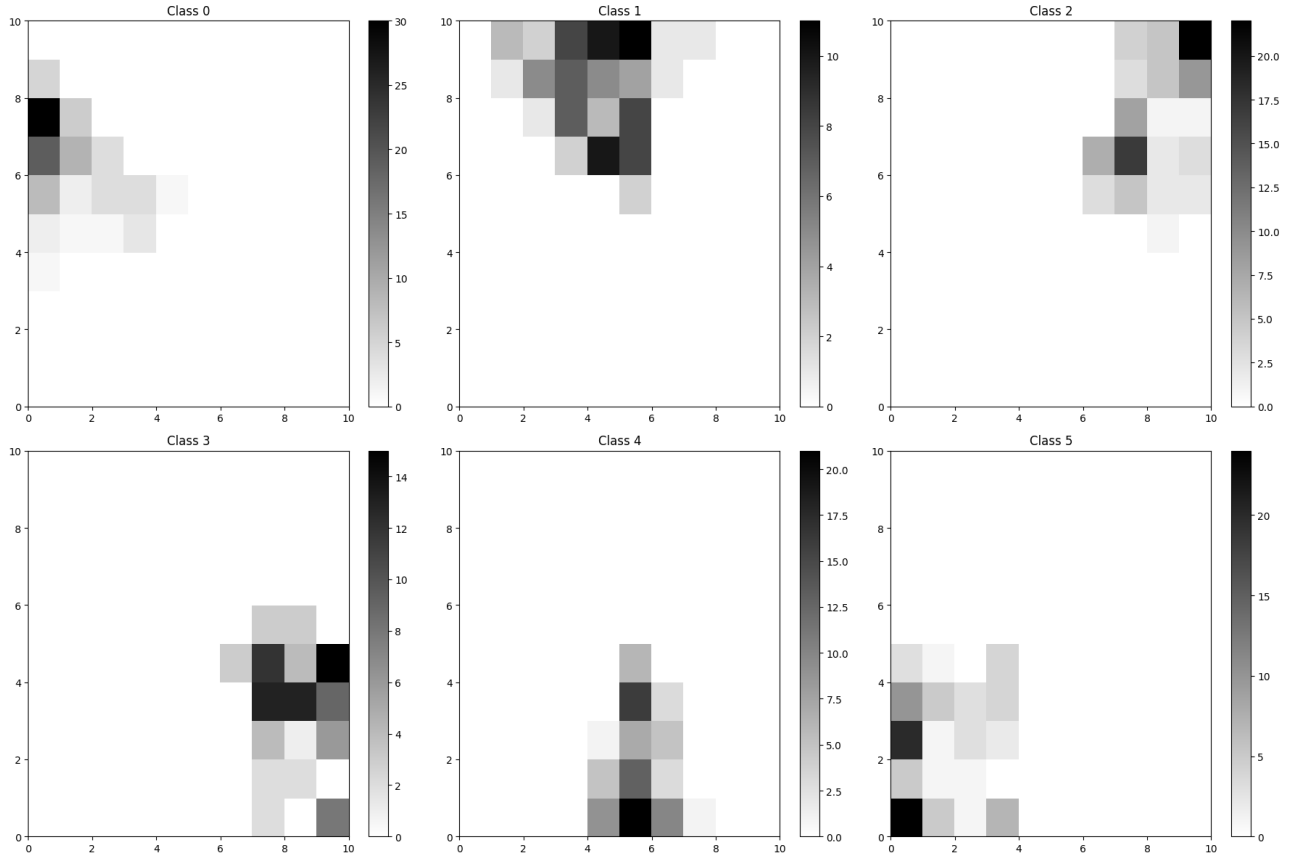
Rysunek 4: Wizualizacja pokazująca, liczność obserwacji z danej klasy w poszczególnych neuronach ($\sigma^2 = 0.1$).



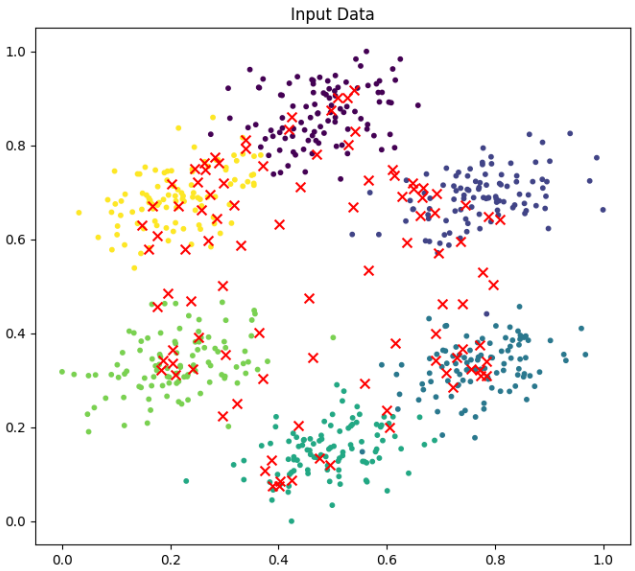
Rysunek 5: Położenie neuronów dla $\sigma^2 = 1$.



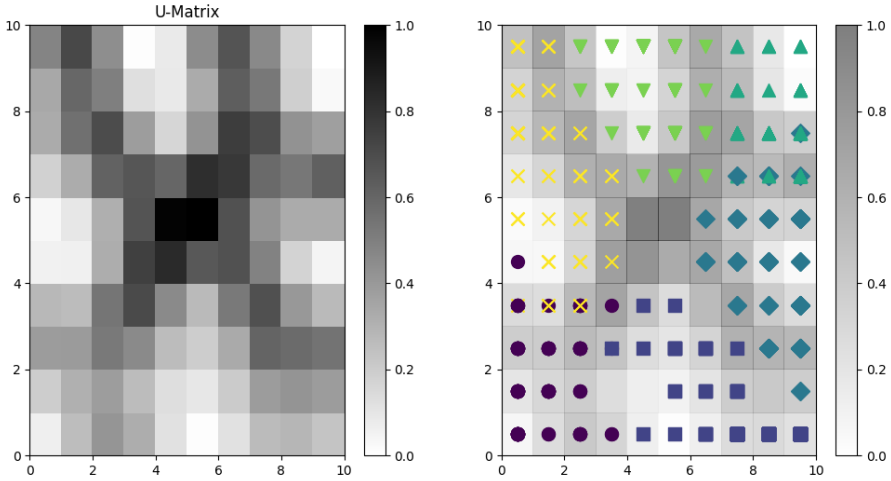
Rysunek 6: Wizualizacja pokazująca, które neurony odpowiadają więcej niż jednej klasie ($\sigma^2 = 1$).



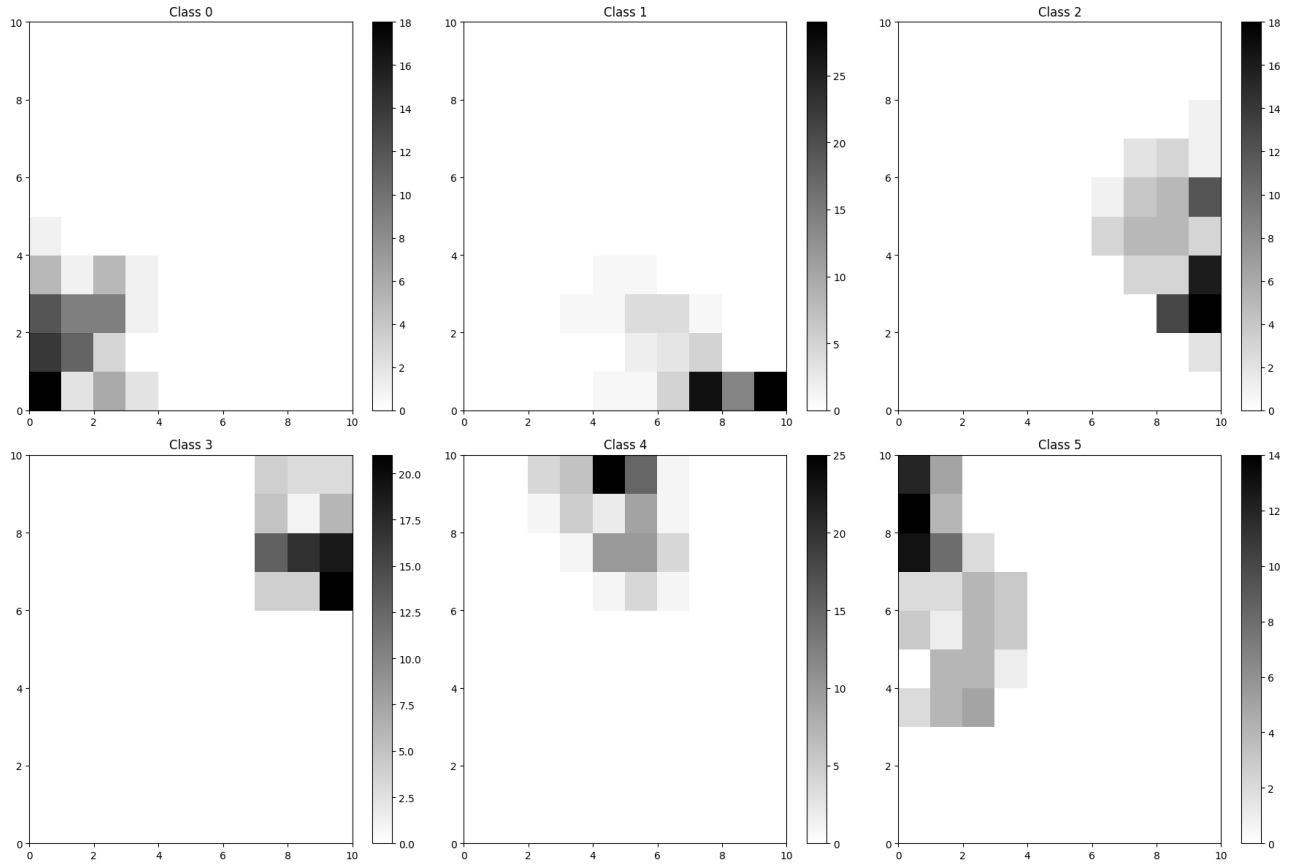
Rysunek 7: Wizualizacja pokazująca, liczność obserwacji z danej klasy w poszczególnych neuronach ($\sigma^2 = 1$).



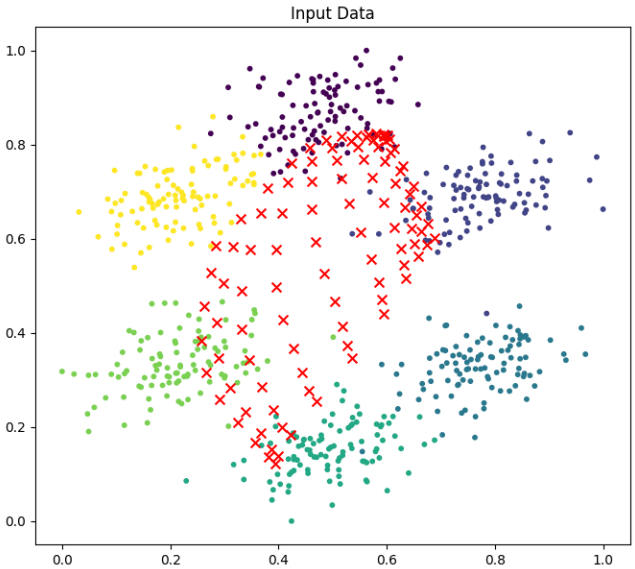
Rysunek 8: Położenie neuronów dla $\sigma^2 = 2$.



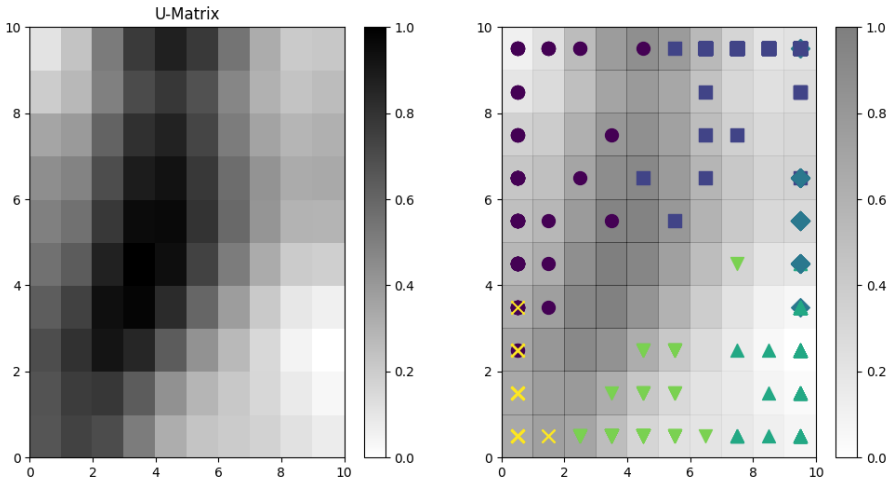
Rysunek 9: Wizualizacja pokazująca, które neurony odpowiadają więcej niż jednej klasie ($\sigma^2 = 2$).



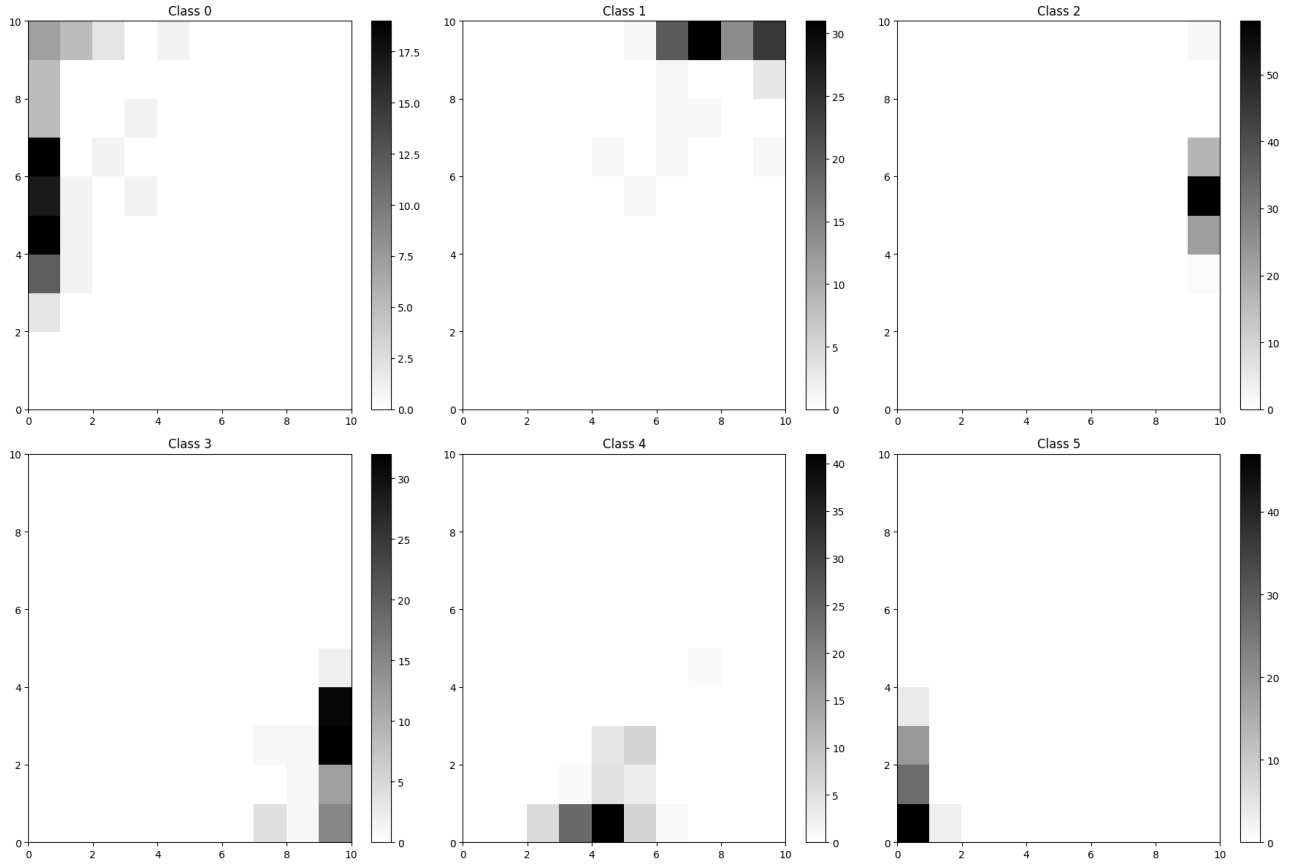
Rysunek 10: Wizualizacja pokazująca, liczność obserwacji z danej klasy w poszczególnych neuronach ($\sigma^2 = 2$).



Rysunek 11: Położenie neuronów dla $\sigma^2 = 10$.



Rysunek 12: Wizualizacja pokazująca, które neurony odpowiadają więcej niż jednej klasie ($\sigma^2 = 10$).



Rysunek 13: Wizualizacja pokazująca, liczność obserwacji z danej klasy w poszczególnych neuronach ($\sigma^2 = 10$).

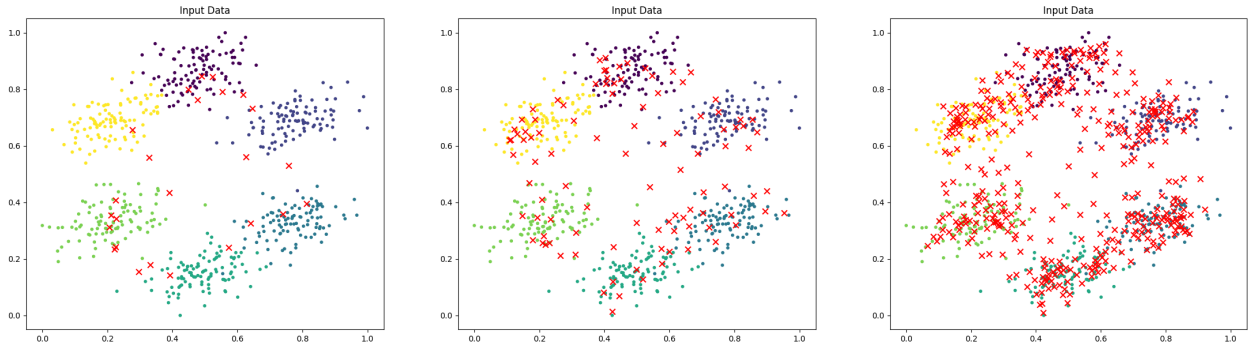
Przebadałem jeszcze w ile klastrów neurony się układają, obierając następującą strategię – trenowałem K-Means dla kolejnych liczb klastrów i z tych modeli ten, który maksymalizuje Silhouette Score. Jedynie dla $\sigma^2 = 10$ liczba klastrów różniła się od 6.

σ^2	liczba klastrów
0.1	6
1	6
2	6
10	2

Tabela 1: Liczba klastrów neuronów w sieci dla poszczególnych szerokości sąsiedztwa.

1.2 Wpływ rozmiarów siatki

Przebadałem także jak wyglądają wyniki dla różnych rozmiarów sieci: 5×5, 10×10 oraz 20×20. Większy rozmiar daje lepszą dokładność, ale znacząco zwiększa czas uczenia – 10×10 okazało się najbardziej optymalne.

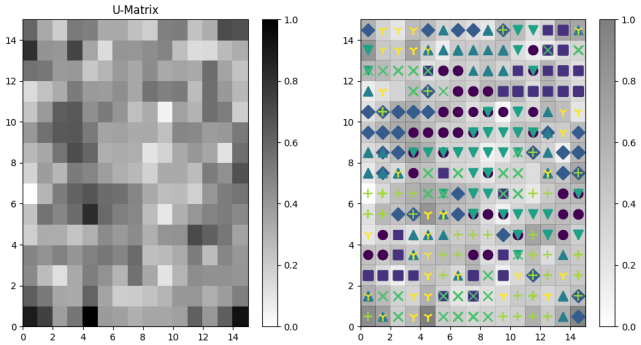


(a) 5×5 (b) 10×10 (c) 20×20

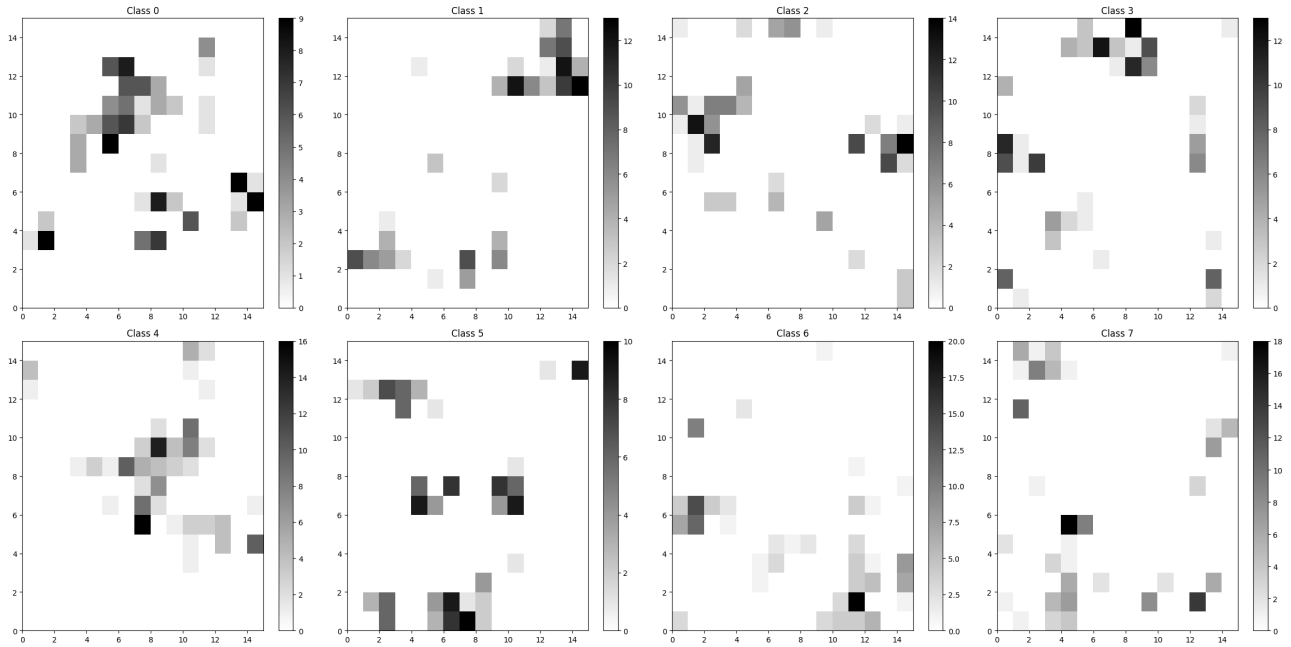
Rysunek 14: Wyniki dla poszczególnych rozmiarów siatek.

1.3 cube

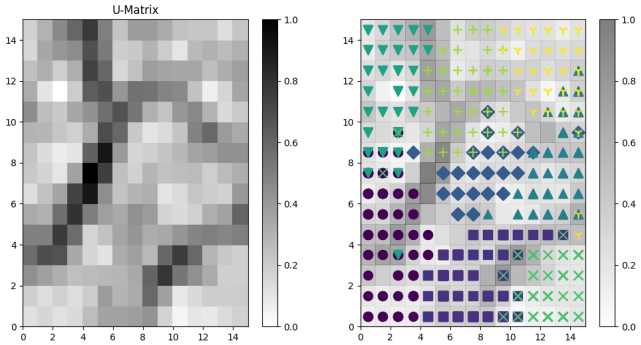
W przypadku tego zbioru zwiększyłem rozmiar siatki do 15×15, z powodu większej liczby klastrów w oryginalnych danych. Wyniki są podobne do poprzedniego zbioru. Sąsiedztwo w okolicy 1 wydaje się dawać najsensowniejsze wyniki.



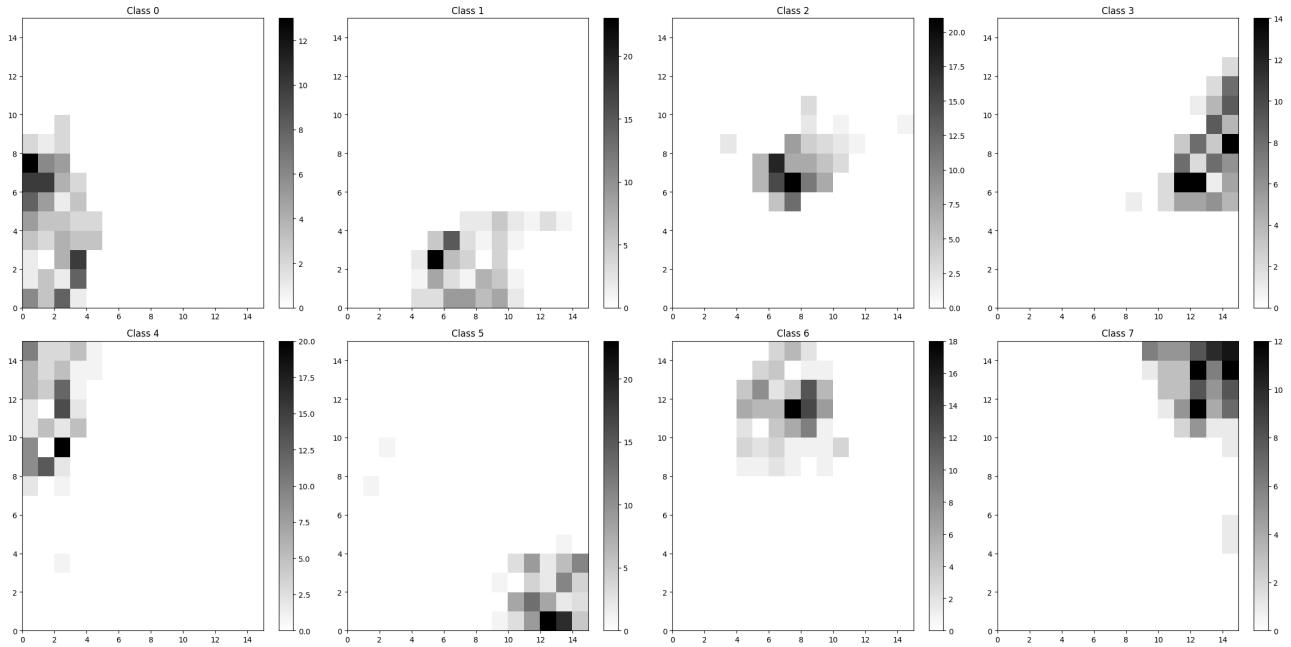
Rysunek 15: Wizualizacja pokazująca, które neurony odpowiadają więcej niż jednej klasie ($\sigma^2 = 0.1$).



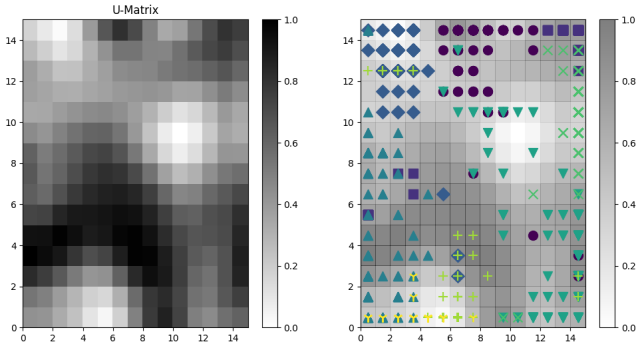
Rysunek 16: Wizualizacja pokazująca, liczność obserwacji z danej klasy w poszczególnych neuronach ($\sigma^2 = 0.1$).



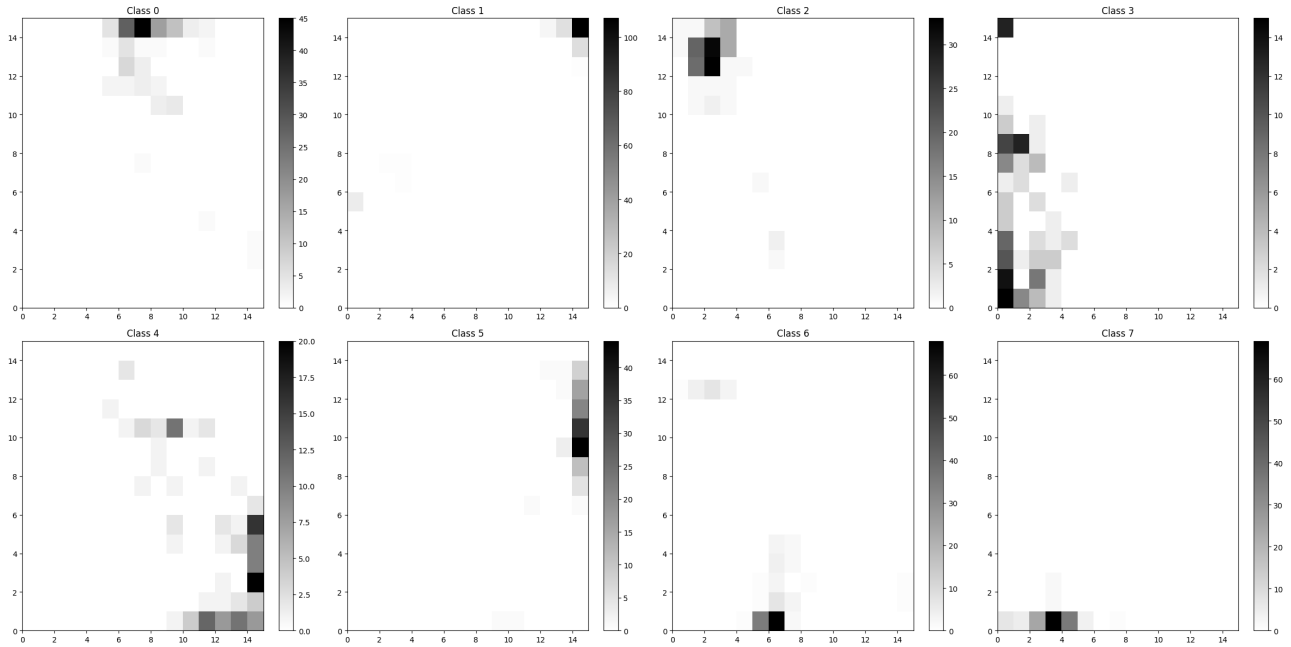
Rysunek 17: Wizualizacja pokazująca, które neurony odpowiadają więcej niż jednej klasie ($\sigma^2 = 1$).



Rysunek 18: Wizualizacja pokazująca, liczność obserwacji z danej klasy w poszczególnych neuronach ($\sigma^2 = 1$).



Rysunek 19: Wizualizacja pokazująca, które neurony odpowiadają więcej niż jednej klasie ($\sigma^2 = 10$).



Rysunek 20: Wizualizacja pokazująca, liczność obserwacji z danej klasy w poszczególnych neuronach ($\sigma^2 = 10$).

2 Siatka sześciokątna

Jedyna zmiana w kodzie potrzebna do implementacji siatki sześciokątnej, to przesunięcie indeksów kolumn w parzystych wierszach o pół. Jeśli chodzi o identyfikowanie klastrów, zastosowałem analogiczne podejście jak poprzednio – wybór optymalnego K-means.

Niestety jednak wyniki nie były zadowalające. Na obu zbiorach testowych (MNIST i HAR) otrzymane tą metodą klastry nie były poprawne, a wyniki nie były zależne od topologii. W przypadku MNIST otrzymałem 20 klastrów, zamiast 10 (wynik mógłby być nawety wyższy gdyby nie ograniczenie z góry maks. liczby klastrów). Natomiast w przypadku HAR tylko 2, zamiast 6.