Metody inteligencji obliczeniowej w analizie danych

Sprawozdanie z budowy sieci Kohonena.

Damian Skowroński — 313506

5 maja 2023

1 Wprowadzenie

Celem projektu była implementacja sieci Kohonena. Taki typ sieci jest szerzej znany pod nazwą SOM - *Self-organizing Map*. Sieć można wykorzystywać do zadań klasteryzacji. W implementacji można dostosowywać sieć i jej uczenie poprzez wybór:

- rozmiaru siatki sieci
- topologii sieci - siatka prostokątna, lub sześciokątna
- funkcji sąsiedztwa, zaimplementowane funkcje to funkcja gaussowska i druga pochodna funkcji gaussowskiej
- parametru szerokości sąsiedztwa dla funkcji sąsiedztwa
- parametru odpowiedzialnego za funkcję wygaszającą uczenie w kolejnych epochach

To sprawozdanie skupia się głównie na wnioskach wyciągniętych podczas pracy z użyciem sieci, a nie na samym kodzie i sposobie implementacji sieci.

2 Test działania na prostych zbiorach

Na początku przetestowałem, czy sieć działa tak jak powinna na prostych zbiorach, z którymi powinna sobie z łatwością poradzić. Przeprowadzałem klasteryzację przez co porównywałem wyniki metryk:

- completeness - miara, czy wszystkie próbki należące do danej klasy są przypisane do tego samego klastra
- homogeneity - miara, czy każdy klaster zawiera tylko próbki należące do jednej klasy
- v-measure - średnia harmoniczna z completeness i homogeinty, daje bardziej ogólną miarę do porównywania sieci

Dla każdego zbioru miałem też zmienne objaśniane jednak nie używałem ich w uczeniu sieci. Przydały się natomiast jej ocenie. Dobrze dopasowana sieć powinna dobrze przewidywać klasy, wobec tego po wytrenowaniu sieci i dopasowaniu przewidywanych klas klastrów do prawdziwych otrzymywałem metrykę F1. Ta metryka ostatecznie będzie najważniejszą w porównywaniu wyników.

W teście sprawdzałem jak sobie radzą sieci z różnymi funkcjami sąsiedztwa. W celu znalezienia optymalnych parametrów wykorzystałem grid search.

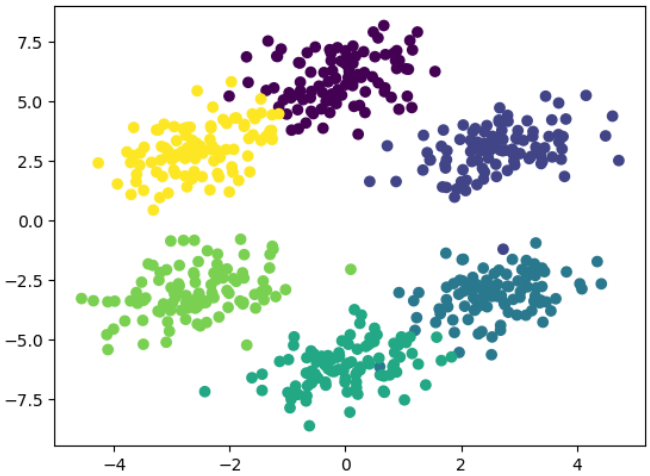
2.1 zbiór hexagon

Na rysunku 1 widać rozkład punktów w tym zbiorze. Zmiennymi objaśnianymi są dwuwymiarowe współrzędne. Labele są przedstawione jako kolory. Zbiór to punkty wygenerowane w okolicach wierzchołków sześciokąta.

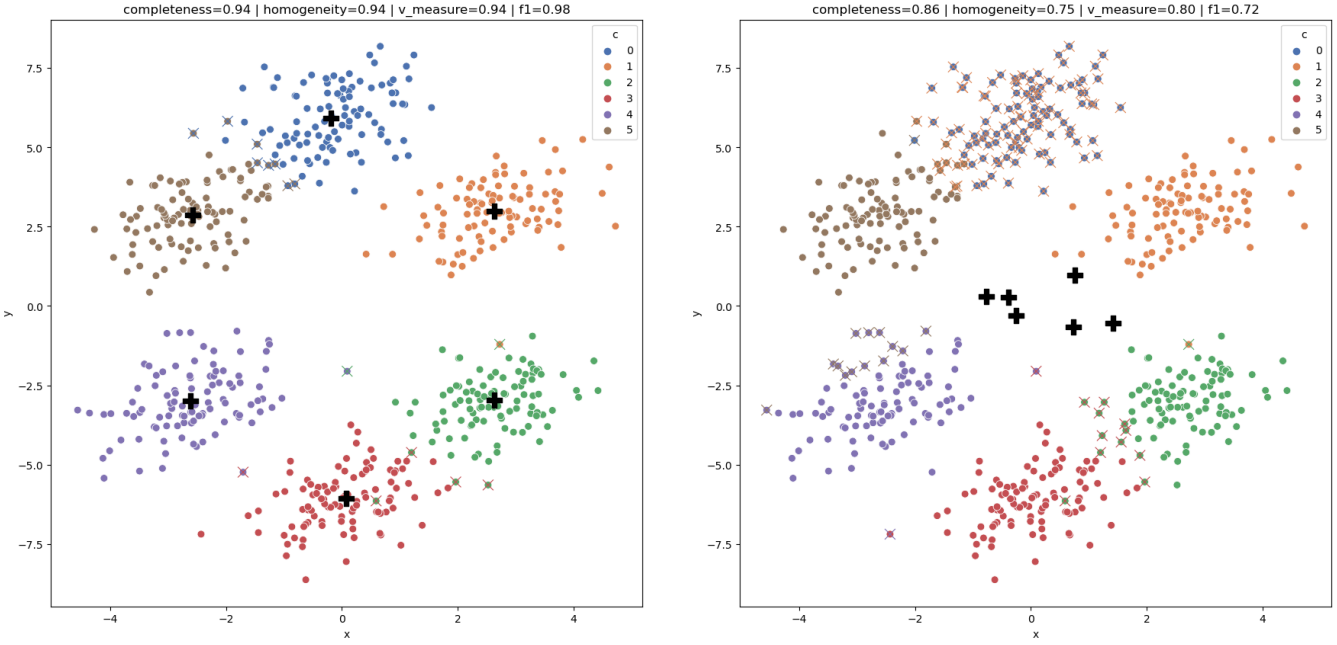
Już podczas grid searchu zauważyłem, że funkcja sąsiedztwa gaussian radzi sobie dużo lepiej niż gaussian second derivative, zawsze otrzymywała znacząco lepsze wyniki v-measure. Gaussian $\sim$ 0.8-0.9, a gaussian second derivative $\sim$ 0.6-0.7.

Na rysunku 2 widać wyniki z dopasowania sieci o 6 neuronach do zbioru:

(a) Dla gaussowskiej funkcji sąsiedztwa można powiedzieć, że sieć idealnie się dopasowała (v-measure = 0.94, F1 = 0.98). Punkty, które zostały źle zaklasyfikowane (są w złym klastrze) są albo na granicy, albo w chmurze punktów innej klasy. Ponieważ sieć ma dostęp tylko do ich współrzędnych to nic dziwnego, że te punkty są źle sklasyfikowane.



Rysunek 1: Rozkład punktów i ich klasy w zbiorze hexagon.



(a) funkcja sąsiedztwa - gaussian

(b) funkcja sąsiedztwa - gaussian second derivative

Rysunek 2: Dopasowanie sieci do zbioru hexagon, czarne plusy to pozycja neuronów

(b) Dla funkcji sąsiedztwa drugiej pochodnej gaussowskiej sieć dopasowała się dużo gorzej. Przede wszystkim wygląda to tak jakby sieć nie działała tak jak powinna, to znaczy neurony nie ustawiły się na środku grup punktów, tylko trochę się do nich przybliżyły. Zwiększenie liczby epoch i zmiana parametrów nie dawała lepszych wyników. W dopasowaniu z rysunku widać, że gdyby sieci udało się lepiej pogrupować górny środkowy klaster to dopasowanie byłoby dobre.

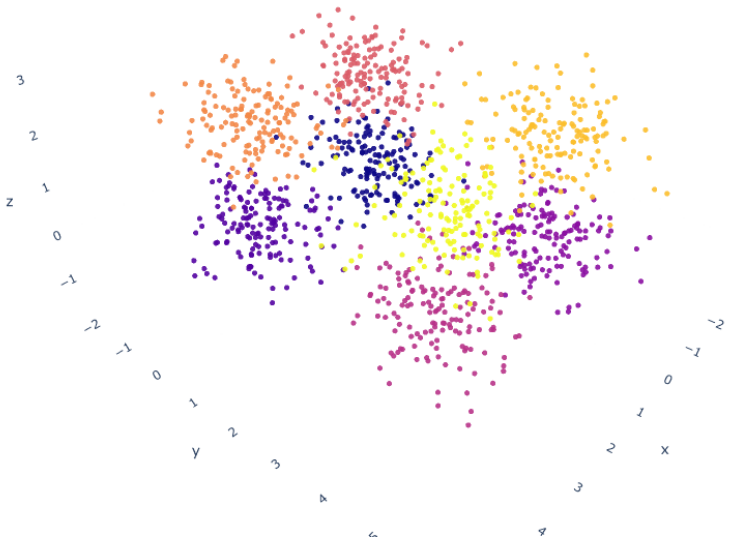
Ważnym faktem jest, że zwiększanie liczby neuronów, czyli zwiększanie rozmiarów siatki nie dawało lepszych wyników.

2.2 zbiór cube

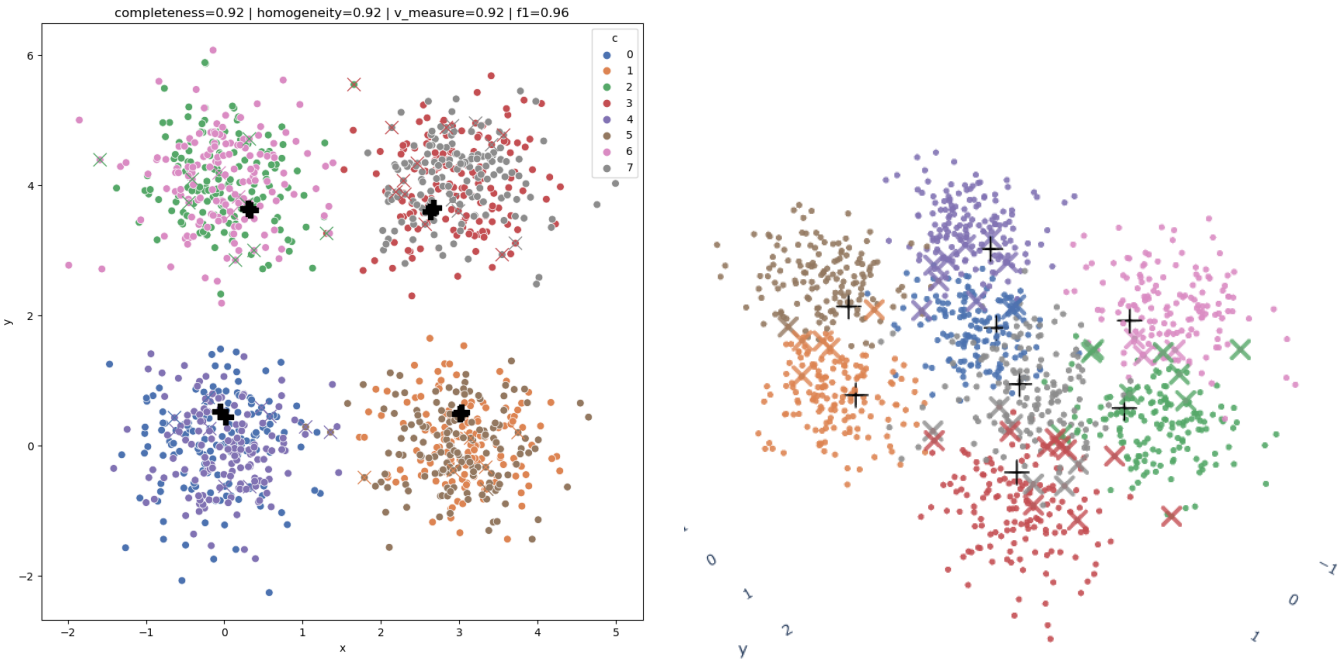
Punkty ze zbioru widać na rysunku 3. Punkty są trójwymiarowe i są rozłożone jakby na wierzchołkach sześcianu. Myślę, że głównym celem było tutaj sprawdzenie, czy to możliwe, żeby siatka dopasowała się do trzech wymiarów. Nie pokazuję wyników dla drugiej pochodnej funkcji gaussowskiej, bo wychodziły bardzo słabe. Do dopasowania sieci wykorzystywałem siatkę rozmiarów 4x2 z 8 neuronami dla 8 klas. Na rysunku 4 widać wyniki klasteryzacji:

(a) Patrząc na wykres dopasowania z góry widać, że pomimo tego, że zbiorowiska punktów pokrywają się, to są odpowiednio rozdzielone na klastry. Z tej perspektywy wygląda jakby neurony miały parami te same położenie. Widać, że sieć się dobrze dopasowała z wynikiem F1 = 0.96 i niewiele się myli.

(b) Na wykresie w trzech wymiarach lepiej widać jak rozkładają się grupy punktów i odpowiadające im neurony. Można zauważyć, że błędy występują głównie na granicach grup z różnych wysokości.



Rysunek 3: Rozkład punktów i ich klasy w trzech wymiarach dla zbioru cube.



(a) widok w dwóch wymiarach z góry

(b) widok w trzech wymiarach

Rysunek 4: Dopasowanie sieci do zbioru cube, czarne plusy to pozycja neuronów.

2.3 Wnioski

Wnioski z tej części są następujące:

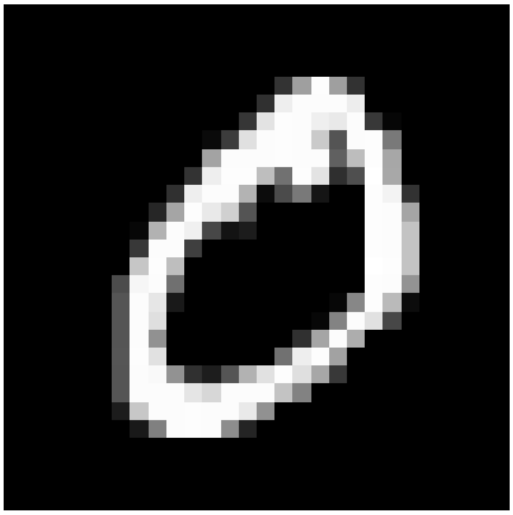
- Sieć potrafi się dobrze dopasowywać do bardzo prostych zbiorów punktów 2, lub 3 wymiarowych.
- Wybór funkcji sąsiedztwa ma bardzo duże znacznie do poprawnego działania sieci. Funkcja gaussowska radziła sobie bardzo dobrze, a jej druga pochodna znacznie słabiej.
- Z przeprowadzonych grid searchów wynika, że dobór reszty parametrów również ma niemałe znaczenie w efektywności sieci.
- Przy testach na tak prostych zbiorach dodawanie większej ilości neuronów niż oczekiwanych klas nie działało na korzyść sieci.

3 Siatka sześciokątna i test działania na trudniejszych zbiorach

Do tej pory używałem tylko sieci o topologii prostokątnej. W tej sekcji porównałem jej działanie z siatką sześciokątną. W teorii siatka sześciokątna może działać lepiej, ponieważ neuron ma 6 bliskich sąsiadów jeśli nie jest na krańcu siatki, w porównaniu do 4 dla siatki prostokątnej.

3.1 zbiór MNIST

Przeprowadziłem klasteryzację na zbiorze MNIST, która polega na rozpoznawaniu obrazów pisanych cyfr. Jedna obserwacja to obrazek 28x28 pikseli z wartościami od 0 do 255. Na rysunku 5 przedstawiona jest przykładowa obserwacja. Wartości pikseli są w odcieniach szarości. Jako input do sieci używam wartości pikseli w wektorze o 784 wartościach.



Rysunek 5: Przykładowa obserwacja ze zbioru MNIST - pisana cyfra 0.

Tutaj, ponieważ cyfry wyglądają różnie jeśli pisane są przez różne osoby i często niektóre są do siebie podobne, kluczowe było wybranie odpowiedniej liczby neuronów. Z jednej strony jeśli neuronów jest za mało to sieć nie będzie w stanie dobrze rozróżniać cyfr. Z drugiej strony jeśli sieć będzie za duża to wyniki również nie wychodzą dobre. Wygląda wtedy jakby niektóre neurony dopasowywały się do szumu. Sprawdzałem najpierw sieci z funkcją gaussowską o rozmiarach siatki:

- 2x5 - 10 neuronów - 10 cyfr, po neuronie na cyfrę
- 5x5 - 25 neuronów, trochę większa sieć
- 10x10 - 100 neuronów, duża sieć
- 8x6 - 48 neuronów, rozmiar sieci po policzeniu wykorzystanych neuronów dla sieci 10x10

3.1.1 siatka 2x5

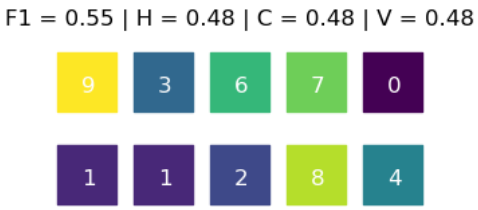
Na rysunku 6 widać, że 10 neuronów to zdecydowanie za mało ze względu na:

- Niektóre cyfry są pomijane przez sieci. W obu topologiach nie ma cyfry 5. W topologii sześciokątnej brakuje też cyfry 2.
- Nie ma wystarczająco dużo neuronów, aby sieć potrafiła dobrze rozróżniać pomiędzy cyframi 4, 7 i 9.
- Są różne sposoby zapisywania cyfr. Topologia prosta zrobiła podział na proste 1 i pochylone 1. Takich wariantów zapisywania może być wiele dla różnych cyfr.
- Widać, że topologie działają trochę inaczej. Cyfry w prostokątnej siatce są bardziej wyraźne, a w topologii sześciokątnej bardziej zamazane, wyglądają jakby miały w sobie też obrazy innych cyfr.
- W tym przypadku siatka prostokątna poradziła sobie lepiej niż sześciokątna.

Wniosek: potrzebna jest siatka pozwalająca na większą ilość neruonów.

(a) prostokątna siatka - dopasowanie neuronów

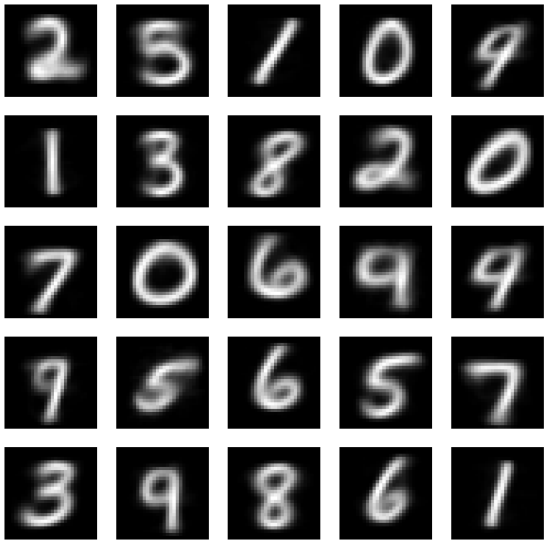
(b) sześciokątna siatka - dopasowanie neuronów



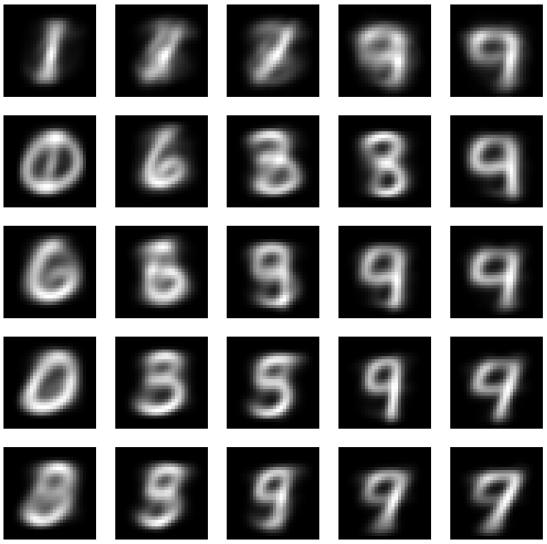
(c) prostokątna siatka - przewidywane klasy

(d) sześciokątna siatka - przewidywane klasy

Rysunek 6: Dopasowanie sieci do zbioru MNIST dla siatki 2x5

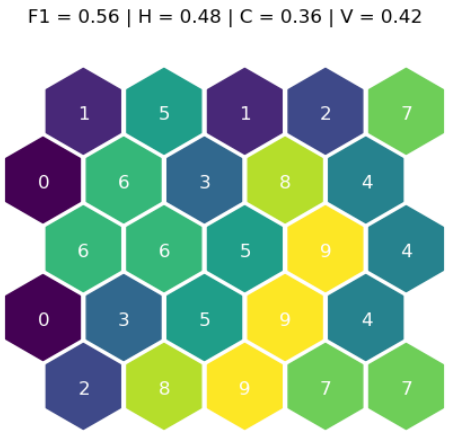


(a) prostokątna siatka - dopasowanie neuronów



(b) sześciokątna siatka - dopasowanie neuronów

(c) prostokątna siatka - przewidywane klasy



(d) sześciokątna siatka - dopasowanie neuronów

Rysunek 7: Dopasowanie sieci do zbioru MNIST dla siatki 5x5

3.1.2 siatka 5x5

Wyniki tych sieci są lepsze zarówno pod względem metryki F1 (i V-measure w przypadku prostokątnej siatki też V-score), ale też pod względem tego co widzimy na dopasowaniach neuronów na rysunku 7. Tutaj znowu neuronów jest za mało dla optymalnego działania sieci, ponieważ:

- Nadal jest duży problem z rozróżnianiem 4 i 9. Tak naprawadę wszystkie neurony reprezentujące 4 wciąż mają górną kreskę.
- W obu siatkach uwidocznił się nowy problem z rozróżnianiem cyfr 3, 5 i 8 (neuron o indeksie (0,1) na wykresie (a) wygląda bardziej jak 5, a jest klastrowany jako 3).
- Znowu widać, że siatki działają w inny sposób. Klastry siatki sześciokątnej bardziej zbierają się w grupy podobnych neuronów. Na wykresie (b) widać, że klastry podobnych do siebie liczb (3,5,8) i (4,7,9) są obok siebie.

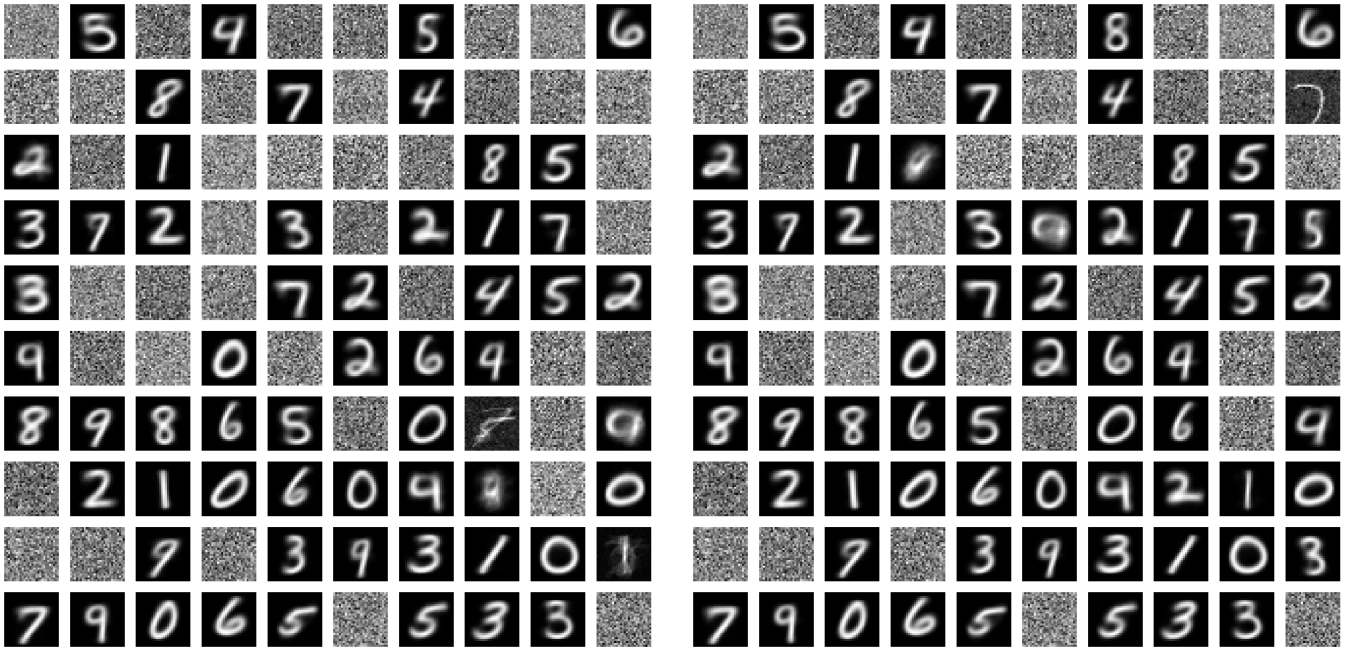
Wnioski to: nadal potrzebna jest większa siatka, możliwe, że wymagane będzie wiecej epoch do dobrego wyuczenia.

3.1.3 siatka 10x10

W celu sprawdzenia, gdzie może być optymalny limit liczby neuronów sprawdziłem siatkę o 100 neuronach, czyli aż 4 razy większą niż poprzednia. Wyniki dopasowania dla siatki prostokątnej w zależności od liczby epoch są na rysunku 8. Ważne informacje jakie wyciągnąłem dla siatki prostokątnej to:

- Za duża liczba neuronów sprawia, że wiele z nich dopasowują się do szumu (tak mi się wydaje).
- Neurony, które są szumem po większej ilości epoch mają szansę dopasować się do którejś cyfry. Może to oznaczać, że dla większej liczby epoch taka sieć by się dobrze dopasowała, jednak na pewno jest to bardzo czasochłonne. 40 epoch na tej sieci zajęło mi $\sim$ 15 min.
- Te sieci mają lepsze wyniki homogeneity, niż poprzednie. Ma to sens, jest więcej klastrów więc jeśli są dopasowane to powinna być mniejsza szansa, że w klastrze są próbki z innego klastra.
- V-measure wyszły takie same jak dla sieci 5x5.
- Completeness score jest mniejszy niż w sieci 5x5, co może wskazywać, że neuronów jest za dużo.
- F1 jest bardzo małe. Co ciekawe po 20 epochach wyszło lepsze niż po 40, mimo że po 40 epochach więcej neuronów się "wyklarowało". Podejrzewam, że tak małe F1 jest związane z "zaszumionymi" neuronami.

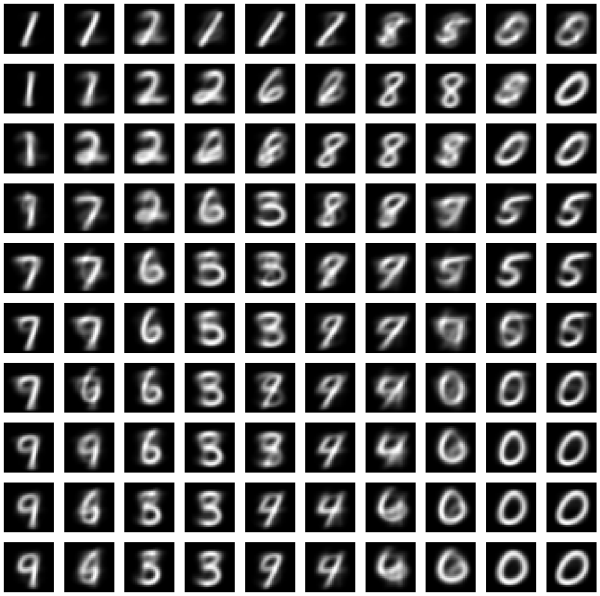
Po policzeniu neuronów, które dają ładny obraz na wykresie (a) otrzymałem wynik 48. Wobec tego następną siecią jaką wybróbowałem (następna sekcja) jest sieć o siatce 8x6 z dokładnie 48 neuronami.



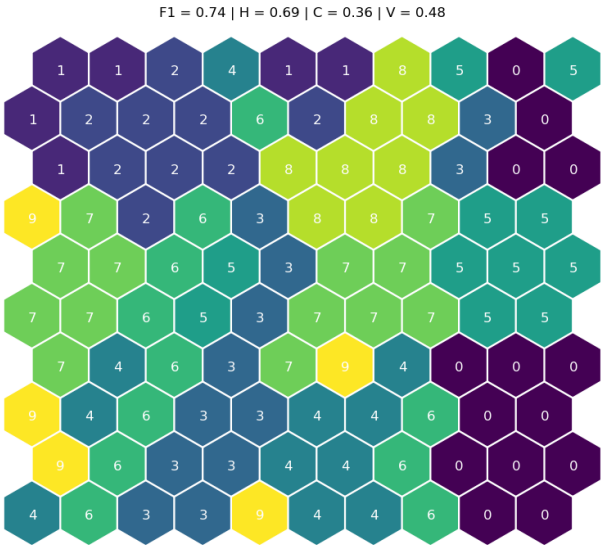
(a) po 20 epochach, F1 = 0.16, H = 0.74, C = 0.43, V = 0.54 (b) po 40 epochach, F1 = 0.10, H = 0.75, C = 0.42, V = 0.54

Rysunek 8: Dopasowanie sieci do zbioru MNIST dla siatki prostokątnej 10x10

Wyniki dopasowania siatki sześciokątnej o rozmiarach 10x10 są na rysunku 9 i znacząco różnią się od dopasowania siatki prostokątnej. Tutaj wszystkie neurony zostały poprawnie wykorzystane. Wynik jest ciekawy, że niektóre neurony wyglądają klarownie i sieć dobrze je przewiduje. Są też takie, z którymi sieć nie jest pewna i wyglądają jak połączenie dwóch podobnych cyfr. Są też neurony, które wyglądają mocno jak konkretna cyfra, jednak przypisywane są im jest zła klasa. Dla tej siatki otrzymałem najlepszy wynik F1 spośród innych rozmiarów. Myślałem, że zwiększanie sieci polepszy mój wynik, jednak wówczas sieć miała problem z dopasowaniem. Możliwe, że da się wybrać odpowiednie parametry, żeby to się udało.



(a) sześciokątna siatka - dopasowanie neuronów

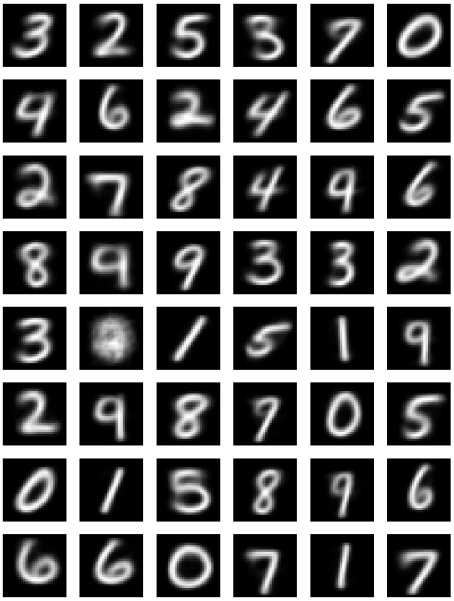


(b) sześciokątna siatka - przewidywane klasy

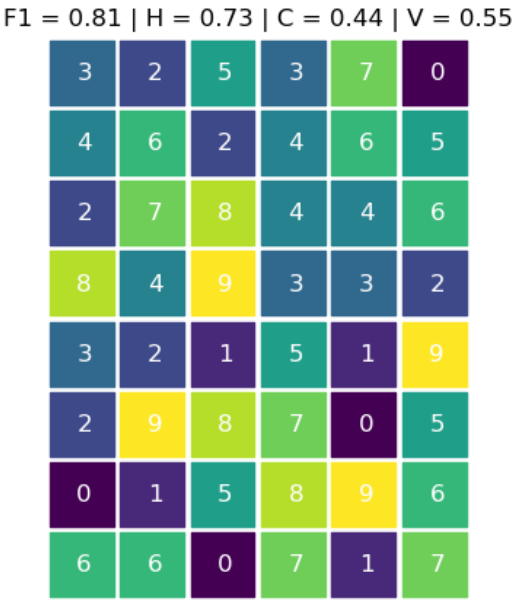
Rysunek 9: Dopasowanie sieci do zbioru MNIST dla siatki sześciokątnej 10x10

3.1.4 siatka 8x6 - siatka prostokątna

Wyniki dopasowania przedstawione są na rysunku 10. Jest to do tej pory zdecydowanie najlepsza sieć. Ma najlepsze do tej pory wyniki F1 i większość cyfr rozpoznawanych przez neurony wydaje się całkiem "ostra". Prawie wcale nie pojawia się tu problem mylenia cyfr 3 i 5, chodź nadal czasem widać "pozostałości" po drugiej cyfrze. Prawie całkowicie został wyeliminowany też problem rozróżniania 9 i 4. Widać, że ilość neuronów wydaje się raczej odpowiednia. Problematyczny jest jedynie neuron wyglądający jak biała plama. Myślałem, ze jest to problem niewystarczającej liczby epoch, jednak po podwojeniu jej nic się nie zmienia. Neurony wyglądają tak samo, wyniki również się nie zmieniają.



(a) dopasowanie neuronów



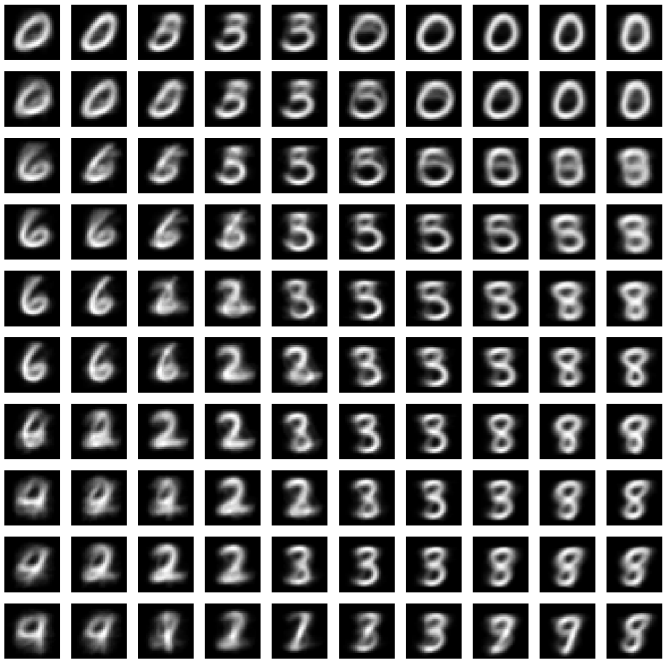
(b) przewidywane klasy

Rysunek 10: Dopasowanie sieci do zbioru MNIST dla siatki prostokątnej 8x6

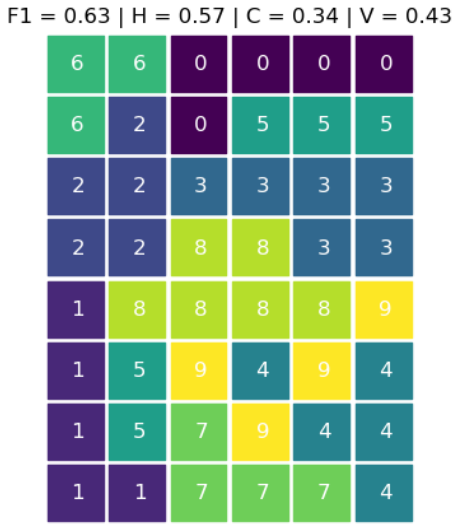
3.1.5 funkcja sąsiedztwa - druga pochodna funkcji gaussowskiej

Mając doświadczenie z obiema topologiami dla funkcji gaussowskiej, wykonałem klasteryzajcę też przy użyciu funkcji sąsiedztwa drugiej pochodnej gaussowskiej (rysunek 1). Wykorzystałem rozmiary siatek, które dały wcześniej najlepsze wyniki. Widać, że te sieci radzą sobie trochę gorzej jednak wyniki nie są słabe. Tym razem siatka sześciokątna okazała się trochę lepsza. Co ciekawe przy użyciu tej funkcji sieć o topologii prostokątnej zachowała się bardziej tak jak się spodziewałem, czyli klastry podobnych wartości są obok siebie w grupach.

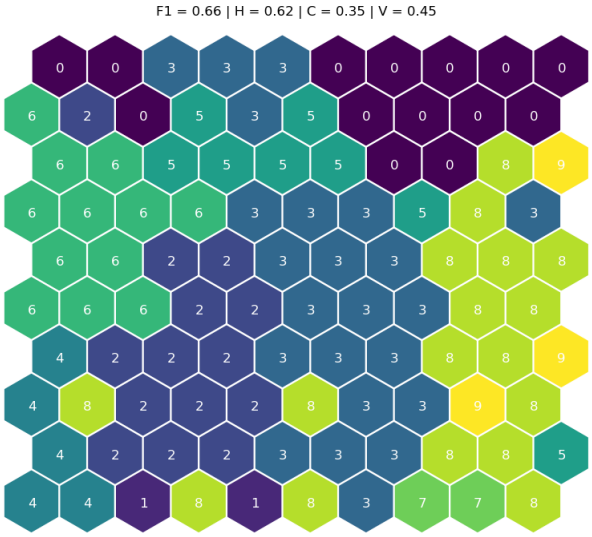
(a) prostokątna siatka - dopasowanie neuronów



(b) sześciokątna siatka - dopasowanie neuronów



(c) prostokątna siatka - przewidywane klasy



(d) sześciokątna siatka - przewidywane klasy

Rysunek 11: Dopasowanie sieci z funkcją sąsiedztwa drugiej pochodnej funkcji gaussowskiej do zbioru MNIST

3.2 zbiór Human Activity Recognition

Zbiór dotyczy 6 aktywności człowieka: chodzenia, wchodzenia i schodzenia po schodach, siedzenia, stania, leżenia. Te aktywności są klasami. Zmienne opisujące sprocesowane odczyty z akcelerometru i żyroskopu z telefonu. Jedna obserwacja odpowiada 561 wartościom ze zmiennych opisujących. Ogólnie to są różne liczby, wiec nie ma dobrego sposobu na ich wizualizację.

3.2.1 siatka 2x3

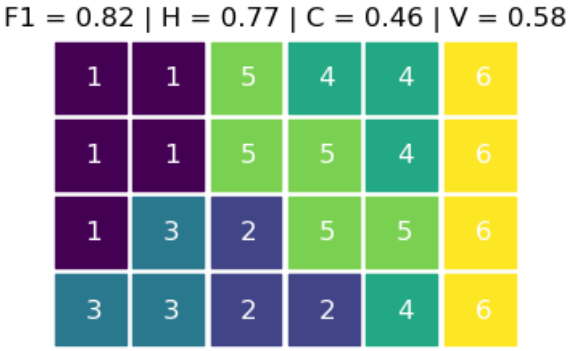
Dla siatki 2x3 na 100 epochach otrzymałem następujące wyniki:

- funkcja sąsiedztwa - gaussian, topologia - rectangle: F1 = 0.53, H = 0.59, C = 0.67, V = 0.63
- funkcja sąsiedztwa - gaussian, topologia - hexagon: F1 = 0.46, H = 0.56, C = 0.67, V = 0.61
- funkcja sąsiedztwa - guassian second derivative, topologia - rectangle: F1 = 0.34, H = 0.41, C = 0.54, V = 0.45
- funkcja sąsiedztwa - guassian second derivative, topologia - hexagon: F1 = 0.32, H = 0.41, C = 0.54, V = 0.47

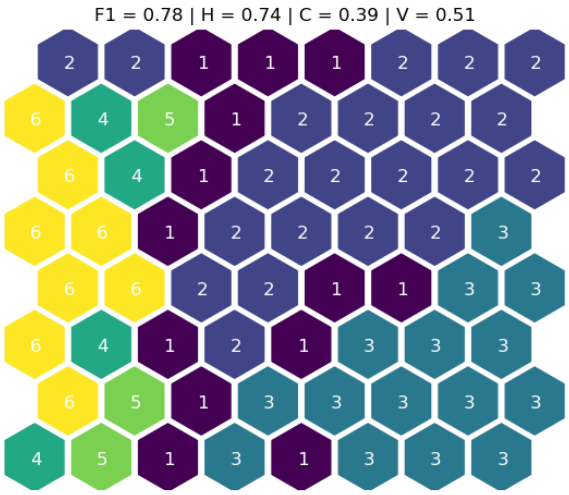
Tutaj znowu gaussowska funkcja sąsiedztwa okazała się znacznie lepsza. Topologia prostokątna dała lepsze wyniki niż sześciokątna.

3.2.2 większe siatki

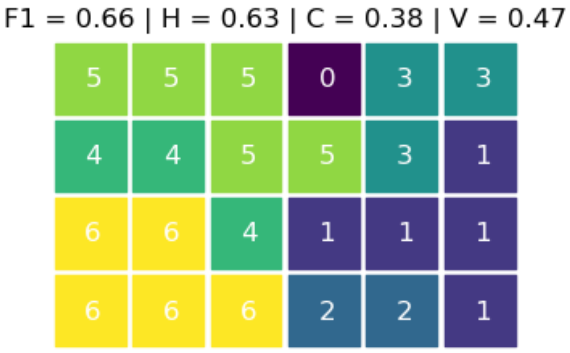
Na rysunku 12 widać jak sieciom udało się dopasować do zbioru w różnych ustawieniach i różnych rozmiarów siatek. Topologia prostokątna dawała najlepsze wyniki dla siatki 4x6. Topologia sześciokątna radziła sobie lepiej na większych siatkach. Wyniki dla gaussowskiej funkcji sąsiedztwa wyszyły bardzo dobre. W przypadku funkcji sąsiedztwa drugiej pochodnej gaussowskiej wyniki są nieco gorsze, lecz i tak zaskakująco dobre w porównaniu do poprzednich zbiorów. Ciekawym zjawiskiem jest fakt, że na wykresie (c) jest neuron odpowiadający klasy "0", która nie istnieje. Okazuje się, że ten neuron nie odpowiada żadnej z obserwacji, tak jakby nie dopasował się do niczego.



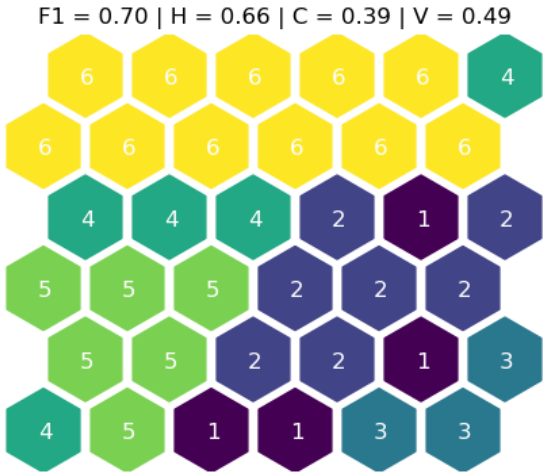
(a) funkcja sąsiedztwa - gaussian, topologia - rectangle



(b) funkcja sąsiedztwa - gaussian, topologia - hexagon



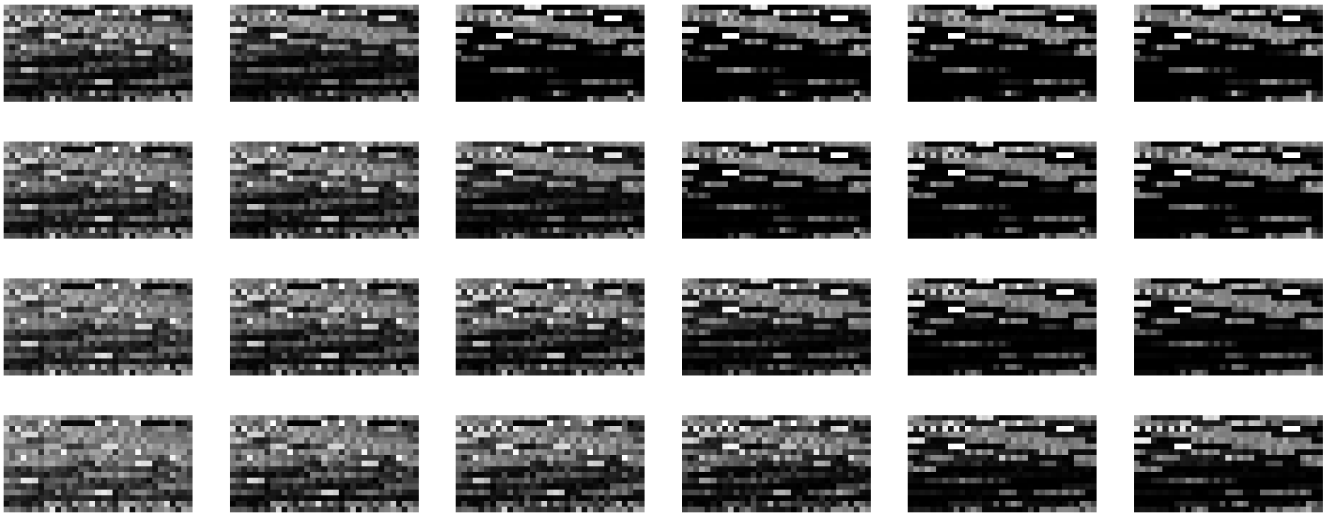
(c) funkcja sąsiedztwa - guassian second derivative, topologia - rectangle



(d) funkcja sąsiedztwa - guassian second derivative, topologia - hexagon

Rysunek 12: Dopasowanie sieci do zbioru Human Activity Recognition z siatką 4x6 w zależności od ustawień

Na rysunku 12 przedstawiłem jeszcze jak wyglądają neuruony po dopasowaniu sieci. W tym przypadku jest to sieć odpowiadająca wykresowi (a) z rysunku 11. Każdy pixel odpowiada to jakiejś zprocesowanej wartości, więc nic dziwnego że nie ma tu łatwej do rozpoznania struktury jak w zbiorze MNIST. Można natomiast zauważyć, że neurony odpowiadające klasom 1,2,3, czyli te z lewej strony są znacznie jaśniejsze niż te z prawej strony. Jest to oczekiwany wynik ponieważ klasy 1,2,3 odpowiadają poruszaniu się: 1-chodzenie, 2-wchodzenie po schodach, 3-schodzenie po schodach, za to klasy 4, 5, 6 odpowiadają za aktywności stacjonarne: 4-siedzenie, 5-stanie, 6-leżenie.



Rysunek 13: Neurony przy dopasowaniu sieci funkcja sąsiedztwa - gaussian, topologia - rectangle do zbioru HAR

3.3 Wnioski

Wnioski z tej części są następujące:

- Sieć potrafiła również dobrze dopasować się do bardziej skomplikowanych danych. W poprzedniej części testowana była na danych o 2, lub 3 zmiennych, a w tej części zmiennych było 784 i 561.
- Oczywiście więcej zmiennych, więcej neuronów, więcej obserwacji oznacza dłuższy czas uczenia.
- Wybór pomiędzy topologią prostokątną, miał duże znaczenie. Zwykle prostokątna topologia dawała lepsze wyniki.
- Funkcja sąsiedztwa gaussian second derivative okazała się być gorsza niż funkcja gaussowska. Za każdym razem wyniki dla niej były gorsze.
- Co ciekawe sieci Kohonena nie zawsze zwracały pogrupowane podobne klastry (tak jak jest na rysunku 11) jak się tego spodziewałem. Dla zbioru MNIST w topologii prostokątnej tak naprawdę wychodziła mozaika bez większego porządku.
- Wybór liczby neuronów w siatce jest kluczowy w działaniu sieci. Za duża siatka sprawiała, że uczenie było bardzo długie, a przewidywanie niedokładne - dawało gorsze wyniki niż siatka o optymalnym rozmiarze. Za mała siatka sprawiała, że sieć nie była w stanie się nauczyć dostatecznie o zbiorze.
- Często zdarzała się sytuacja, kiedy sieć była już dobrze dopasowana, ale większa liczba epoch sprawiała, że dopasowanie pogarszało się. Podejrzewam, że spowodowane to było za dużym learning ratem, albo za małym learning rate decayem.

4 Podsumowanie

W ramach tego projektu została zaimplementowana sieć Kohonena składająca się z siatki neuronów o topologi sześciokątnej, lub prostokątnej. Sieć można z sukcesami wykorzystywać do problemu klasteryzacji na zbiorach bardzo prostych i trochę trudniejszych. Praca z użyciem sieci na zbiorach pozwoliła na wyciągnięcie cennych wniosków dotyczących wyboru parametrów sieci. Ostatecznie wybory są zależne od danych, jednak niektóre funkcje sąsiedztwa okazały się po prostu gorsze od innych.