INTELIGENTNA ANALIZA DANYCH

Zadanie 2 – Sieci samoorganizujące się

Informatyka, studia dzienne, semestr VI, rok akademicki 2016/2017,

Laboratoria: czwartek, 8:15

Rafał Lebioda 195639
Piotr Kaźmierczak 195611

Spis treści

Cel zadania

Zadanie polegało utworzeniu sieci samoorganizującej się która umożliwiłaby znalezienie najlepszego rozkładu neuronów, który by najlepiej odzwierciedlał losowy zbiór punktów treningowych na płaszczyźnie. W tym celu należało zaimplementować trzy różne algorytmy :

- Algorytm k-średnich
- Algorytm Kohonena
- Algorytm gazu neuronowego

Sieci samoorganizujące

Sieci samoorganizujące się należą do typu sieci uczonych „bez nauczyciela”, zwanych także „sieciami nienadzorowanymi”. Oznacza to, że podczas treningu dla podawanych danych wejściowych nie są przedstawiane żadne wzorce wyjścia (prawidłowe odpowiedzi). Zadaniem sieci jest dopiero stworzenie takich wzorców podczas etapu uczenia się. W sieciach z uczeniem konkurencyjnym (uczeniem z rywalizacją) po prezentacji wzorca wejściowego następuje określenie neuronu wygrywającego i tylko ten neuron, ewentualnie grupa sąsiadujących z nim neuronów aktualizuje swoje wagi, tak by zbliżyć je do aktualnego wzorca. Ogólna idea wyboru neuronu polega na znalezieniu takiego neuronu, którego wektor wag jest najbardziej podobny, najbliższy pewnej miary podobieństwa, metryki, do prezentowanego wzorca wejściowego. Oczekuje się, że podobne wzorce wejściowe powinny wywoływać podobne odpowiedzi sieci.

Architektura sieci samoorganizującej

Bardzo istotną kwestią jest struktura sieci neuronowej. Pojedynczy neuron jest mechanizmem bardzo prostym i przez to niewiele potrafiącym. Dopiero połączenie wielu neuronów ze sobą umożliwia prowadzenie dowolnie skomplikowanych operacji. Najczęściej stosuje się w tego typu sieciach architekturę jednokierunkową jednowarstwową. Jest to podyktowane faktem, że wszystkie neurony

muszą uczestniczyć w konkurencji na równych prawach. Dlatego każdy z nich musi mieć tyle wejść ile jest wejść całego systemu.

Algorytm Kohonena

Adaptacja wag neuronów

W trakcie uczenia sieci samoorganizujących na wejście każdego neuronu podawany jest N-wymiarowy sygnał x ze zbioru wzorców uczących (losowo lub w ustalonej kolejności). Wagi połączeń synaptycznych tworzą wektor $W_i=[w_{i1}, w_{i2}, ..., w_{iN}]$. We współzawodnictwie zwycięża jeden neuron, którego wagi najmniej różnią się od odpowiednich składowych wektora x.

Następnie wagi neuronu zwycięzcy podlegają adaptacji, która ma na celu zbliżenie go do typowego reprezentanta grupy. Istnieją dwie znane metody aktualizacji wektorów wag neuronów. W naszym podejściu zastosowaliśmy metodę która polega na lekkim przyciąganiu wektora wag w kierunku wektora wejściowego :

$$W_i(k+1) = W_i(k) + \gamma_i(k)(x - W_i(k)) \quad \text{gdzie} \quad \gamma - \text{współczynnik nauki}$$

Istnieją dwa algorytmy zmiany wag:

- WTA (Winner Takes All) – w metodzie tej tylko zwycięski neuron ma możliwość adaptacji swoich wag
- WTM (Winner Takes Most) – W metodzie tej oprócz wag zwycięskiego neuronu zmianie podlegają również wagi sąsiadów według reguły :

$$W_i(k+1) = W_i(k) + \gamma_i(k)G(i,x)(x - W_i(k)) \qquad \text{gdzie } G(i,x) jest\ funkcją\ sąsiedztwa$$

Kohonen zaproponował dwa rodzaje sąsiedztwa: prostokątne i gaussowskie. Pierwsze ma postać:

$$G(i,x) = \begin{cases} 1\ dla\ d(i,w) \leq \lambda \\ 0\ dla\ d(i,w) > \lambda \end{cases} \quad \text{gdzie } \lambda - \text{promień sąsiedztwa}$$

Jednakże w naszym programie zostało zastosowane sąsiedztwo gaussowskie :

$$G(i,x) = \exp\left(-\frac{d^2(i,w)}{2\ \lambda^2}\right)$$

O stopniu adaptacji neuronów z sąsiedztwa zwycięzcy decyduje tutaj nie tylko odległość neuronu i-tego od zwycięzcy, ale również promień sąsiedztwa. Tak więc stopień adaptacji jest zróżnicowany.

Miara odległości między wektorami

Istnieje kilka metod mierzenia odległości między wektorami. Najczęściej używane miary to :

- Miara euklidesowa
- Iloczyn skalarny
- Miara według normy L1 (Manhattan)
- Miara według normy

W naszym zadaniu zdecydowaliśmy się użyć miary euklidesowej która jest dana wzorem :

$$d(x, W_i) = \sqrt{\sum_{j=1}^{N} (x_j - W_j^{(i)})^2}$$

Martwe neurony

Lepsze rezultaty samoorganizacji obserwuje się, jeśli algorytm uczący uwzględnia liczbę zwycięstw poszczególnych neuronów i organizuje proces uczenia w taki sposób, aby dać szansę zwycięstwa neuronom mniej aktywnym. W praktycznym zastosowaniu tej zasady w sieciach samoorganizujących wprowadza się potencjał p_i dla każdego neuronu, modyfikowany po każdej k-tej prezentacji wzorca wejściowego x, według zależności :

$$p_i(k+1) = \begin{cases} p_i(k) + \frac{1}{n} \; dla \; i \neq w \\ p_i(k) - p_{min} dla \; i = w \end{cases}$$

Współczynnik p_{min} oznacza minimalny potencjał upoważniający do udziału we współzawodnictwie. Jeśli aktualna wartość potencjału spadnie poniżej p_{min}, to neuron i-ty "odpoczywa", a zwycięzcy poszukuje się spośród pozostałych neuronów. Maksymalną wartość potencjału ogranicza się na poziomie równym 1.

Algorytm gazu neuronowego

Algorytm ten działa w identyczny sposób jak algorytm Kohonena z wyjątkiem tego, że neurony są sortowane według odległości do wektora wejściowego i ich nowe wagi są wyznaczane na podstawie miejsca w szeregu.

Algorytm k-średnich

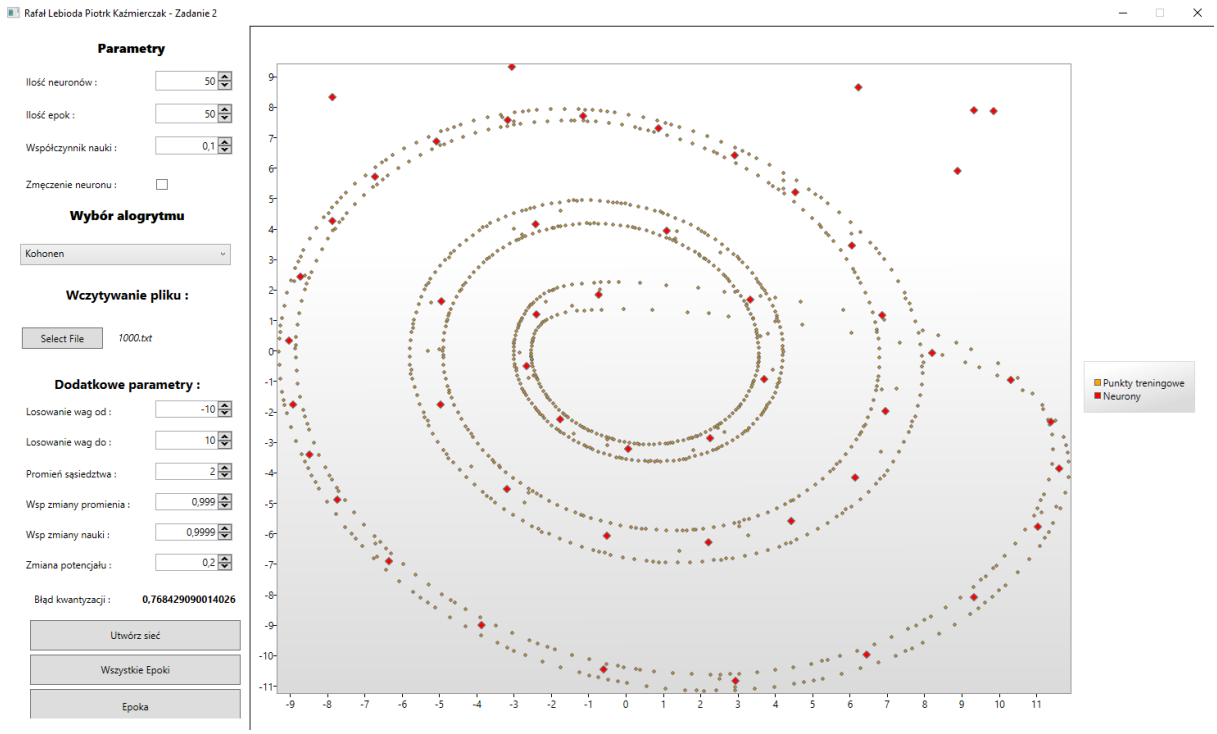
Algorytm k-średnich służy do podziału danych wejściowych na z góry założoną liczbę klas.

Algorytm można podzielić na następujące kroki :

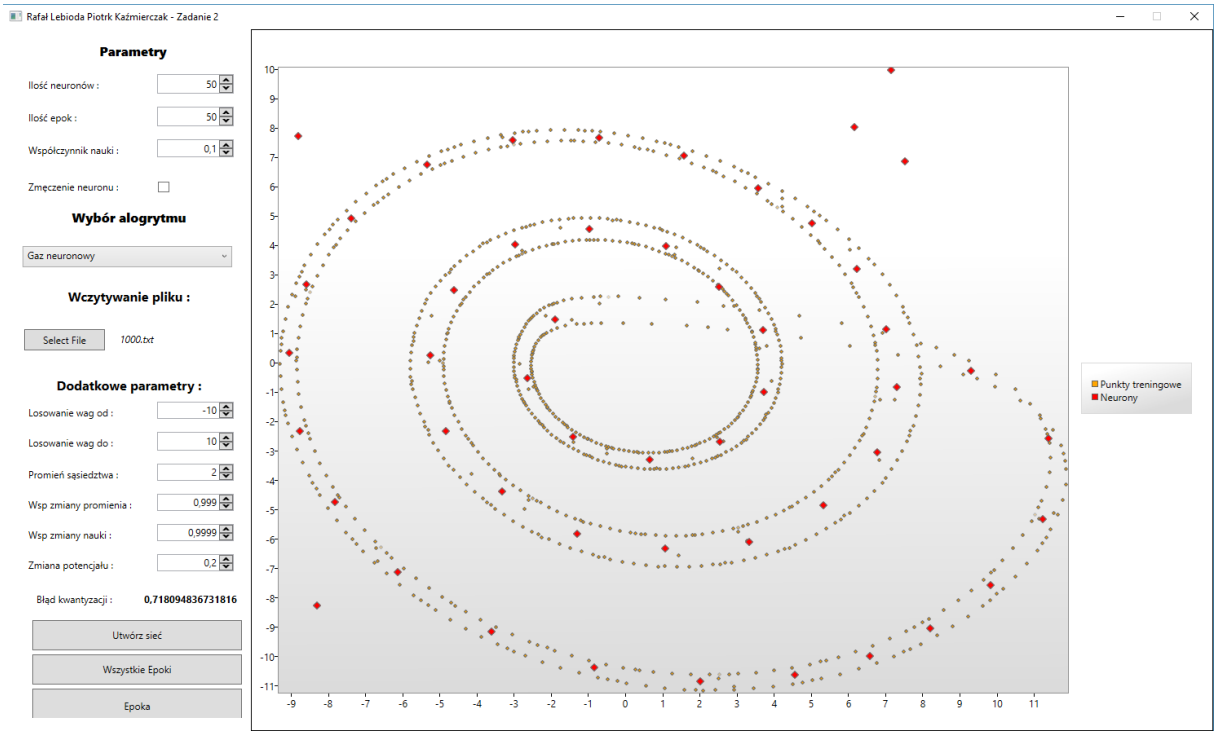
1. Pierwszym krok algorytmu polega na wylosowaniu współrzędnych neuronów. Te neurony będą określać grupy dla danych wejściowych. Współrzędne startowe punktów mogą wpływać na wyniki końcowy.
2. Dla każdego wektora wejściowego znajdujemy najbliższy neuron i przypisujemy go do niego
3. Zmieniamy wagi neuronów na średnią arytmetyczną współrzędnych punktów należących do jego grupy
4. Krok drugi i trzeci powtarzamy aż do osiągnięcia kryterium zbieżności, którym najczęściej jest stan w którym nie zmieniła się przynależność punktów do klas.

Część 1 – Algorytmy Kohonena i gazu neuronowego

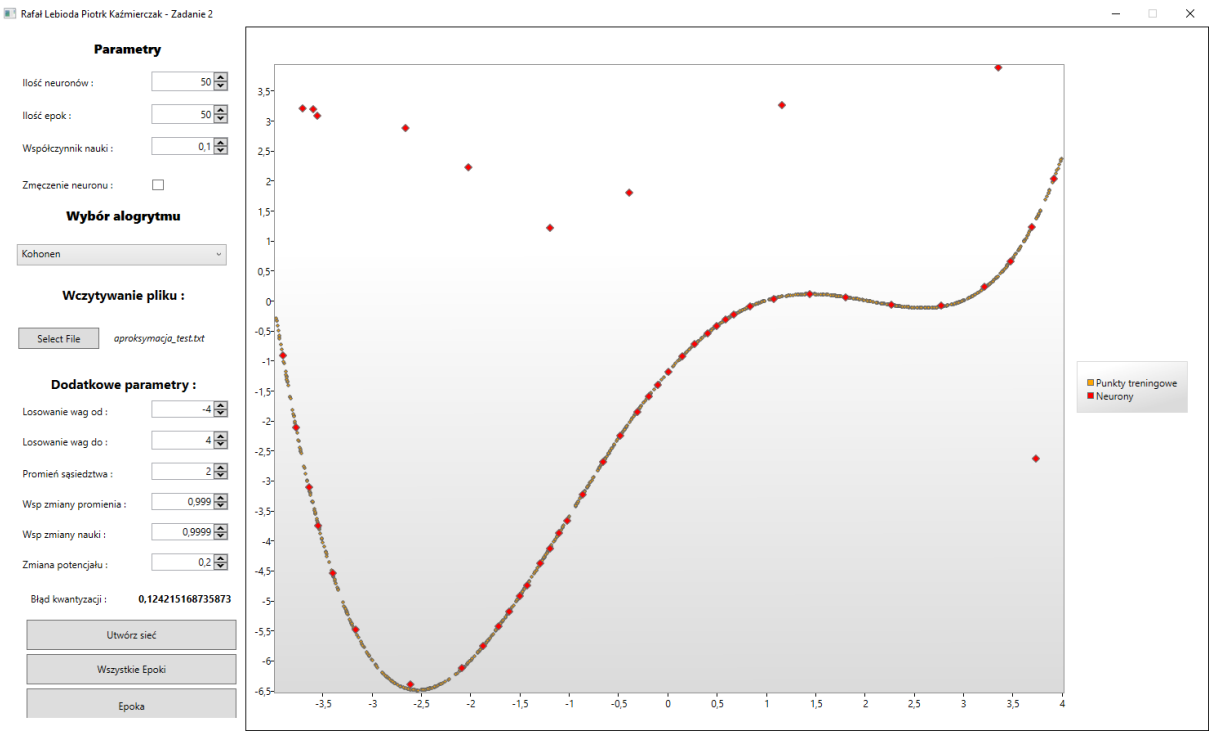
Testowaliśmy dane zadanie na dwóch zbiorach punktów różniących się od siebie kształtem w jaki układały się punkty. Dla obu zbiorów wyniki są zadowalające. Dla porównania prezentujemy poniżej zastosowanie tych samych dla obu przypadków dla algorytmu Kohonena jak i gazu neuronowego.



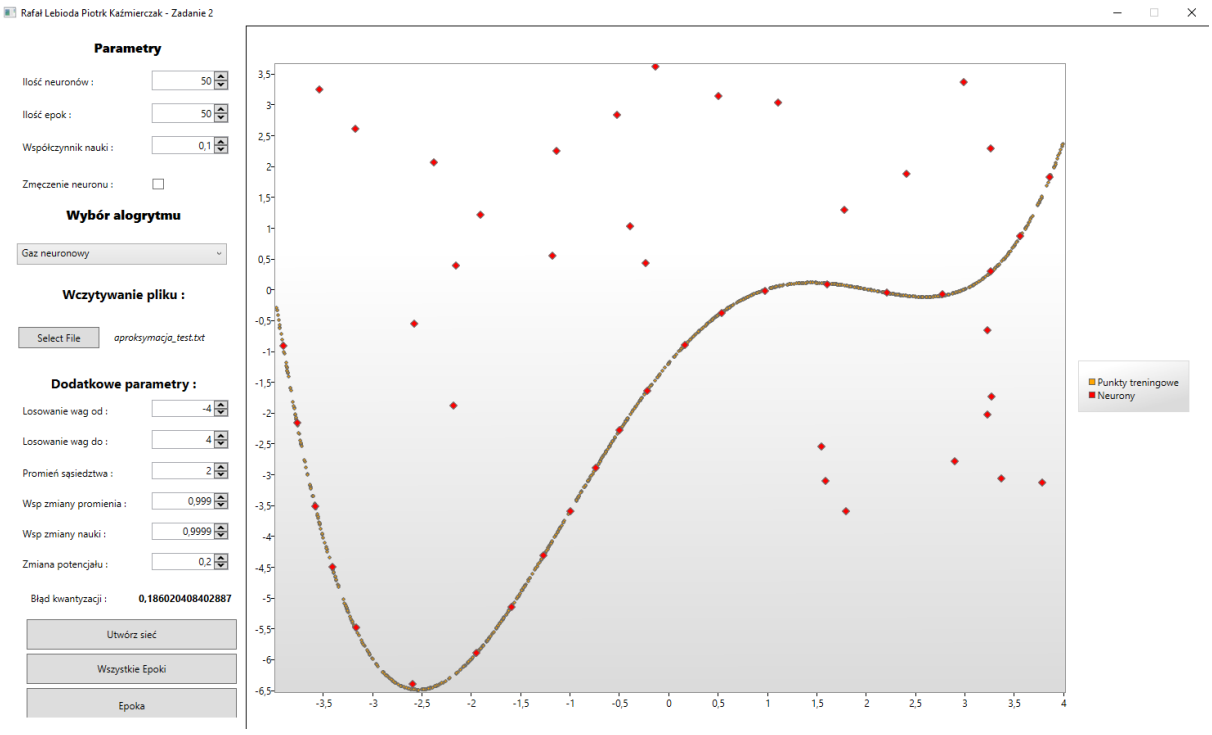
Wynik uczenia dla pierwszego zbioru – Algorytm Kohonena



Wynik uczenia dla pierwszego zbioru – Algorytm gazu Neuronowego



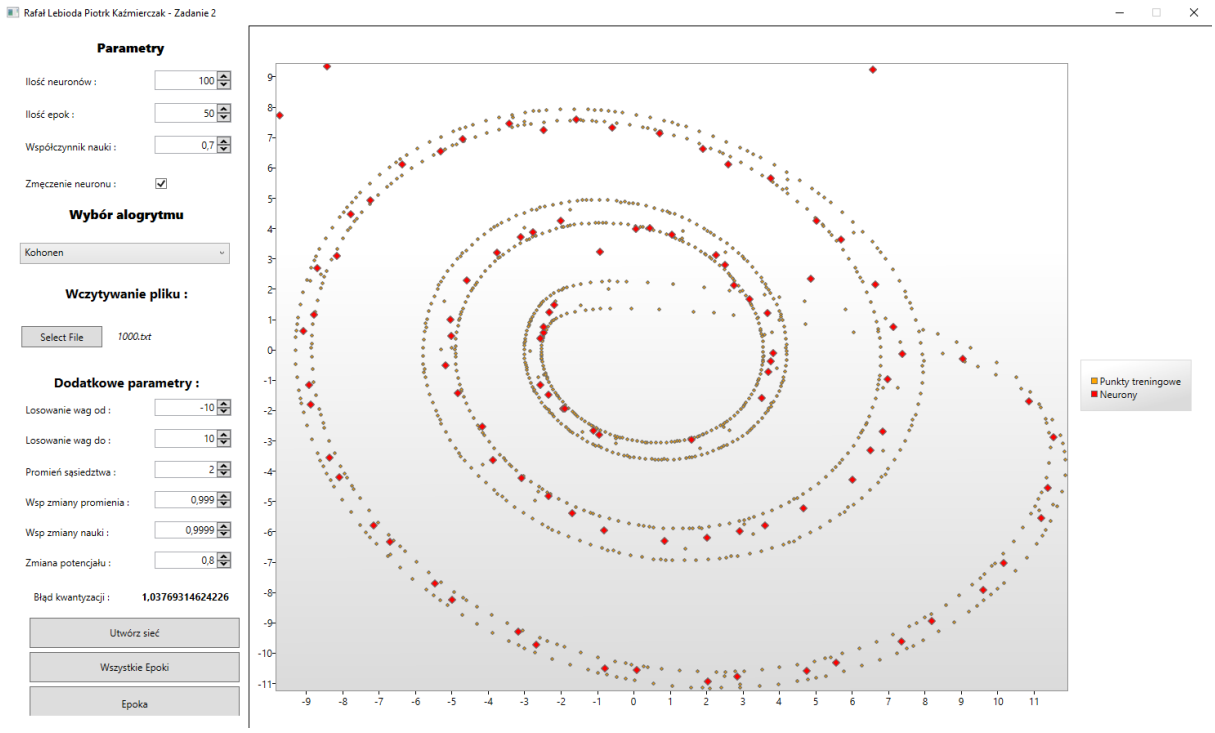
Wynik uczenia dla drugiego zbioru – Algorytm Kohonena



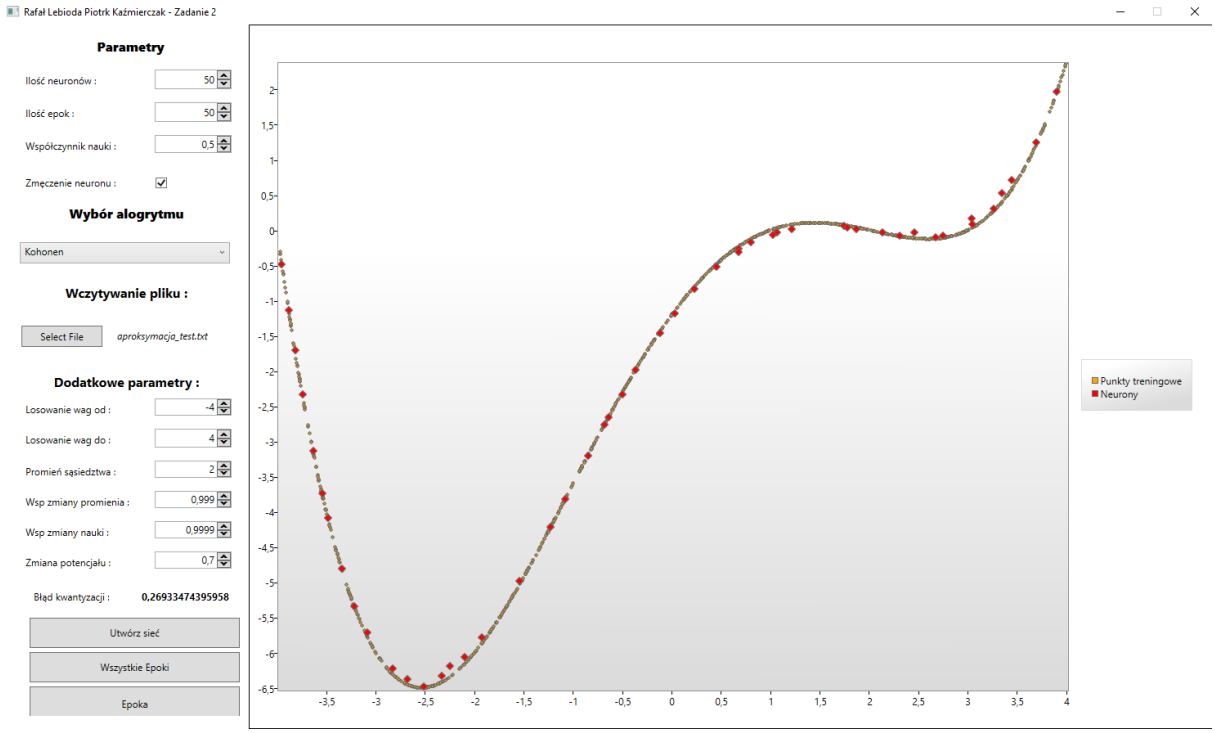
Wynik uczenia dla drugiego zbioru – Algorytm gazu neuronowego

Za każdym razem położenie neuronów jest losowane z wybranego przedziału, jednakże można zaobserwować, że pewnym problemem są martwe neurony tzn. takie które są cały czas nieaktywne i nie adaptują swoich wag. Wprowadzony mechanizm zmęczenia neuronów wraz z ustawionym poprawnie współczynnikiem potencjału powoduje, że problem ten zostaje wyeliminowany.

Oczywiście może się zdarzyć, że jakiś neuron pomimo zastosowanego mechanizmu pozostanie nieaktywny, ale będzie to skutkiem niefortunnego wylosowania położenia. Dla przykładu wyniki po zastosowaniu mechanizmu zmęczenia.



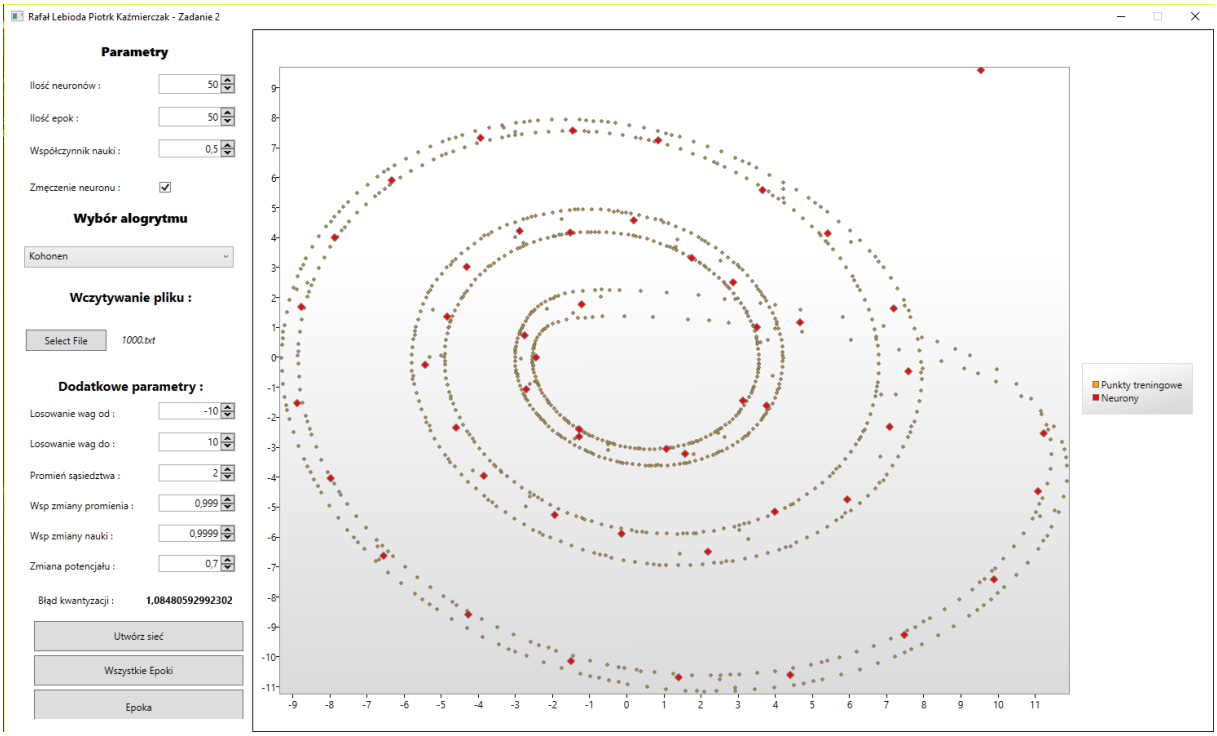
Zastosowanie mechanizmu męczenia neuronów – 3 martwe neurony ze 100



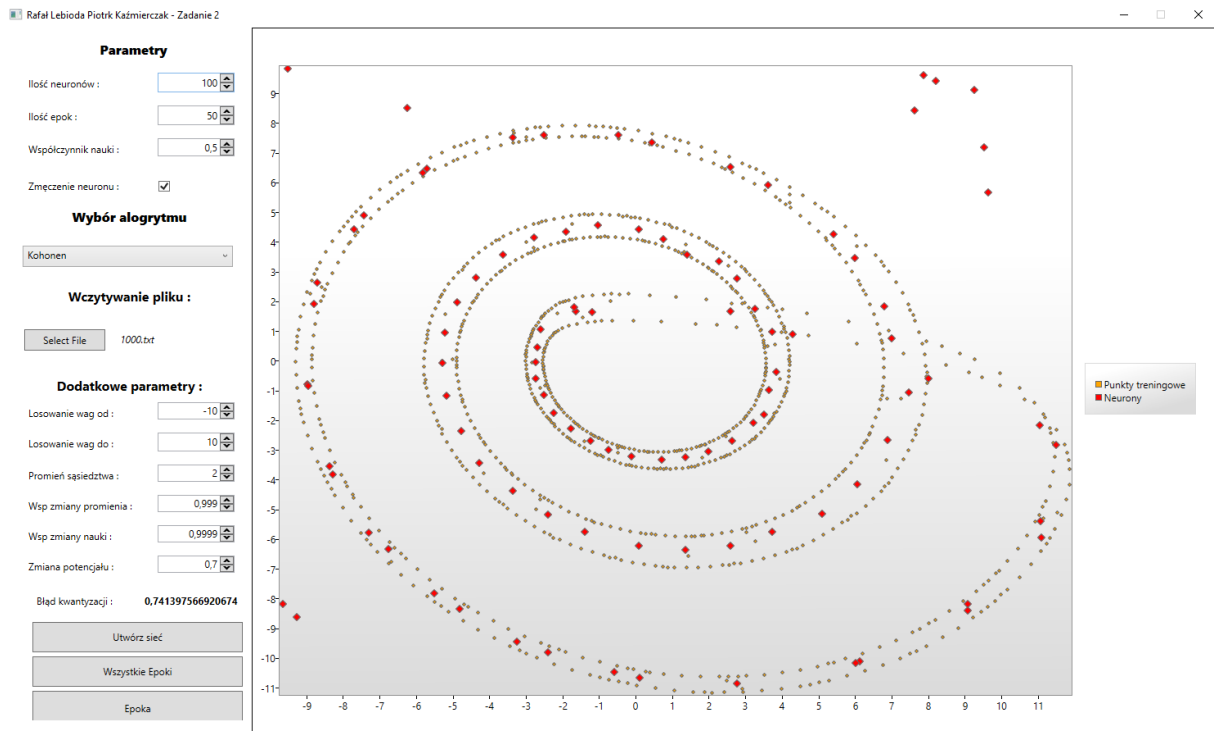
Zastosowanie mechanizmu męczenia neuronów – 0 martwych neuronów z 50

Jak widać na powyższych obrazkach efekt jest zadowalający i eliminuje jeśli nie całkowicie to w większym stopniu efekt martwych neuronów.

W programie występuje wiele współczynników które mają wpływ na wynik nauki jednakże w naszych testach zazwyczaj stosowaliśmy wartości które wydają nam się być najlepsze. Najistotniejszym parametrem jest współczynnik nauki i ilość neuronów stosowana do nauki. Im więcej neuronów tym lepsze odwzorowanie zbioru punktów co za tym idzie błąd kwantyzacji się zmniejsza co widać na poniższych obrazach



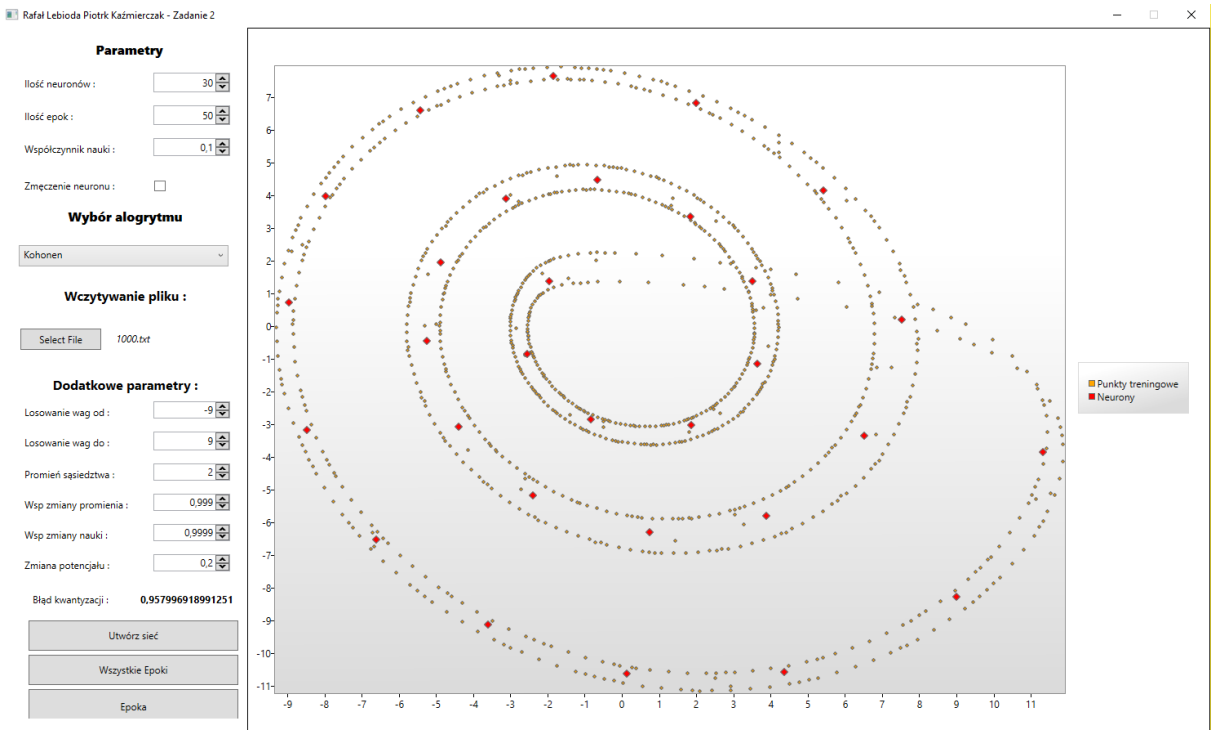
Użycie 50 neuronów – błąd kwantyzacji równy 1,08



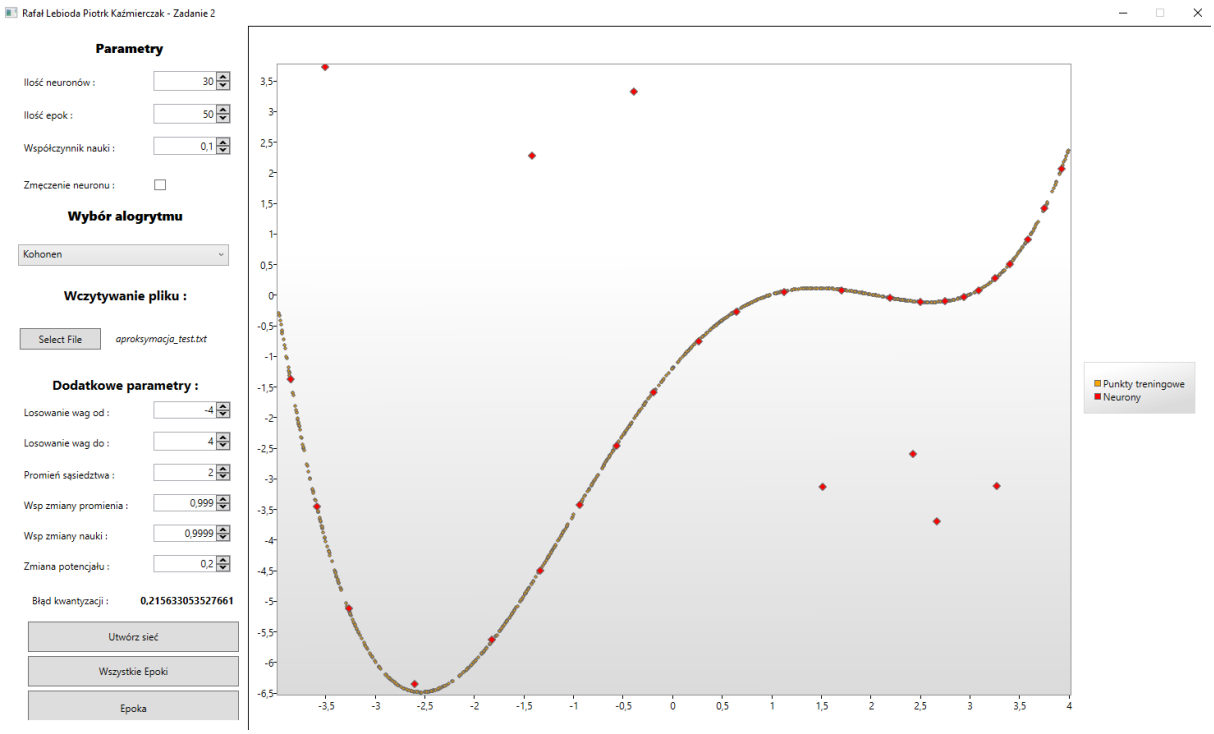
Użycie 100 neuronów – błąd kwantyzacji równy 0,74

Część 2 – Algorytmy k-średnich

Jest to identyczne zadanie jak poprzednie z tym, że został zastosowany algorytm k-średnich. Poniżej porównanie działania programu dla dwóch różnych zbiorów.



Wynik uczenia dla pierwszego zbioru – Algorytm k-średnich



Wynik uczenia dla drugiego zbioru – Algorytm k-średnich

Wnioski

Zaimplementowane metody uczenia się bez nadzoru wydają się być skuteczne i spełniają swoje zadanie jakim jest odwzorowanie zbioru punktów treningowych. Im więcej neuronów zastosujemy tym odwzorowanie jest lepsze. Należy jednak pamiętać, że celem jest użycie jak najmniejszej liczby neuronów. Istotne jest by współczynnik nauki oraz promień sąsiedztwa malały w czasie wtedy efekty nauki są najlepsze.

Źródła

- http://galaxy.agh.edu.pl/~vlsi/AI/koho_t/
- http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad4/w4.htm
- http://marcin.asia.w.interiowo.pl/kohonen/teoria.html