Informatyka, studia zaoczne, inż. III rok semestr VI

Inteligentna analiza danych **2015/2016**

prowadzący: dr inż. Bartłomiej Stasiak

Data oddania:_____ Ocena:_____

Marek Gadzalski 191422
Grzegorz Głąb 191425

Zadanie 3. Grupowanie i sieci SOM

1. CEL

Celem zadania jest implementacja trzech algorytmów grupowania danych:
- algorytmu k-średnich
oraz dwóch algorytmów samoorganizującej się mapy (SOM):
- algorytmu Kohonena
- algorytmu gazu neuronowego

2. WPROWADZENIE

2.1. Algorytm k-średnich

Algorytm k-średnich jest prostym algorytmem grupowania danych. Podstawą tego algorytmu są tzw. centroidy czyli środki grup. W pierwszym etapie pozycje centroid ustalane są losowo, zazwyczaj w zakresie odpowiadającym zakresowi danych wejściowych. Następnie każda dana ze zbiory wejściowego przypisywana jest najbliższej centroidzie (zgodnie z przyjętą metryką). Następnie każda centroida przemieszczana jest na środek zbioru danych przypisanych do tej centroidy. Taki cykl powtarzany z góry założoną liczbę razy lub do momenty gdy pozycje centroid będą ustabilizowane. Podstawową wadą tego algorytmu jest znaczny wpływ początkowego ustawienia centroid na proces grupowania. Przy większej liczbie centroid część nie posiada przypisanych im danych w związku z tym nie biorą one udziału w procesie adaptacji.

2.2. Sieć Kohonena

Sieci Kohonena zbudowane są z jednej warstwy neuronów, które reprezentują pewne podzbiory danych wejściowych. Początkowo wagi neuronów (których liczba odpowiada liczbie wymiarów przestrzeni danych) ustalane są losowo. W kolejnych iteracjach algorytmu neurony współzawodniczą ze sobą o reprezentowanie wektorów wejściowych, wygrywa ten wektor, który znajduje się najbliżej wektora wejściowego (zgodnie z przyjętą metryką). Następnie wagi neuronu aktualizowane tak aby zbliżyć się do danego wektora wejściowego. Wyróżniamy dwa zasadnicze warianty algorytmu WTA (ang. *winner takes all*) oraz WTM (ang. *winner takes most*). W pierwszym wariancie jedynie wagi neuronu zwycięskiego są aktualizowane i w dużym stopniu przypomina on algorytm k-średnich ze wszystkim jego wadami, w związku z tym jest on rzadko wykorzystywany.
Drugi wariant zakłada, że oprócz neuronu zwycięskiego równiej jego sąsiedzi podlegają adaptacji. Sąsiedztwo może być ustalane zasadniczo na dwa sposoby:

- wszystkie neurony w założonym promieniu sąsiedztwa λ od zwycięscy (łącznie ze zwycięzcą) adaptowane są w taki sam sposób, mówimy wtedy o sąsiedztwie prostokątnym.

$$G(i,x) = \begin{cases} 1, dla\ d(i,j) \leq \lambda \\ 0, dla\ pozostałych \end{cases}$$

gdzie j to numer zwycięskiego neuronu
d(i,j) – odległość i-tego neuronu od zwycięscy

- wszystkie neurony są adaptowane, przy czym stopień adaptacji zależy od odległości od zwycięzcy. Najczęściej wykorzystuję się funkcję gaussowską (mówimy wtedy o sąsiedztwie gaussowskim)

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

Dla zwycięzcy wartość tej funkcji wynosi 1, natomiast dla pozostałych neuronów wartość funkcji jest w przedziale (0,1).

Wartość parametru λ może ulegać zmianie wraz z postępem adaptacji. Na przykład wykładniczo:

$$\lambda(k) = \lambda_{max}\left(\frac{\lambda_{min}}{\lambda_{max}}\right)^{\left(\frac{k}{k_{max}}\right)}$$

gdzie k to liczba iteracji.

W procesie adaptacji wagi neuronów modyfikowane są wg następującego wzoru:

$$w_i(k+1) = w_i + \eta(k)G(i,x)[x - w_i(k)]$$

gdzie: w_i - waga i-tego neuronu
η - współczynnik kroku
G(i,x) – funkcja sąsiedztwa
x – dana wejściowa
k - iteracja

Również współczynnik kroku η może być modyfikowany w trakcie procesu adaptacji. W naszym programie zastosowano zmianę wykładniczą podobną jak dla parametru λ:

$$\eta(k) = \eta_{max}\left(\frac{\eta_{min}}{\eta_{max}}\right)^{\left(\frac{k}{k_{max}}\right)}$$

W sieciach Kohonena pojawia się problem tzw. martwych neuronów, czyli takich, które na początku adaptacji znalazły się daleko od danych i nie są w ogóle adaptowane. Istnieje szereg rozwiązań eliminacji tego problemu. Zastosowane przez nas rozwiązanie polega na zliczaniu liczby zwycięstw każdego neuronu. Liczba ta jest następnie uwzględniania przy obliczaniu odległości od zwycięscy:

$$\hat{d}(i,j) = W_i d(i,j)$$

gdzie W_i to liczba zwycięstw i-tego neuronu + 0,1.
j – neuron zwycięski

Zmodyfikowana odległość jest następnie przekazywana do funkcji sąsiedztwa, co sprawia, że silnie aktywowane neurony są „spowalniane” (odległość jest od innych zwycięzców jest zwielokrotniona przez liczbę zwycięstw), a neurony które nie wygrały ani razu będą dla traktowane jakby znajdowały się znacznie bliżej od zwycięzcy niż w rzeczywistości (wartość bazowa W wynosi dla nich 0.1). Modyfikacja ta stosowana jest tylko na początku procesu adaptacji (w naszym programie przez 5% pierwszych iteracji). Po tym okresie wszystkie neurony są zlokalizowane w pobliżu danych i dalsza nauka może przebiegać w sposób normalny.

2.3. Algorytm gazu neuronowego

Algorytm gazu neuronowego różni się od sieci Kohonena przede wszystkim funkcją sąsiedztwa. Dla każdego wektora wejściowego ustalany jest zwycięzca, następne wszystkie pozostałe neurony szeregowane są względem odległości od zwycięscy. Funkcja sąsiedztwa ma postać:

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

gdzie m(i) jest pozycją i-tego neurony na liście odległości (dla zwycięscy m(i) = 0)

Takie podejście sprawia, że odległość od neuronu zwycięskiego ma mniejszy wpływ na aktywacje neuronów i w znacznym stopniu to zapobiega pojawianiu się martwych neuronów.

Modyfikacja wag postępuje w sposób analogiczny od sieci Kohonena:

$$w_i(k+1) = w_i + \eta(k)G(i,x)[x - w_i(k)]$$

Podobnie jak w przypadku sieci Kohonena wartości parametrów λ i η ulegają zmianie w trakcie adaptacji, przy czym algorytm gazu neuronowego jest znacznie bardziej wrażliwy na zmiany tych parametrów co sprawia, że muszą one zostać precyzyjniej dobrane. Nie ma za to konieczności wprowadzania poprawki zapobiegającej pojawianiu się martwych neuronów.

3. IMPLEMENTACJA

Aplikacja została napisana w języku Java z wykorzystaniem graficznego interfejsu Swing. Kod programu został podzielony na funkcjonalne pakiety

Pakiet algorithms – zawiera implementacje algorytmów grupowania
Pakiet gui – zawiera implementacje interfejsu graficznego
Pakiet gui.vornoi – zawiera implementacje tworzenia diagramów Woronoja. Wykorzystano gotową implementację: https://github.com/ptitfred/delaunay-java

Program umożliwia wybór spośród trzech wymienionych we wstępie algorytmów. Dla każdego z nich można wybrać liczbę neuronów/centroid oraz liczbę iteracji.
Dodatkowo dla algorytmów SOM można wybrać minimalne i maksymalne wartości parametrów η i λ, które ulegają wykładniczej zmianie w trakcie adaptacji.
Dla sieci Kohonena można wybrać pomiędzy prostokątną a gaussowską funkcją sąsiedztwa.

Odległości obliczane są w metryce euklidesowskiej
Proces nauki może być prowadzony ręcznie iteracja po iteracji bądź w sposób ciągły, który umożliwia obserwację na bieżąco zmian położeń centroid/neuronów.
Dla wszystkich algorytmów błąd grupowania liczony jest w ten sam sposób:

$$E = \frac{1}{N}\sum_{p=1}^{N}\left\|x_p - w_{j(p)}\right\|^2,$$

gdzie N – liczba wektorów wejściowych
x_p – wektor wejściowy
$w_{j(p)}$ – numer neuronu zwycięskiego dla wektora wejściowego x_p

4. MATERIAŁY

4.1. Dane

Wszystkie doświadczenia zostały przeprowadzone na dwóch plikach z danymi:

attract.txt oraz attract_small.txt

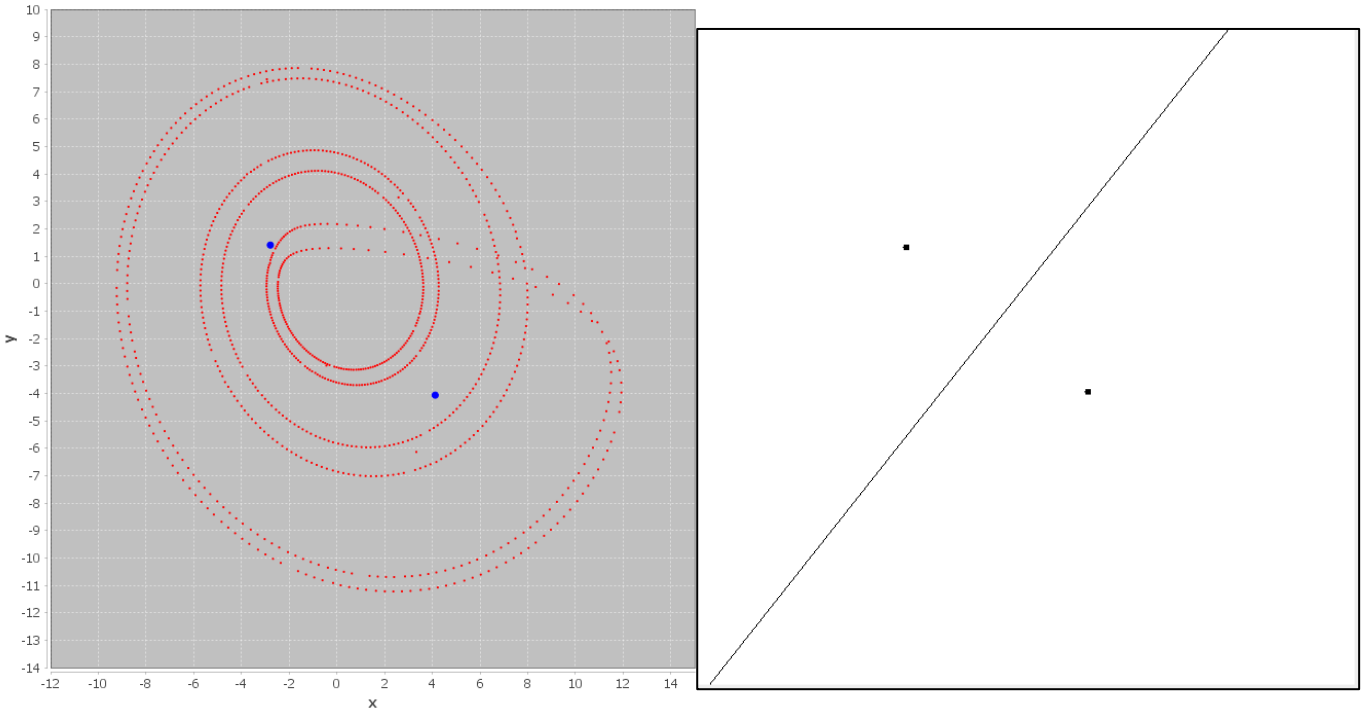
5. WYNIKI I WNIOSKI

Wszystkie prezentowane doświadczenia były prowadzone przez 100 iteracji

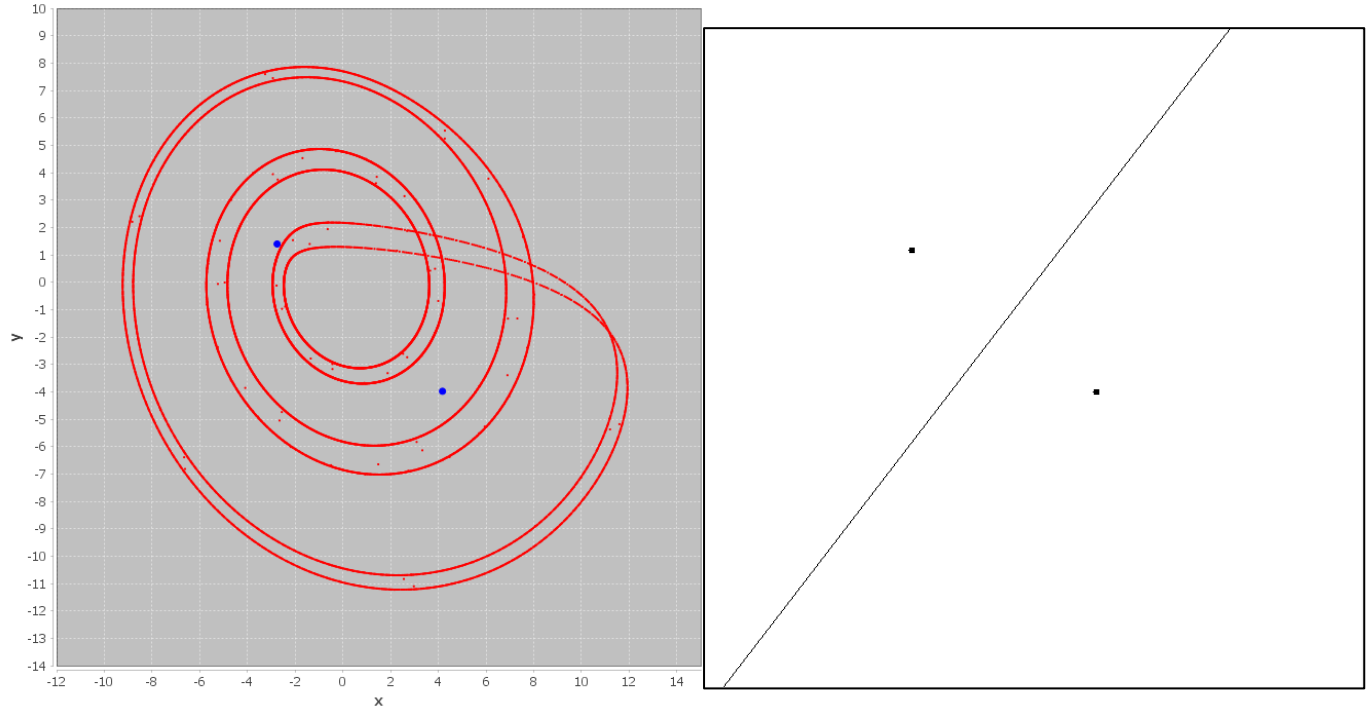
5.1. Algorytm k-średnich

Badanie skuteczności grupowania danych algorytmem k-średnich przy liczbie centroid w przedziale 2 -10 oraz 30.

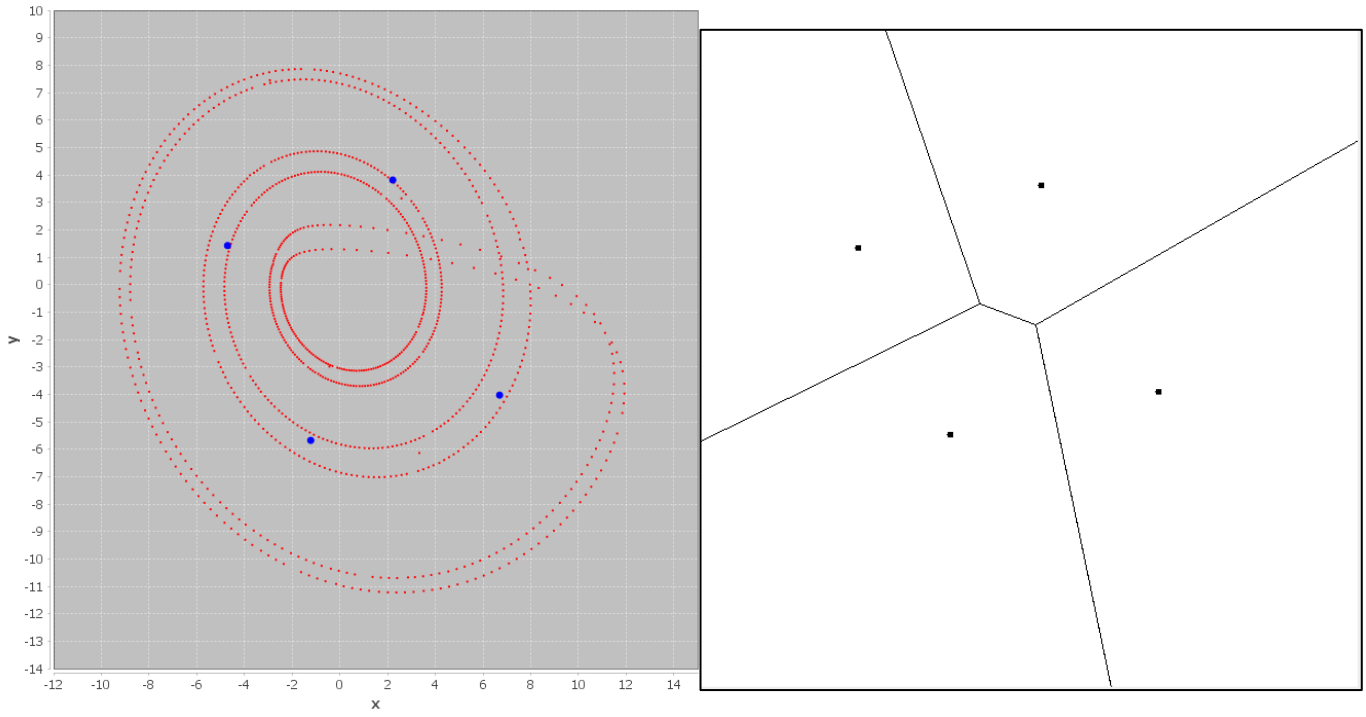
dane	liczba centroid	średni błąd na końcu adaptacji
attract_small.txt	**2**	**27,84**



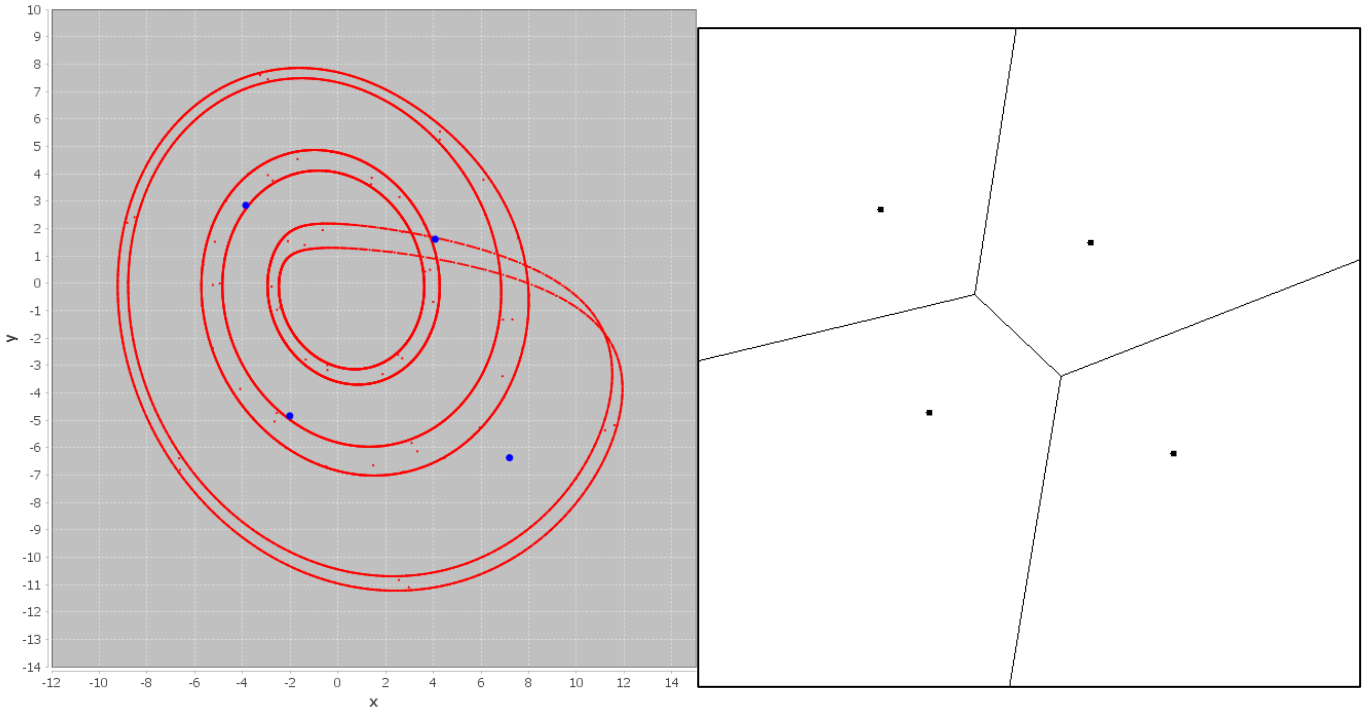
dane	liczba centroid	średni błąd na końcu adaptacji
attract.txt	**2**	**27,86**



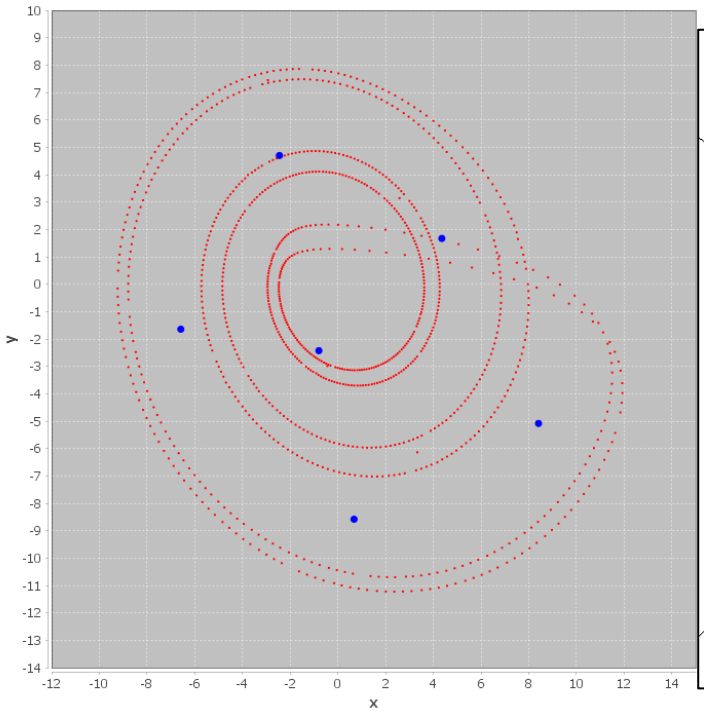
dane	liczba centroid	średni błąd na końcu adaptacji
attract_small.txt	4	14,47

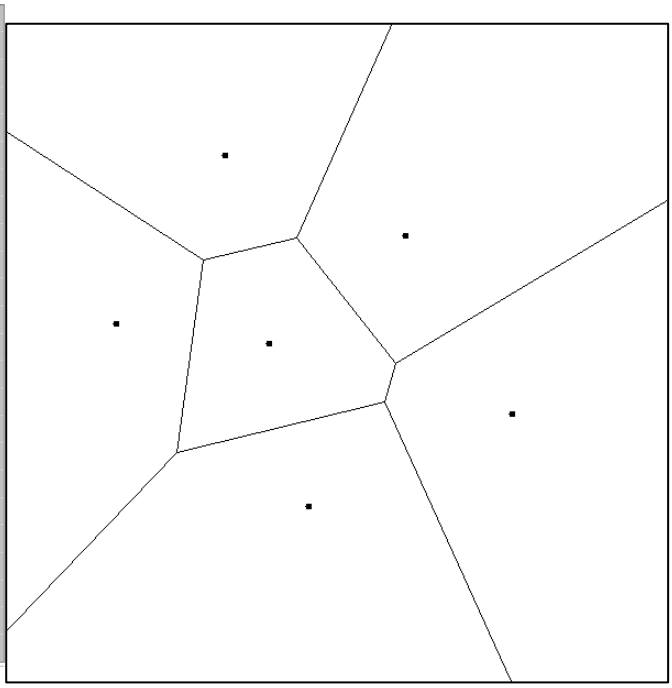


dane	liczba centroid	średni błąd na końcu adaptacji
attract.txt	4	14,44

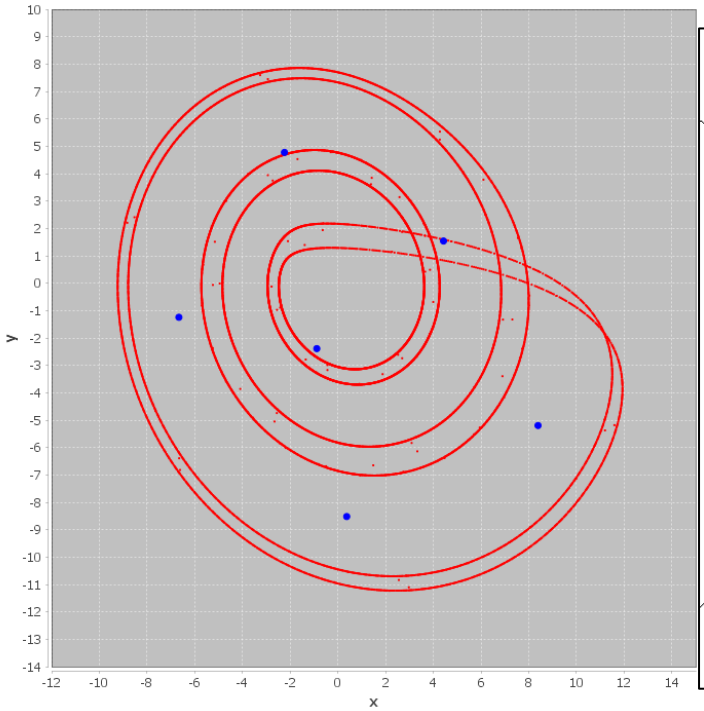


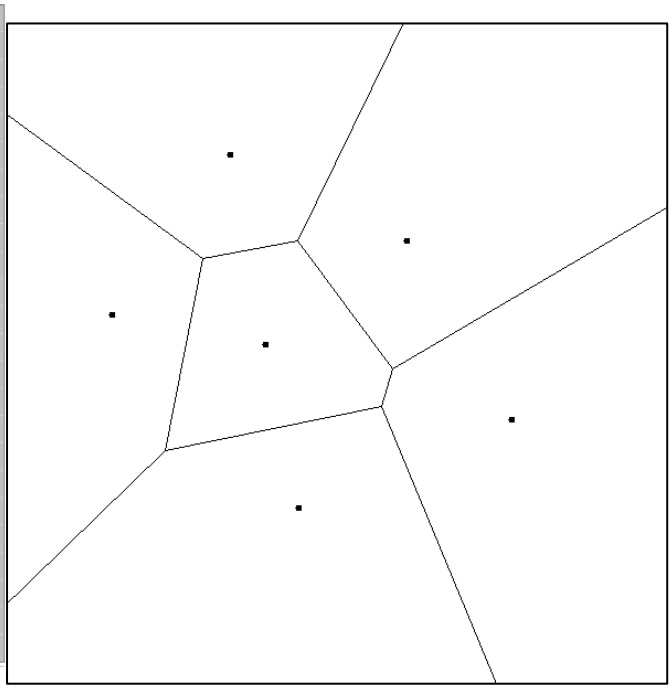
dane	liczba centroid	średni błąd na końcu adaptacji
attract_small.txt	6	9,99



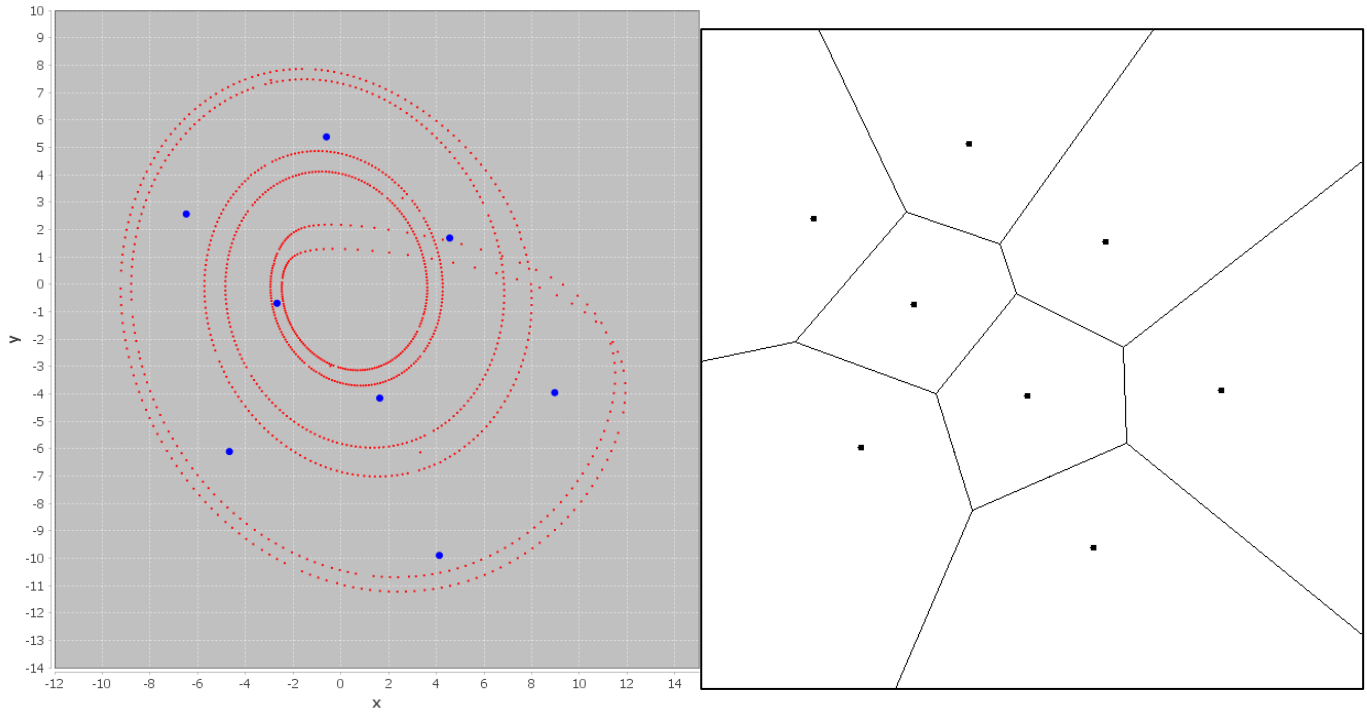


dane	liczba centroid	średni błąd na końcu adaptacji
attract.txt	6	9,85

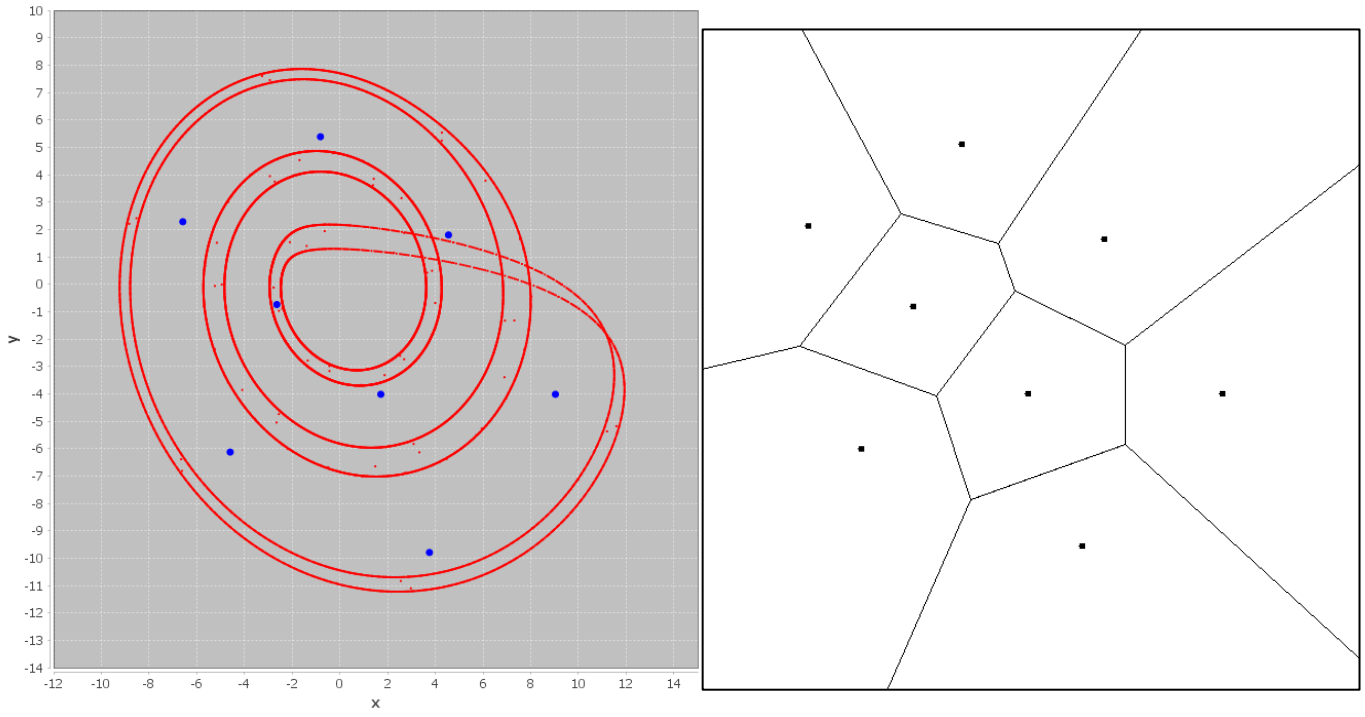




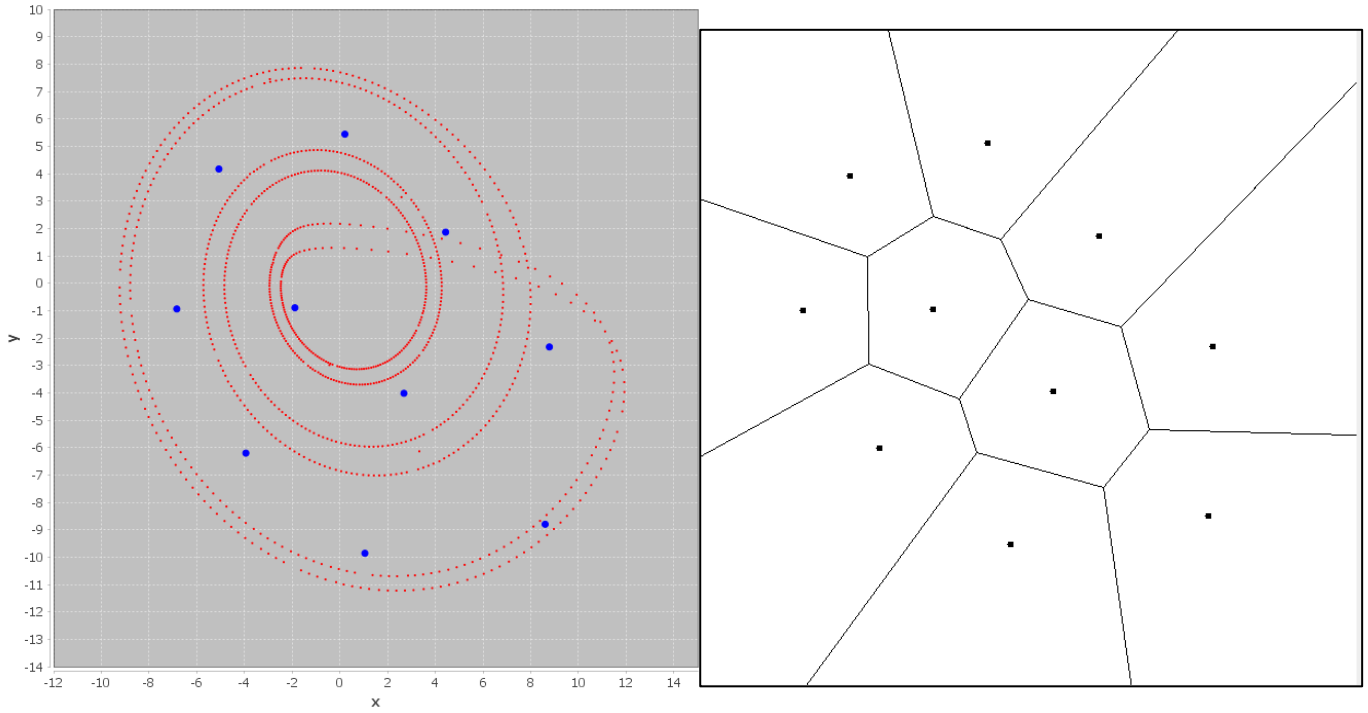
dane	liczba centroid	średni błąd na końcu adaptacji
attract_small.txt	8	7,07



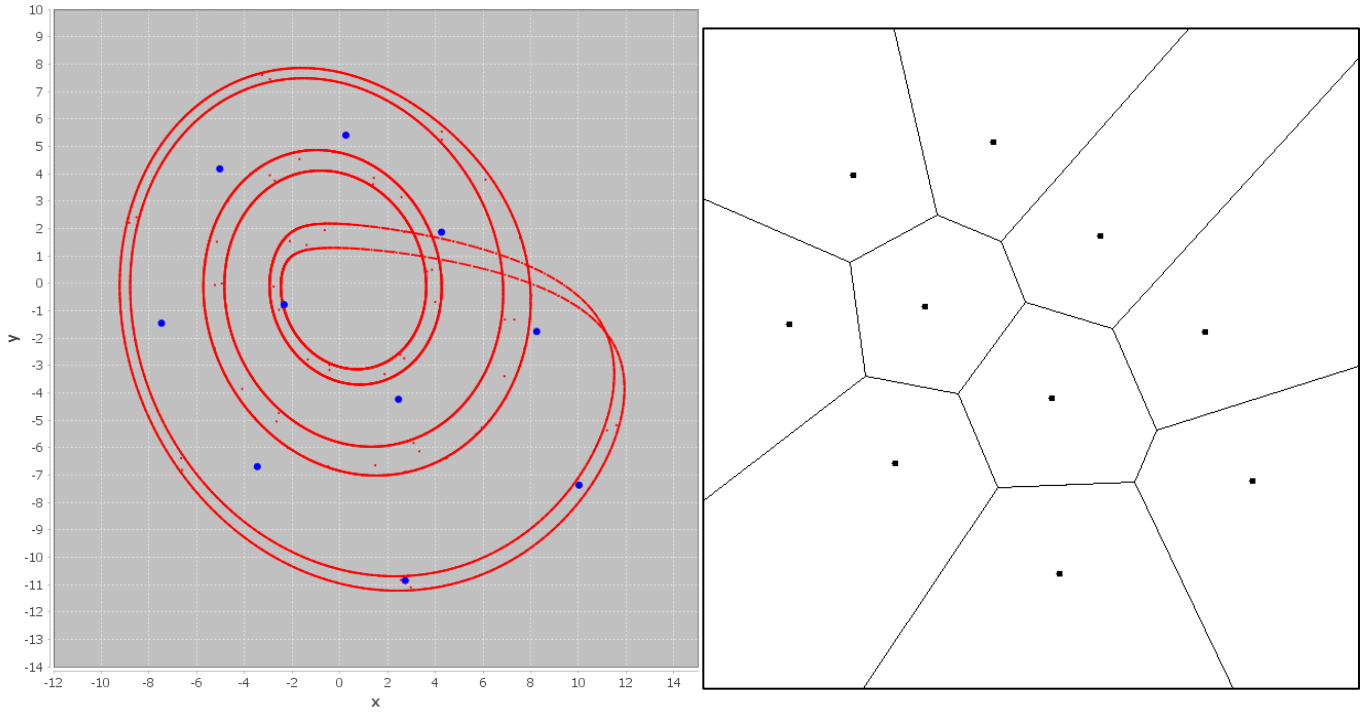
dane	liczba centroid	średni błąd na końcu adaptacji
attract.txt	8	7,10



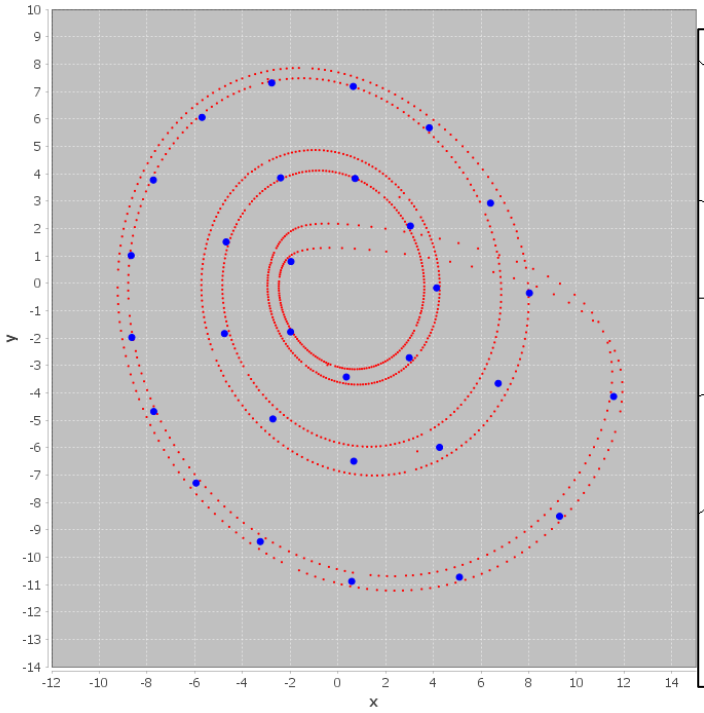
dane	liczba centroid	średni błąd na końcu adaptacji
attract_small.txt	10	5,85

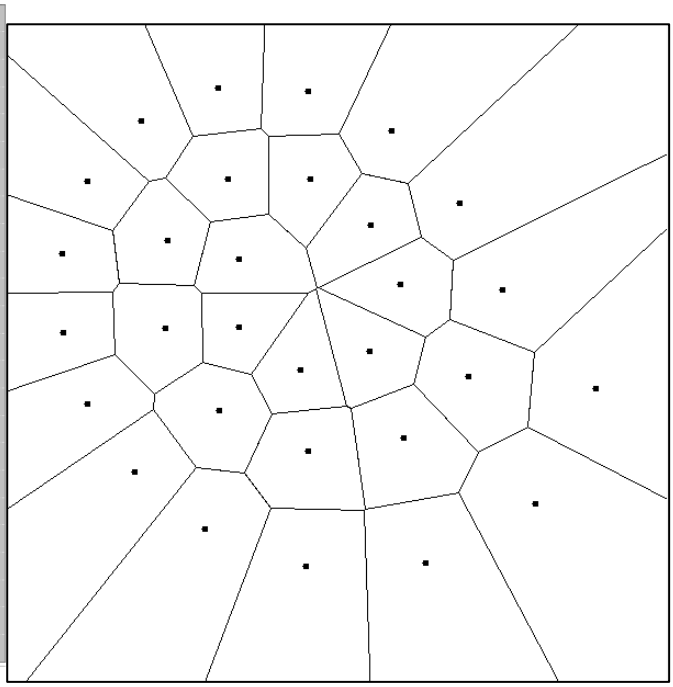


dane	liczba centroid	średni błąd na końcu adaptacji
attract.txt	10	5,57

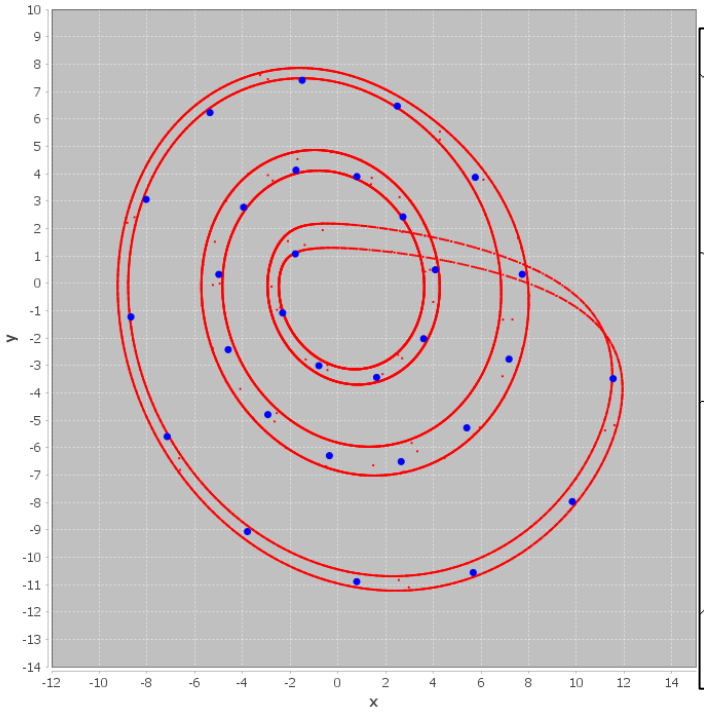


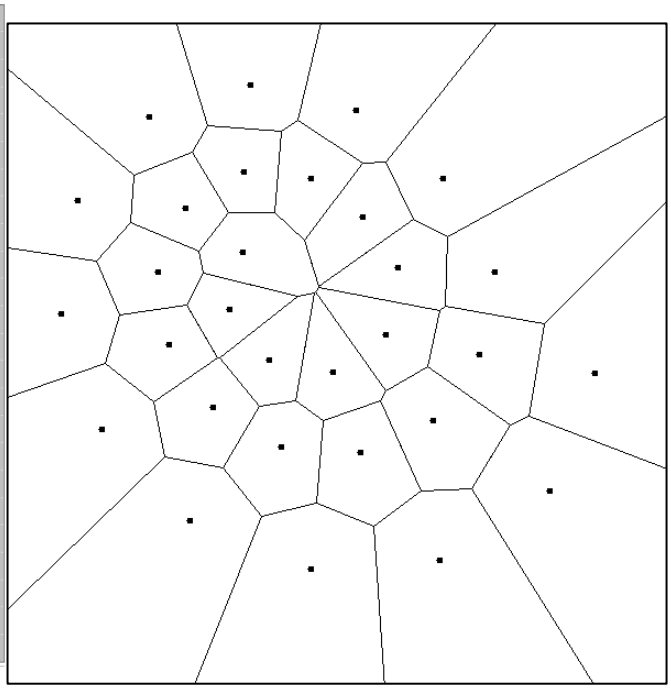
dane	liczba centroid	średni błąd na końcu adaptacji
attract_small.txt	30	1,20





dane	liczba centroid	średni błąd na końcu adaptacji
attract.txt	30	1,21



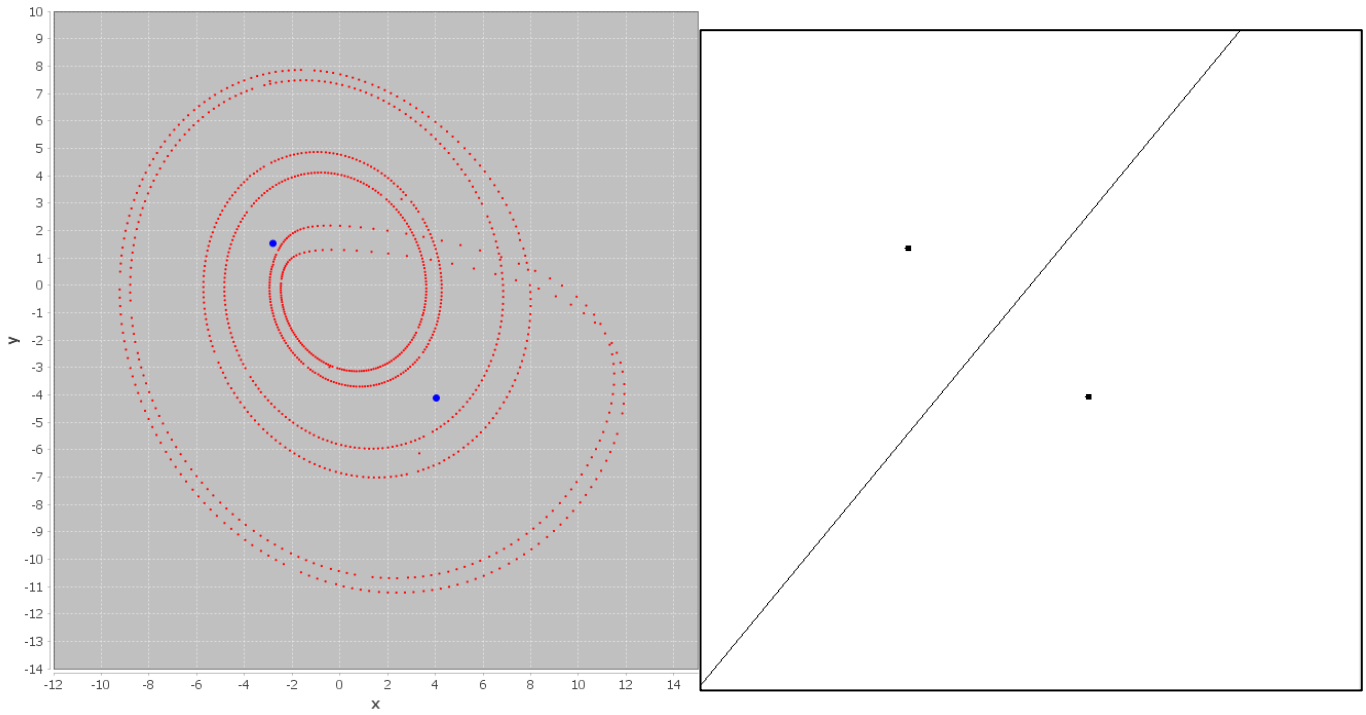


5.2. Sieci SOM

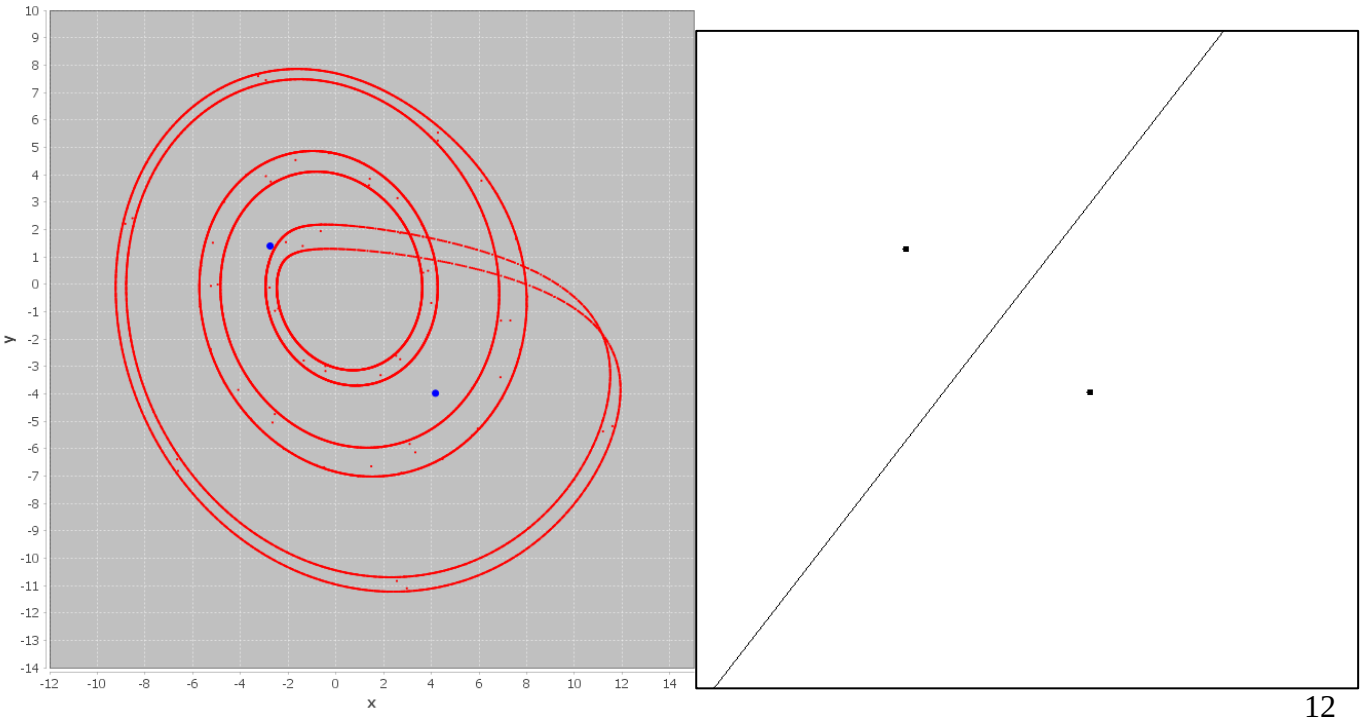
Badanie skuteczności grupowania danych sieciami SOM przy liczbie neuronów w przedziale 2 -10 oraz 30.

Sieci Kohonena

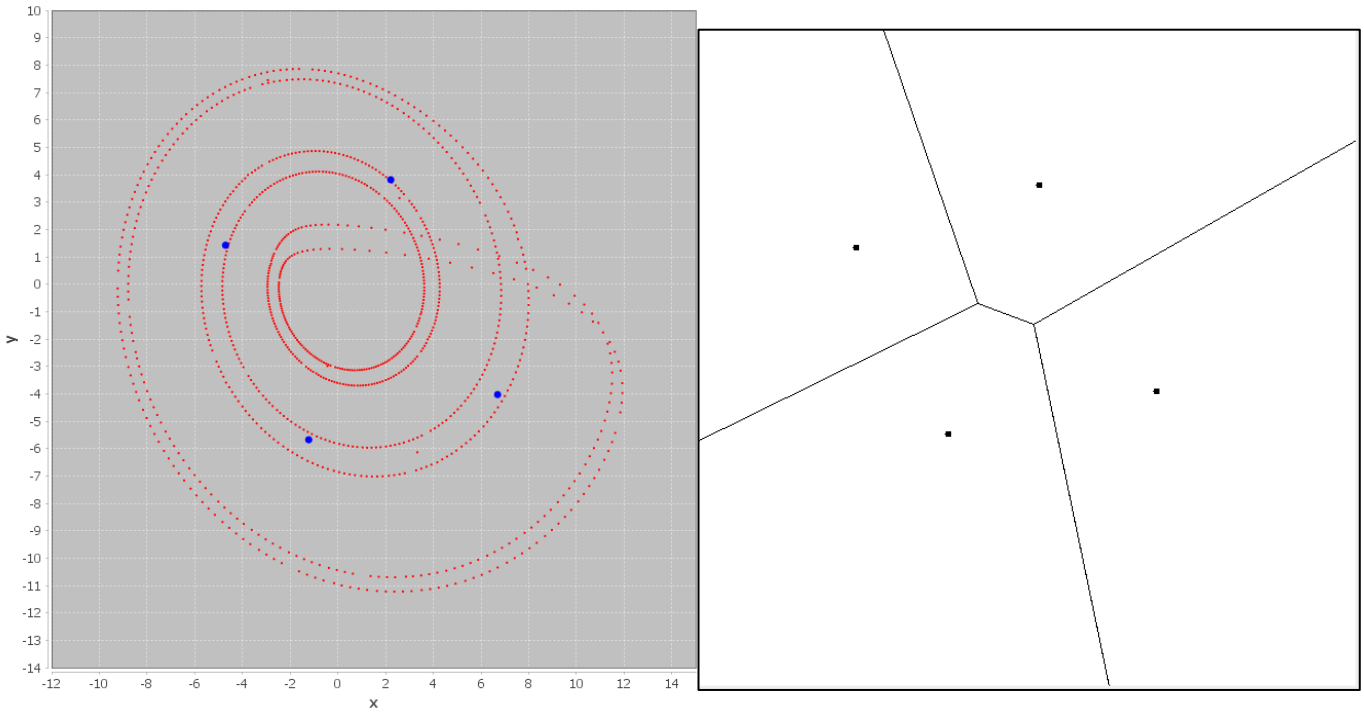
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract_small.txt	**2**	**28,12**	**0,5 – 0,05**	**0,1 – 0,01**



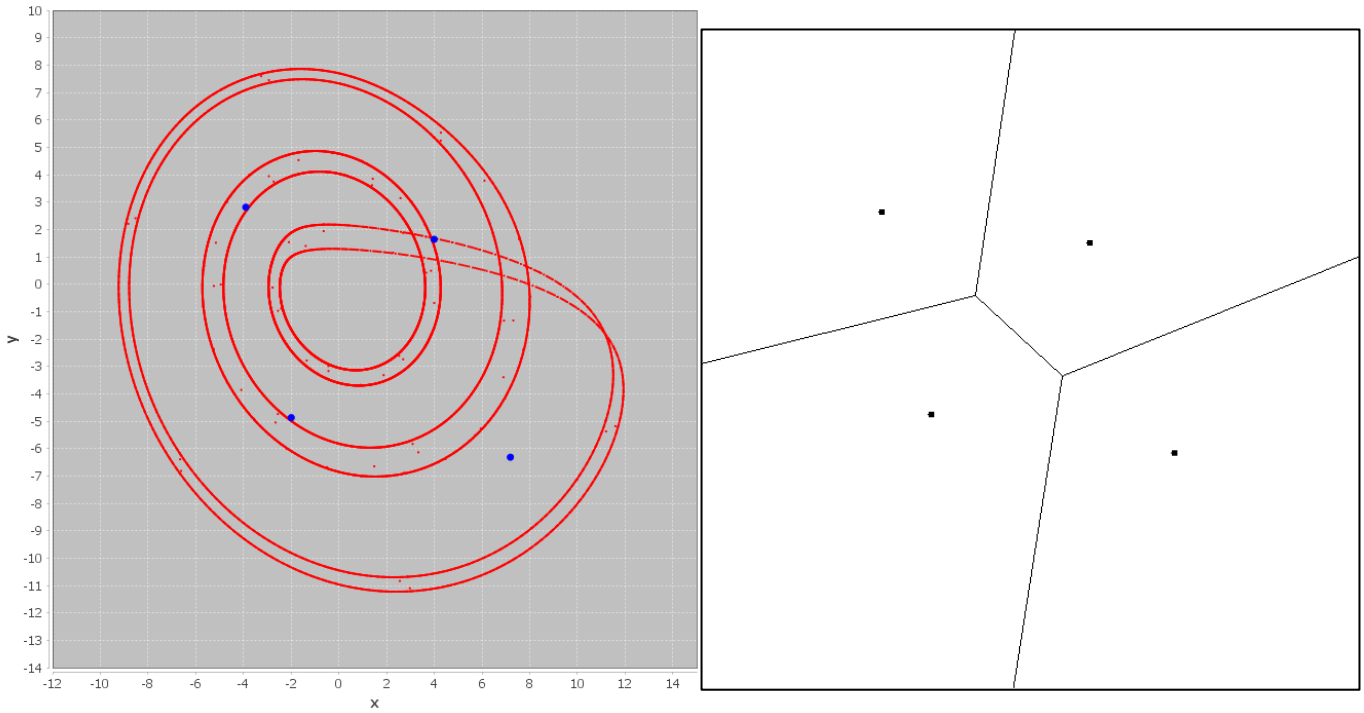
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract.txt	**2**	**28,14**	**0,5 – 0,05**	**0,1 – 0,01**



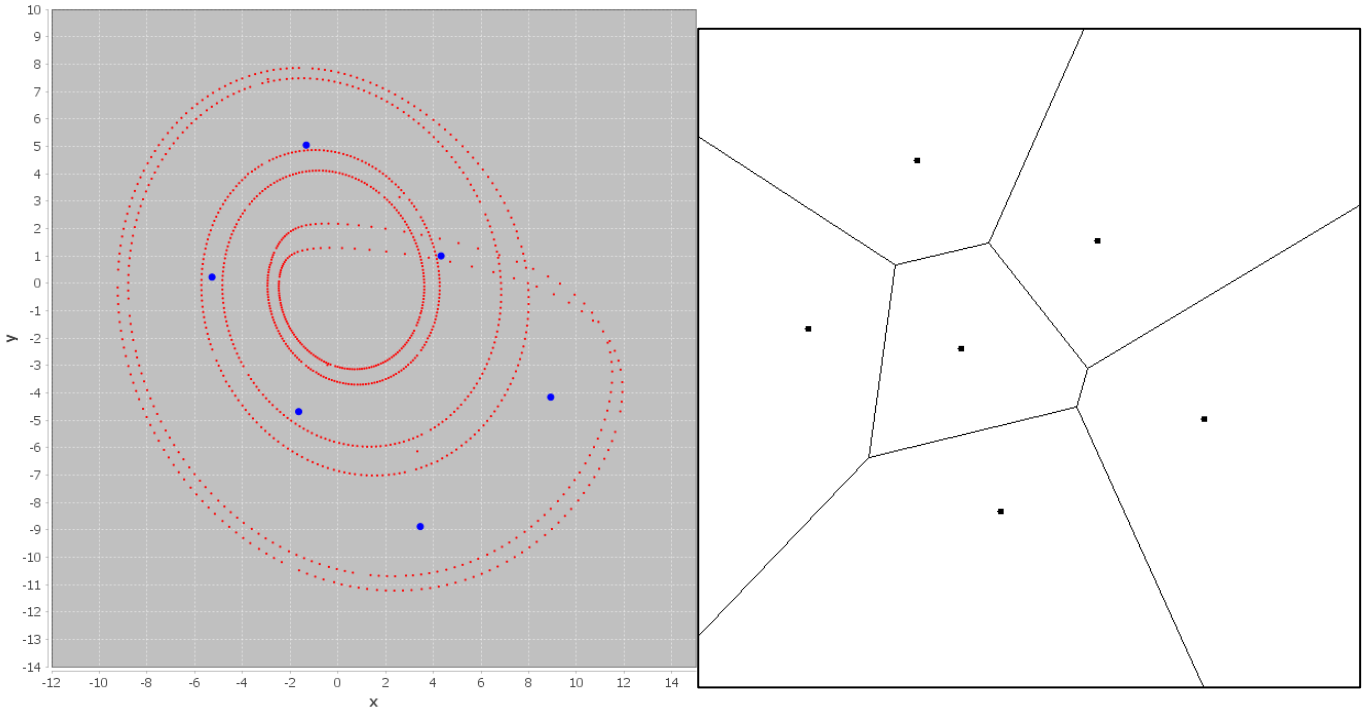
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract_small.txt	**4**	**14,72**	**0,5 – 0,05**	**0,1 – 0,01**



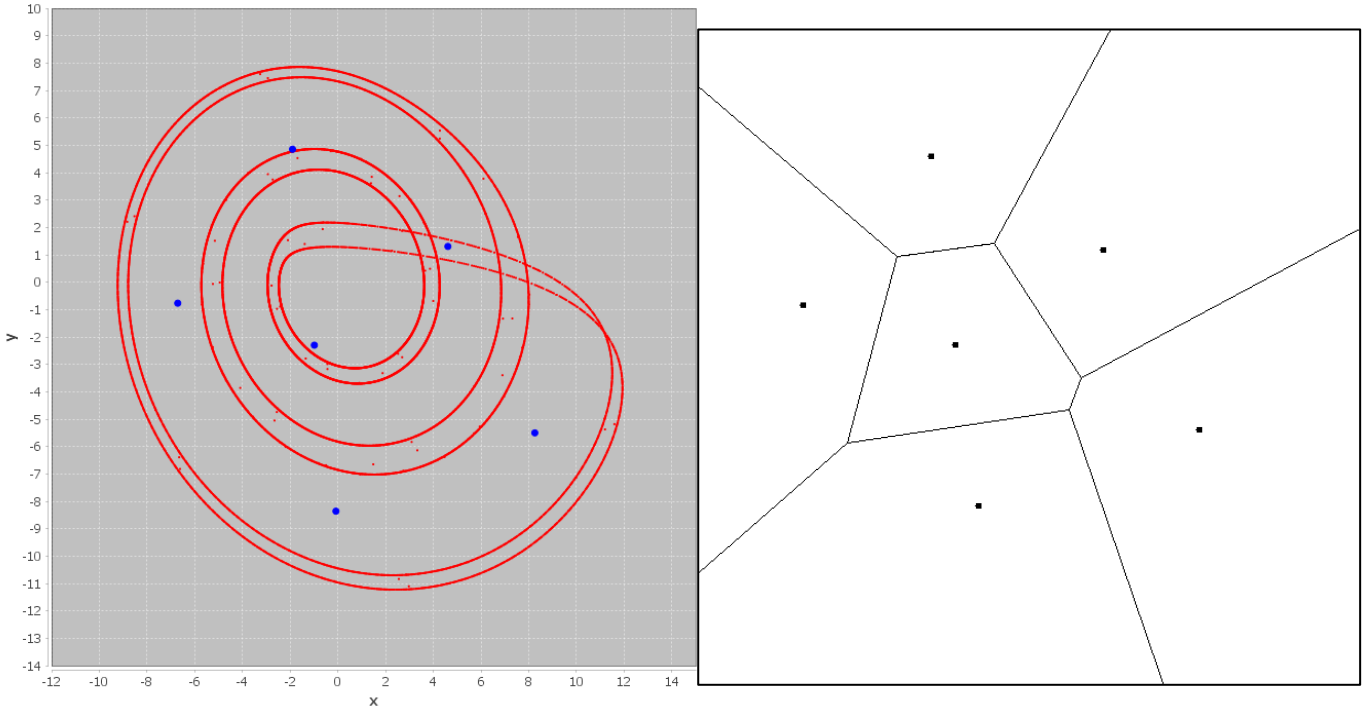
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract.txt	**4**	**14,59**	**0,5 – 0,05**	**0,1 – 0,01**



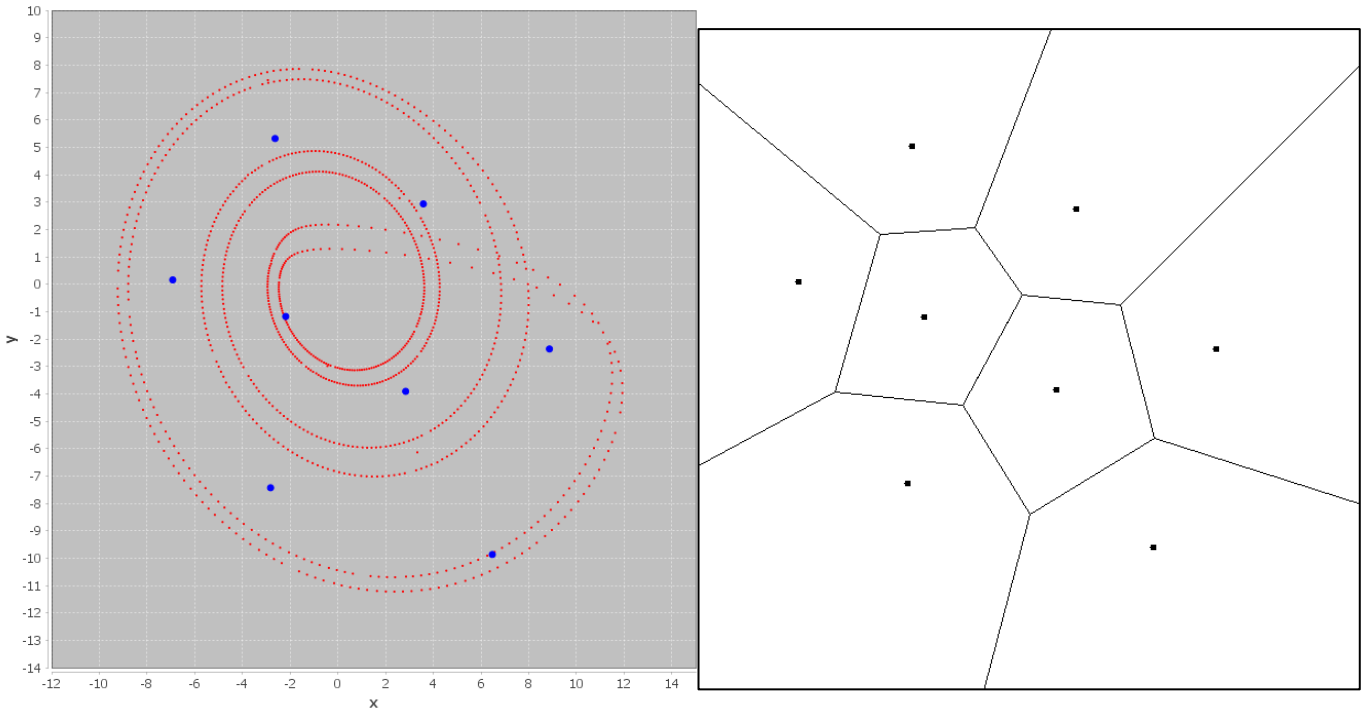
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract_small.txt	**6**	**9,93**	**0,3 – 0,05**	**0,2 – 0,001**



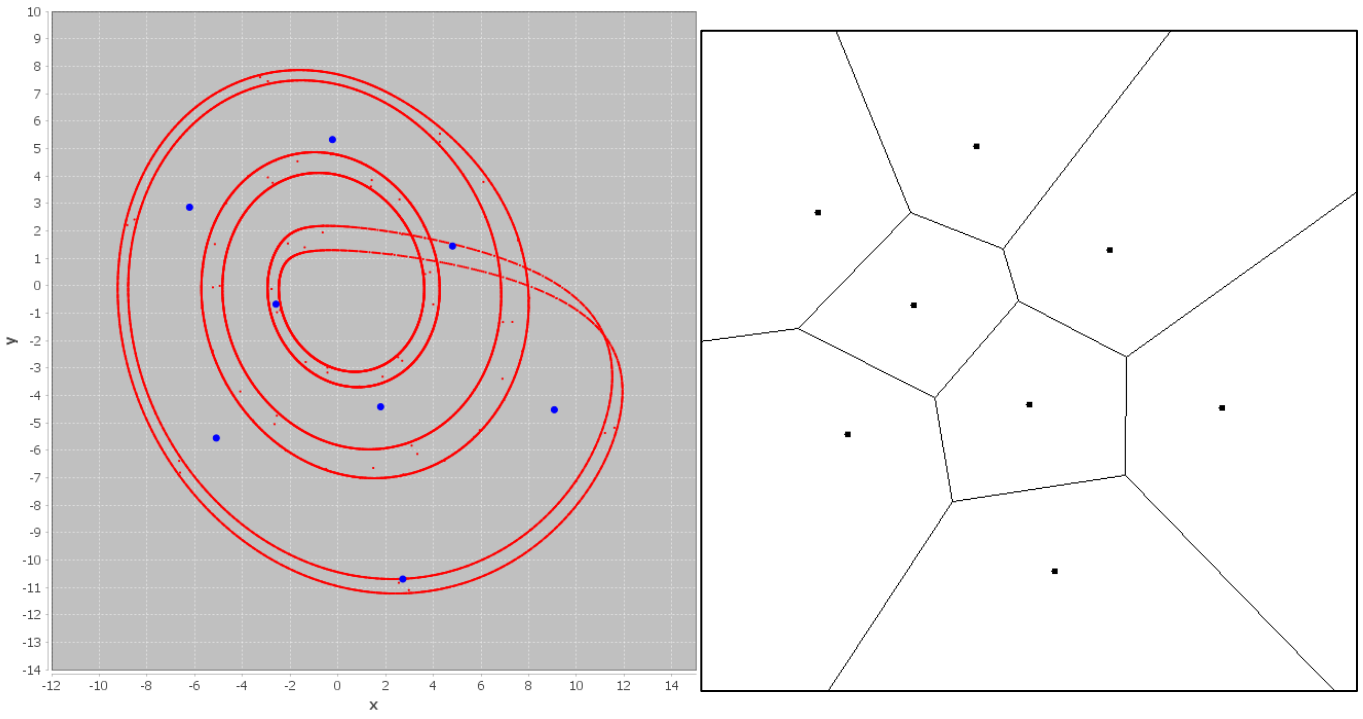
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract.txt	**6**	**9,95**	**0,3 – 0,05**	**0,1 – 0,001**



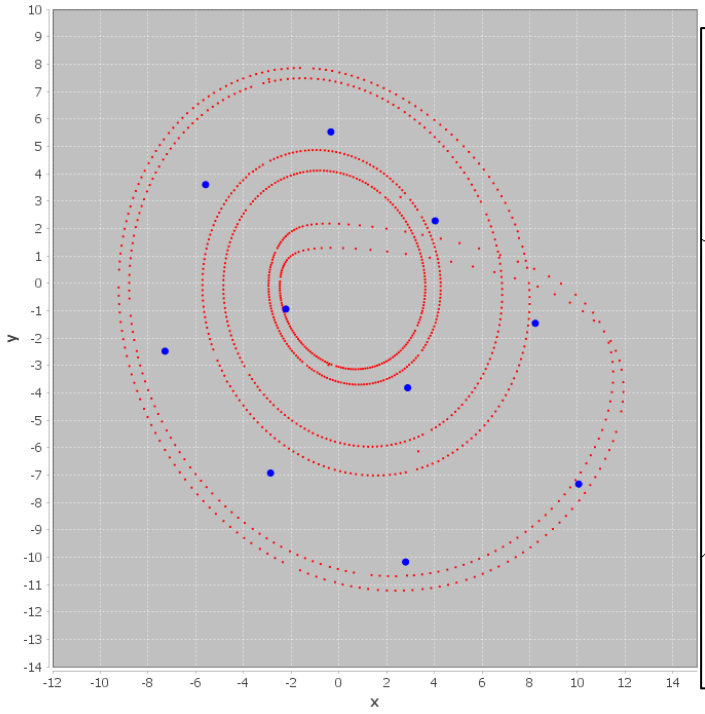
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract_small.txt	**8**	**7,21**	**0,3 – 0,05**	**0,2 – 0,001**

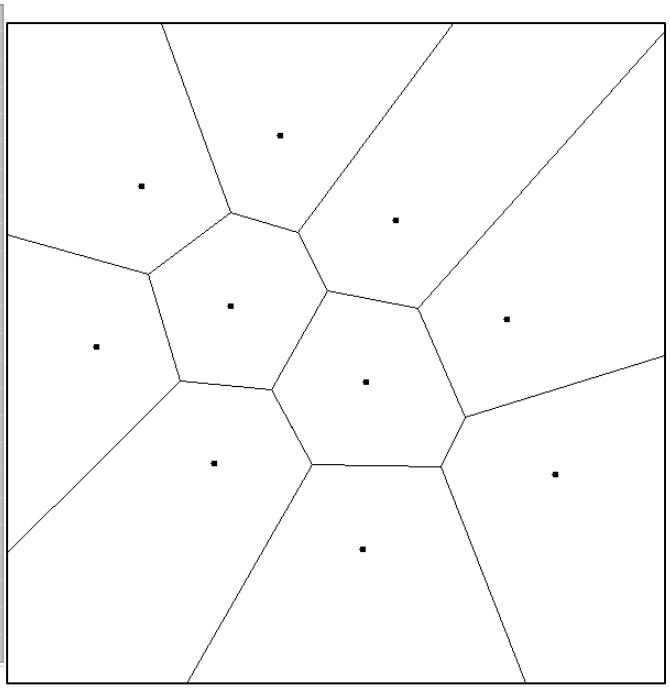


dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract.txt	**8**	**7,16**	**0,3 – 0,05**	**0,2 – 0,01**

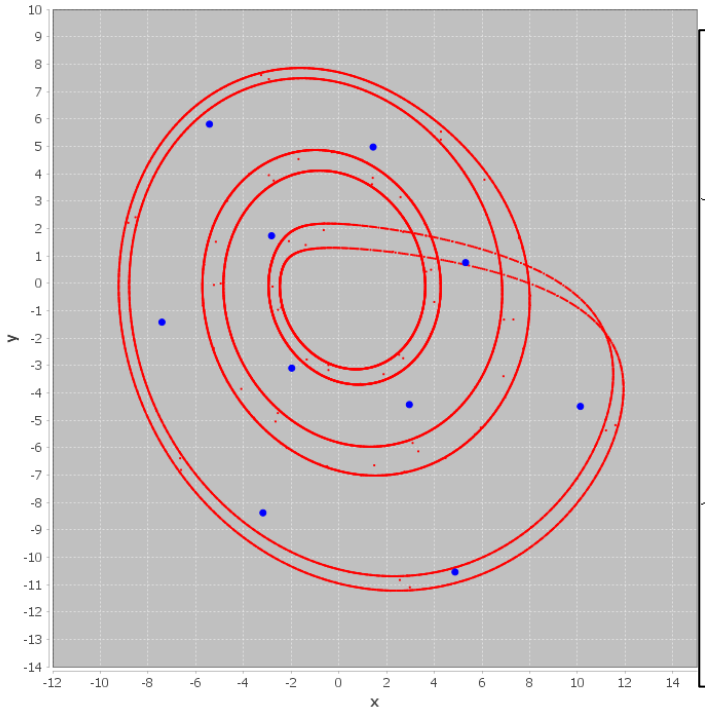


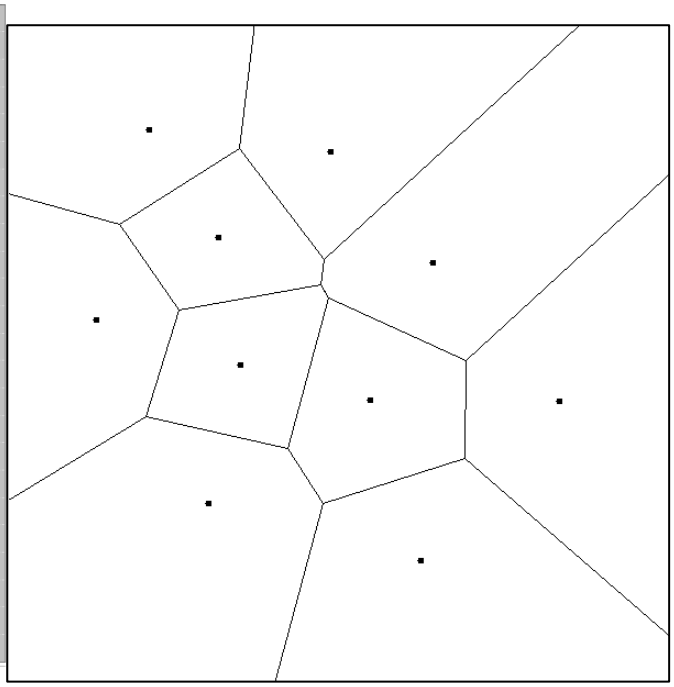
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract_small.txt	**10**	**5,68**	**0,3 – 0,05**	**0,2 – 0,001**



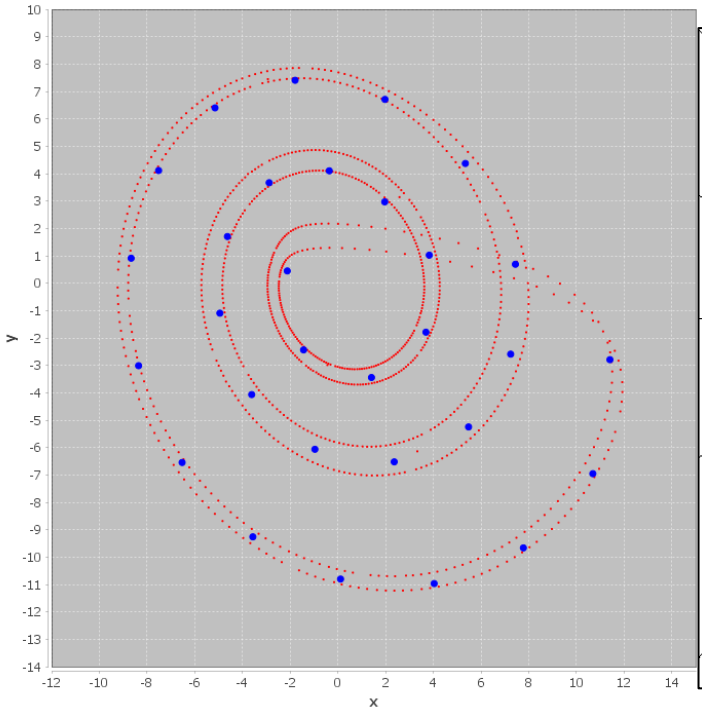


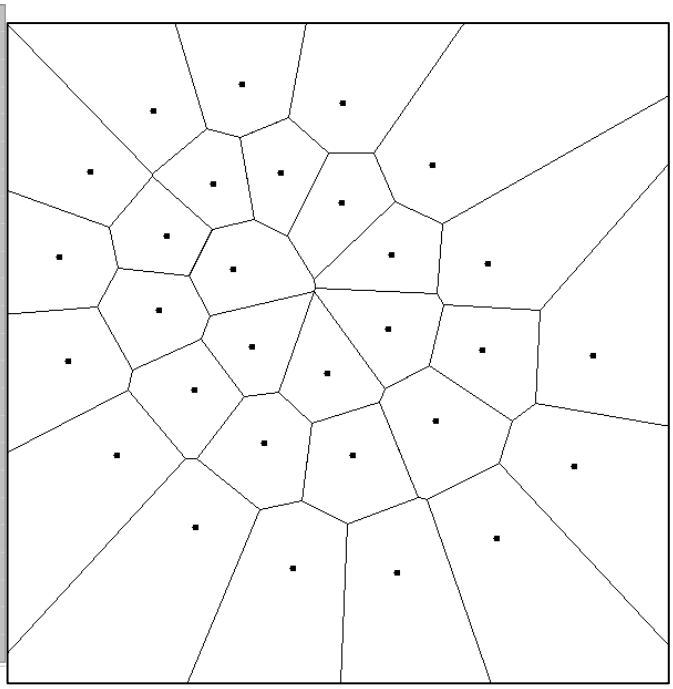
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract.txt	**10**	**5,68**	**0,3 – 0,05**	**0,1 – 0,01**



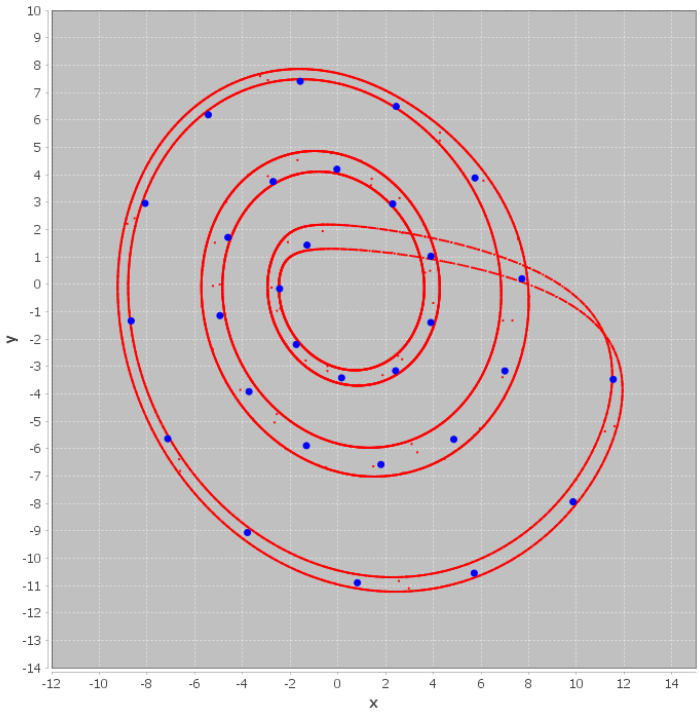


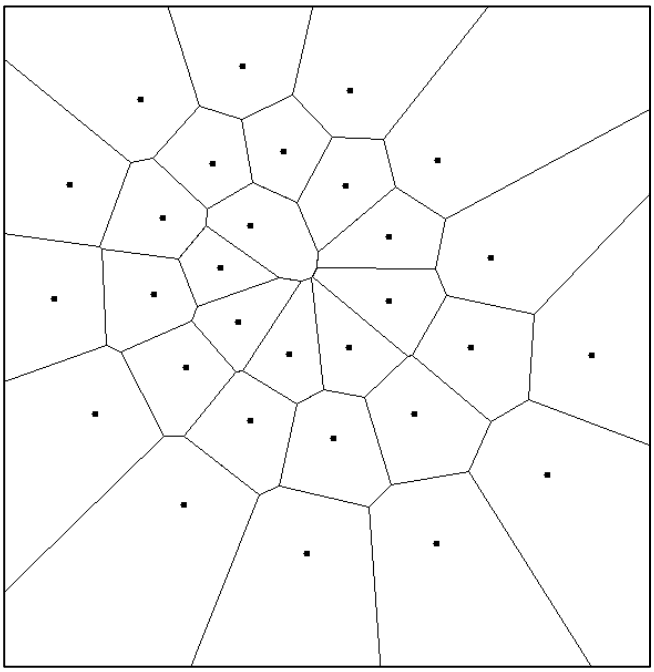
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract_small.txt	**30**	**1,16**	**0,3 – 0,05**	**0,2 – 0,001**



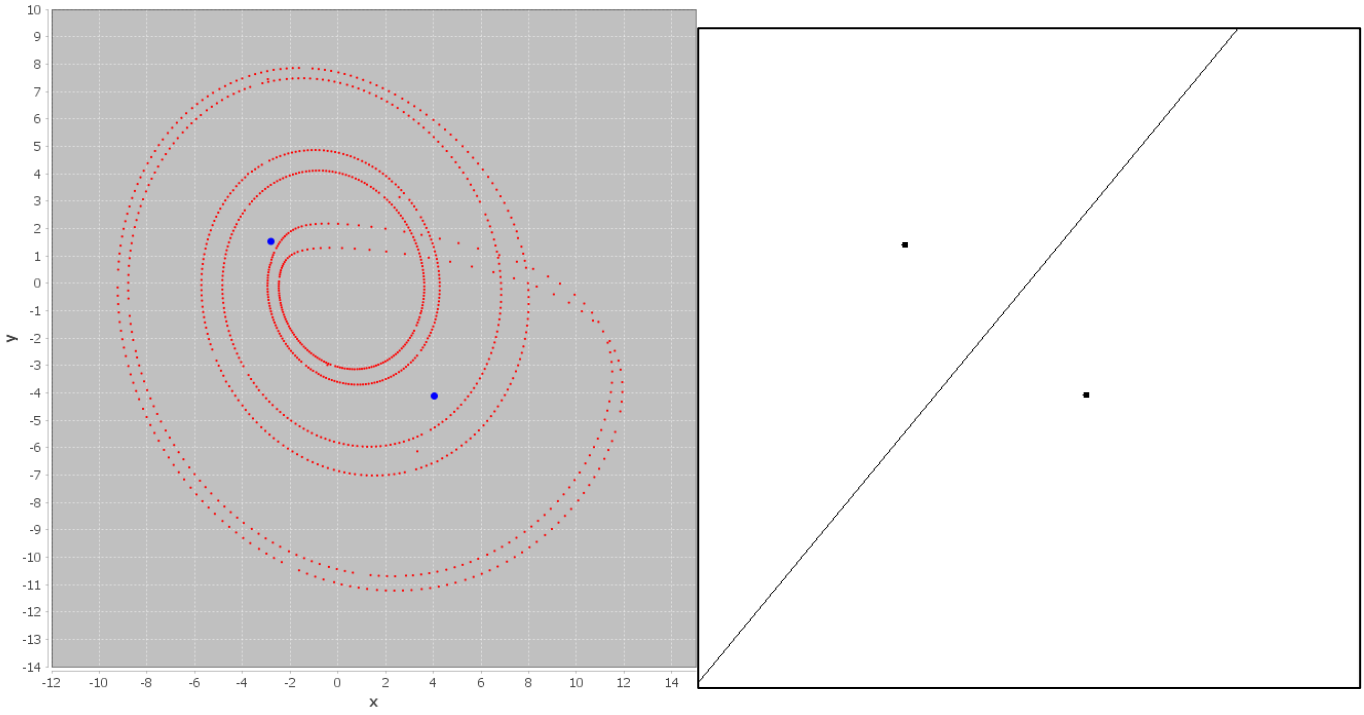


dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract.txt	**30**	**1,22**	**0,3 – 0,05**	**0,2 – 0,001**

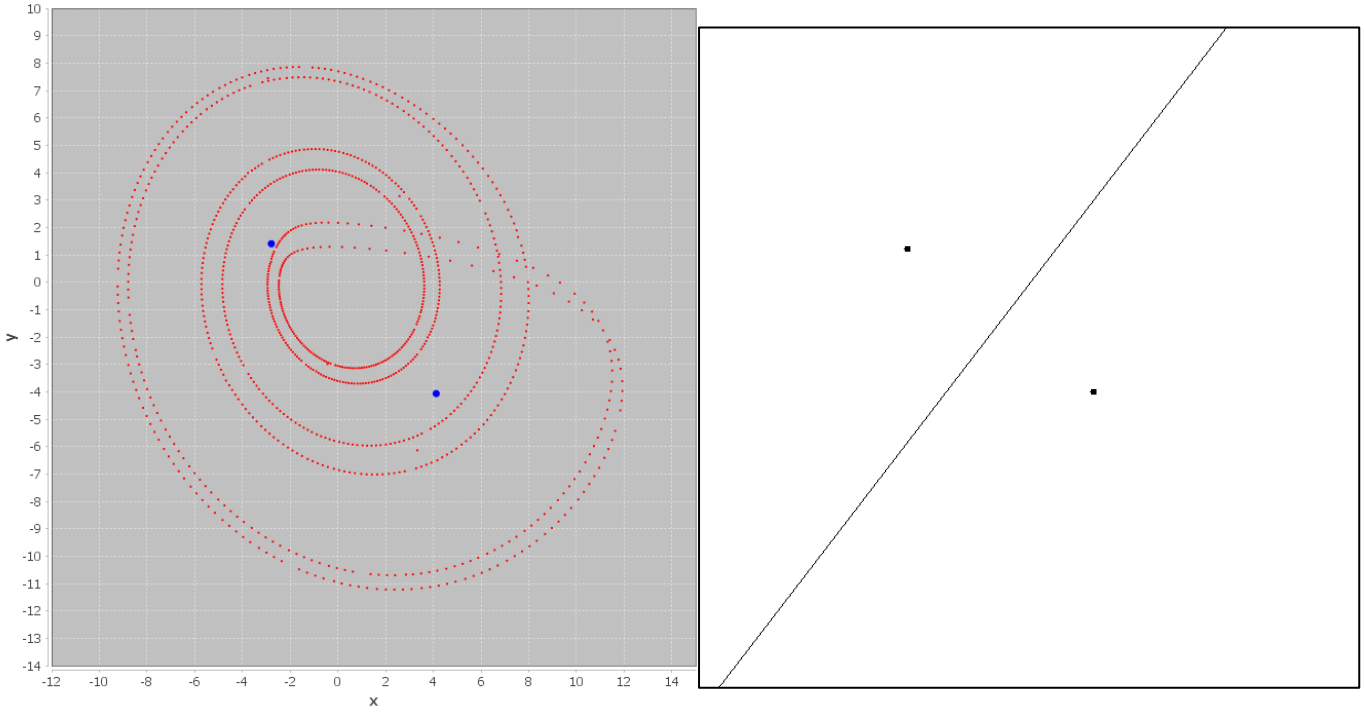




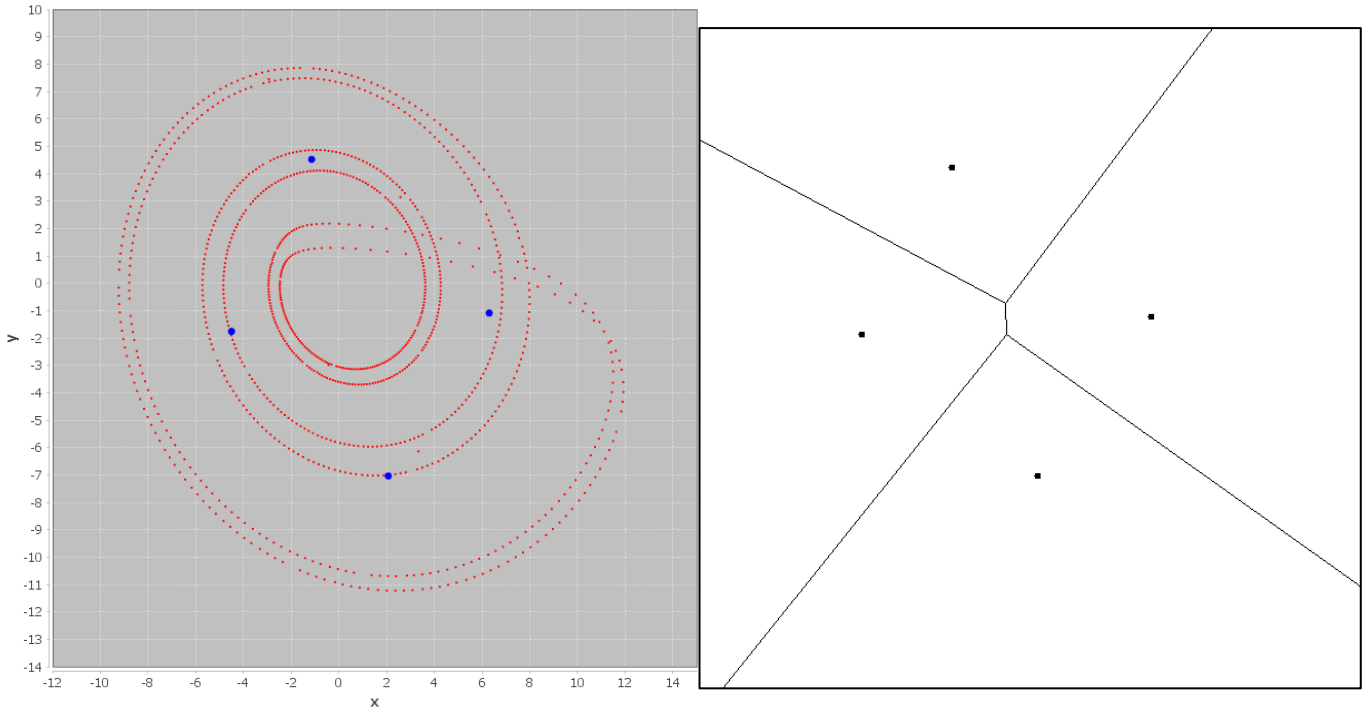
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract_small.txt	**2**	**27,86**	**0,8 – 0,001**	**0,2 – 0,001**



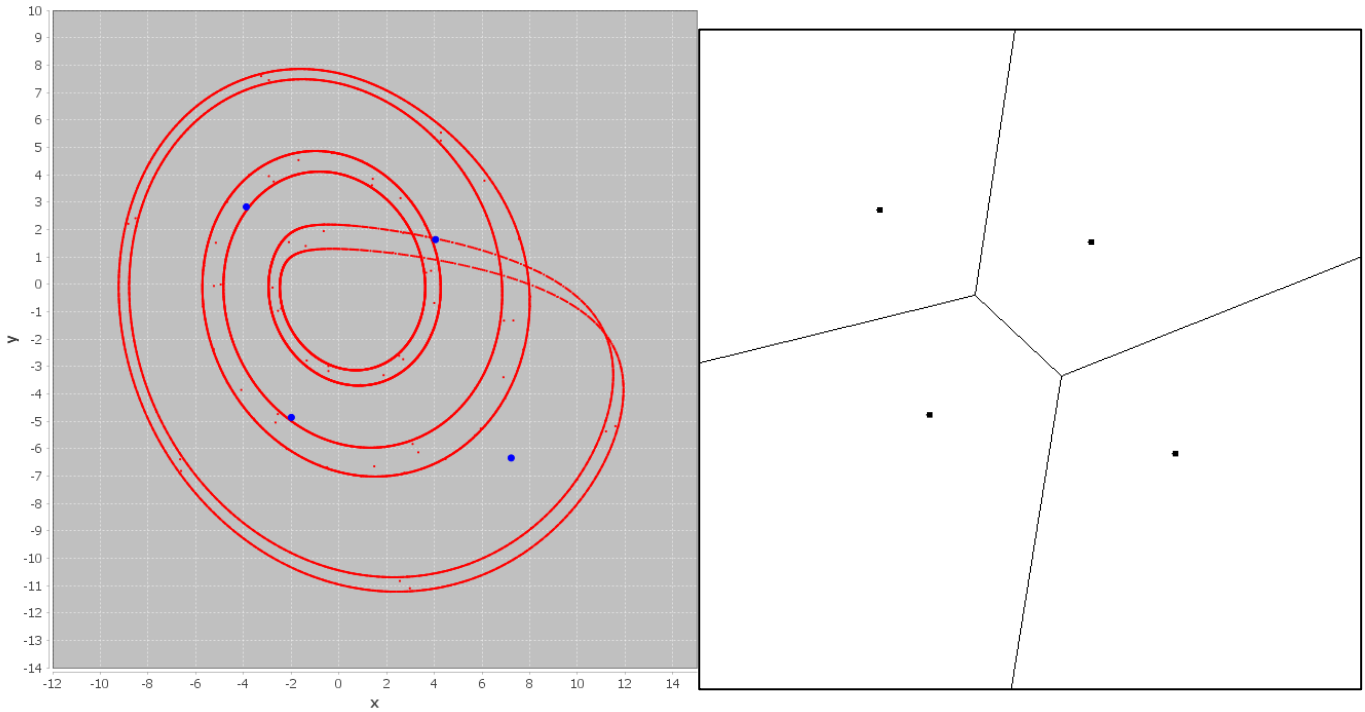
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract.txt	**2**	**27,89**	**0,8 – 0,001**	**0,2 – 0,001**



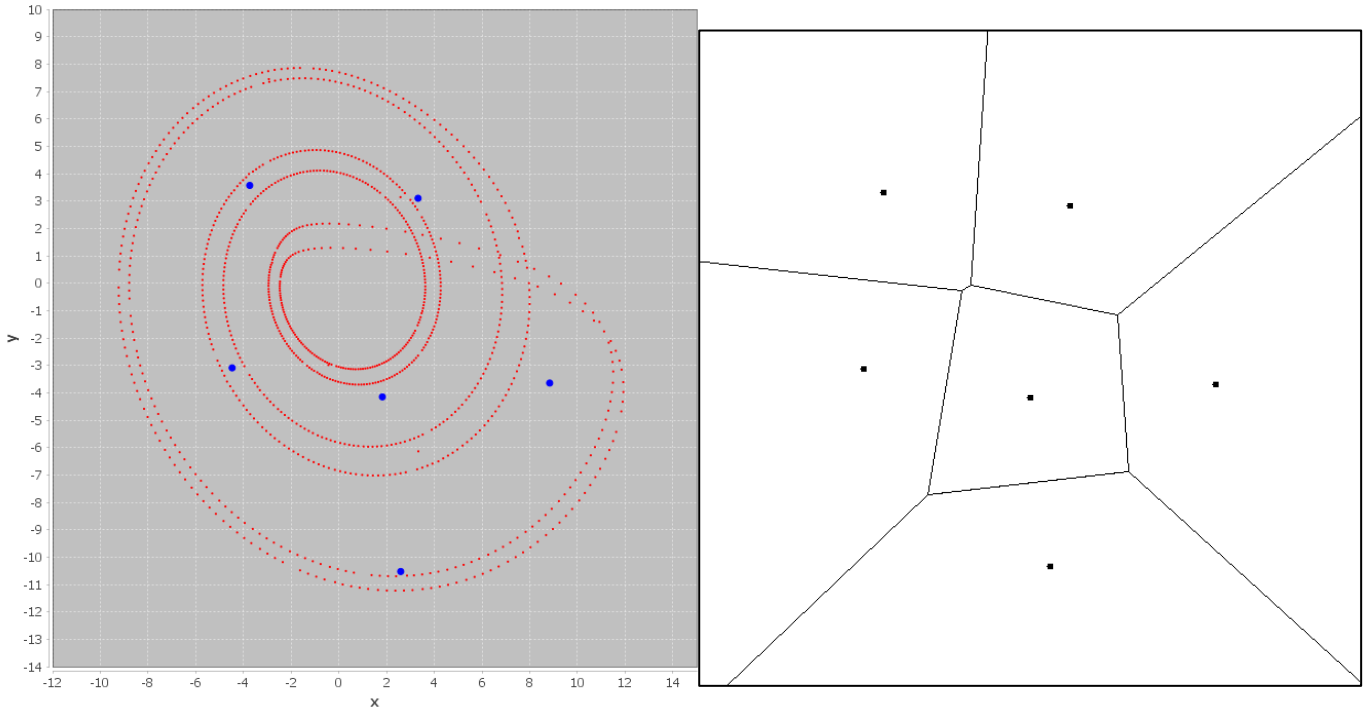
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract_small.txt	**4**	**14,55**	**0,8 – 0,001**	**0,2 – 0,001**



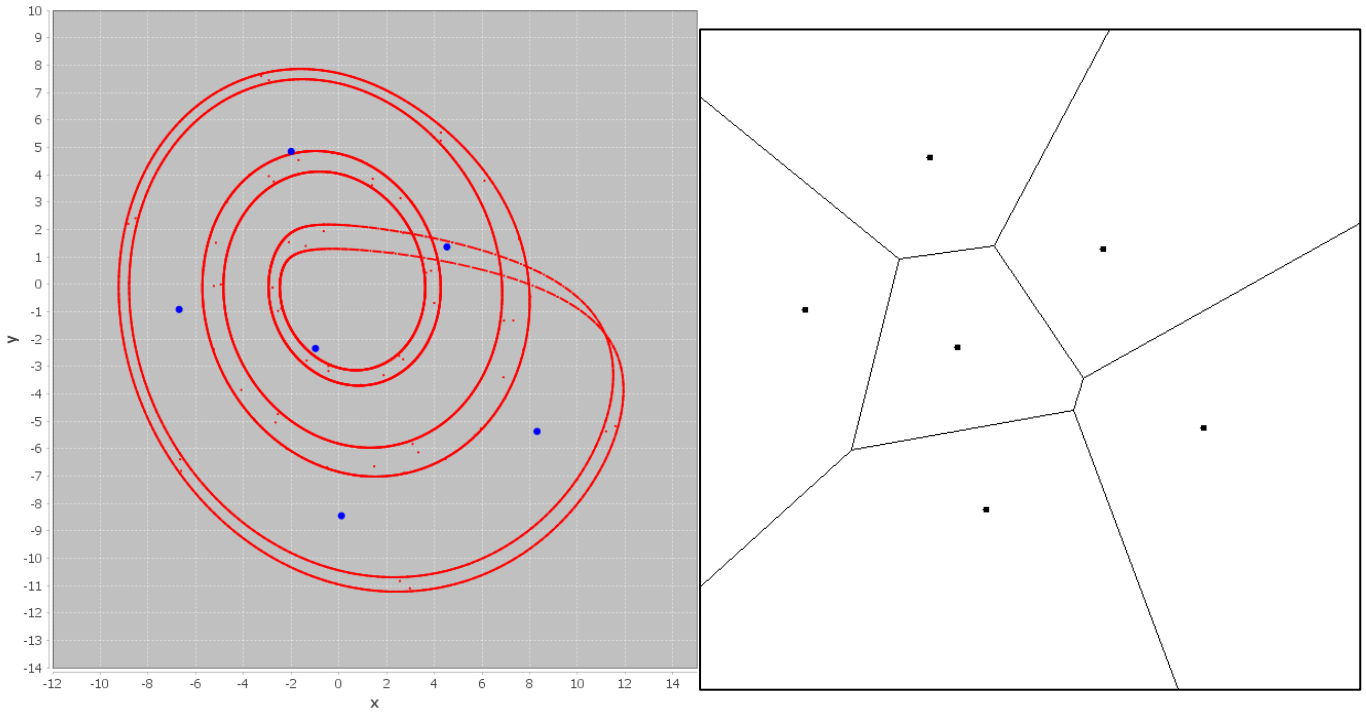
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract.txt	**4**	**14,46**	**0,8 – 0,001**	**0,2 – 0,001**



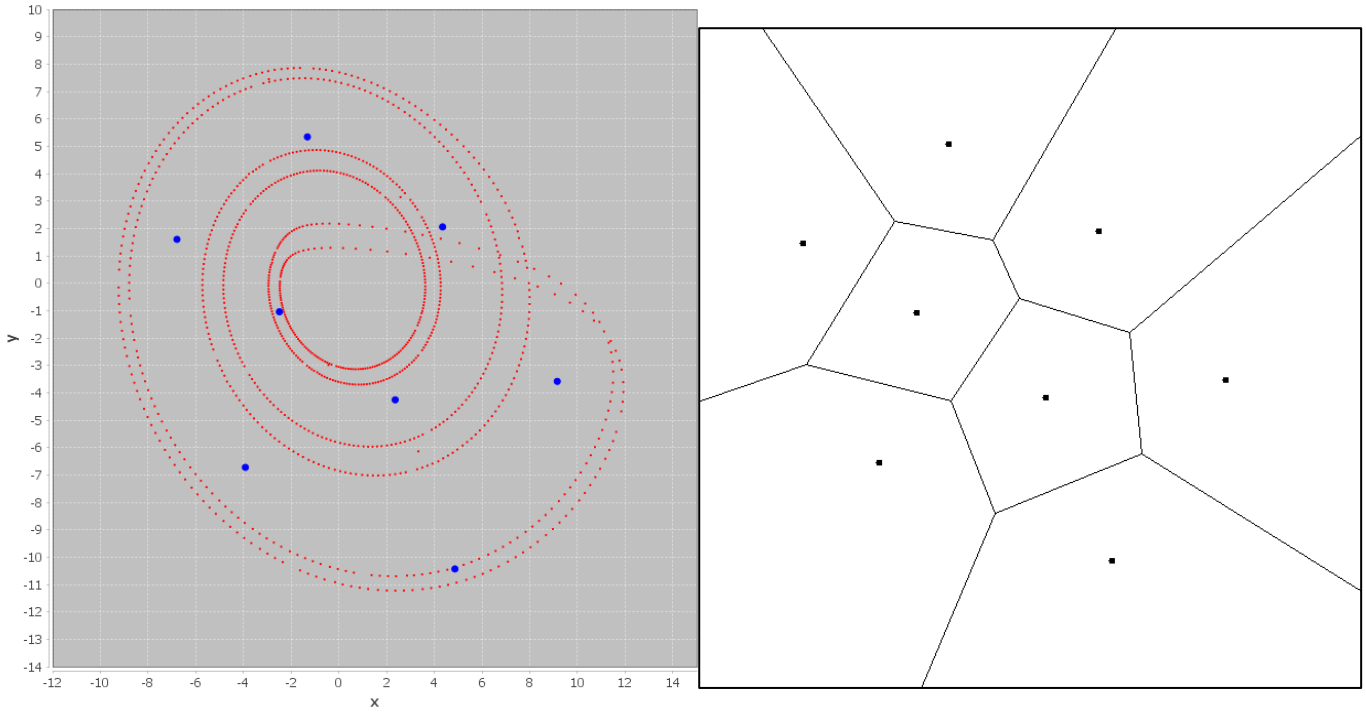
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract_small.txt	**6**	**9,93**	**0,8 – 0,001**	**0,2 – 0,001**



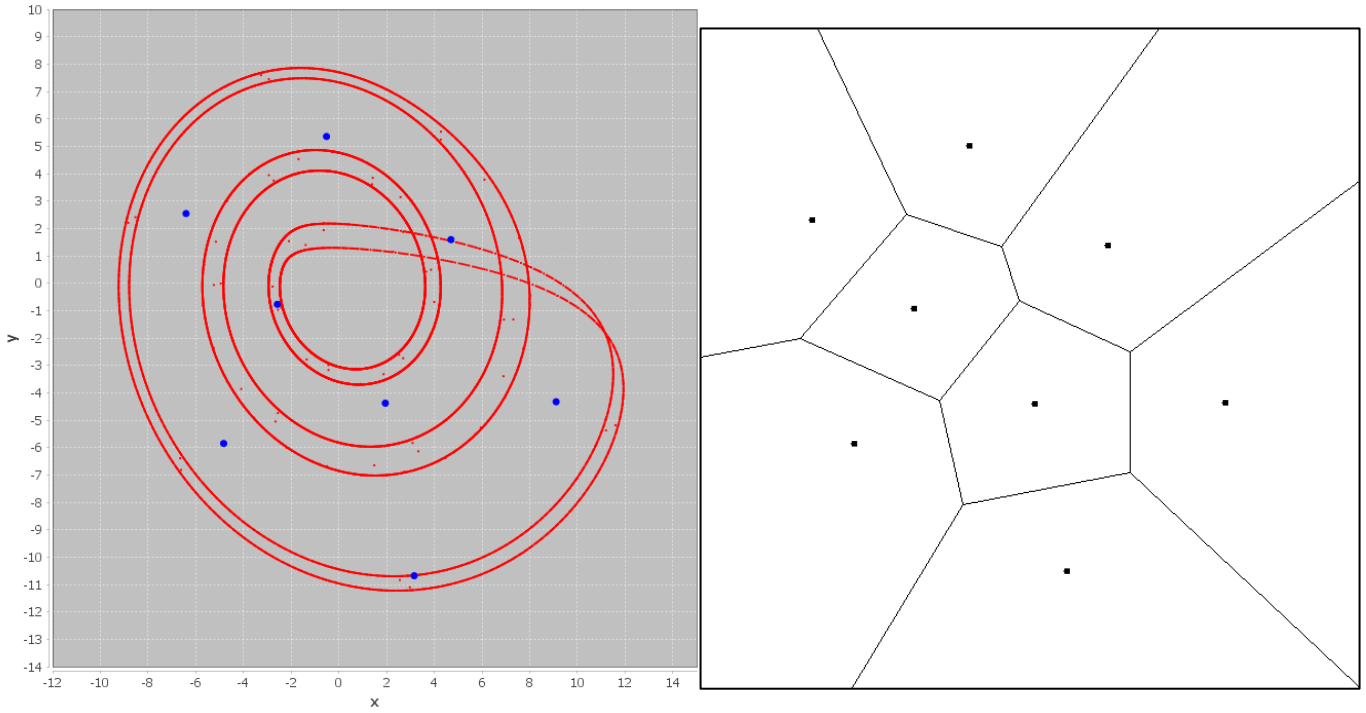
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract.txt	**6**	**9,86**	**0,8 – 0,001**	**0,2 – 0,001**



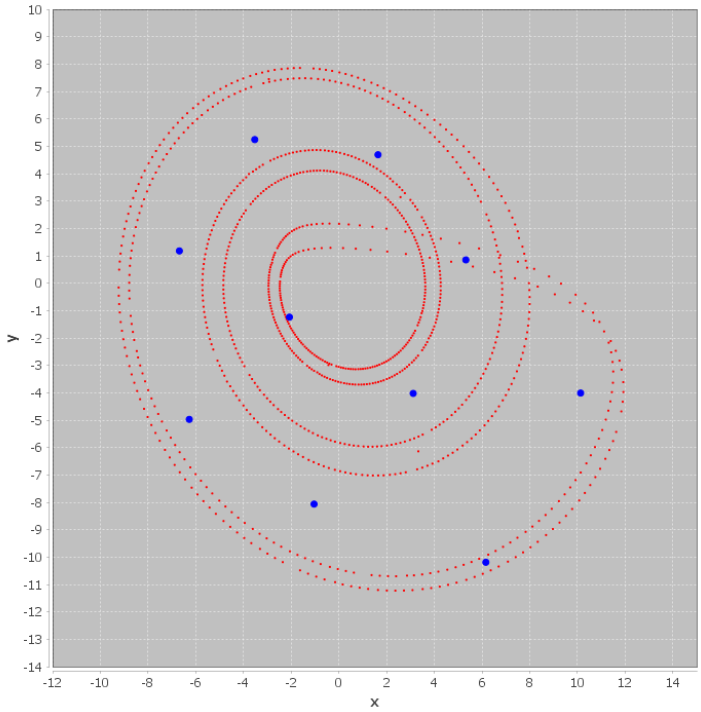
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract_small.txt	**8**	**7,11**	**0,8 – 0,0001**	**0,4 – 0,005**

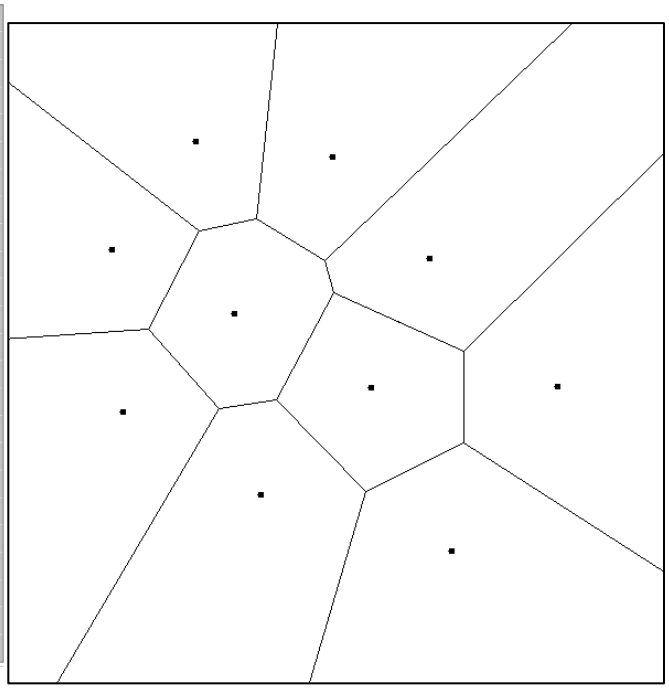


dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract.txt	**8**	**7,09**	**0,8 – 0,0001**	**0,4 – 0,005**

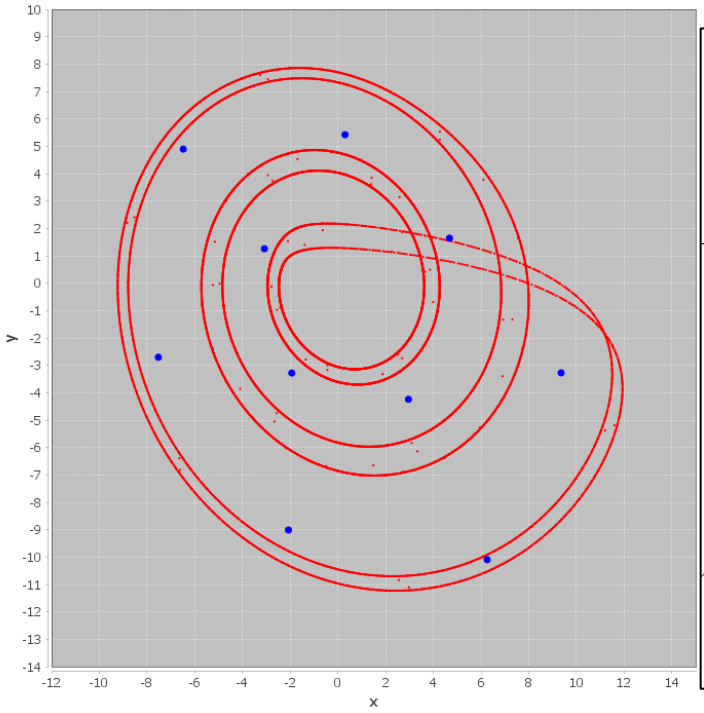


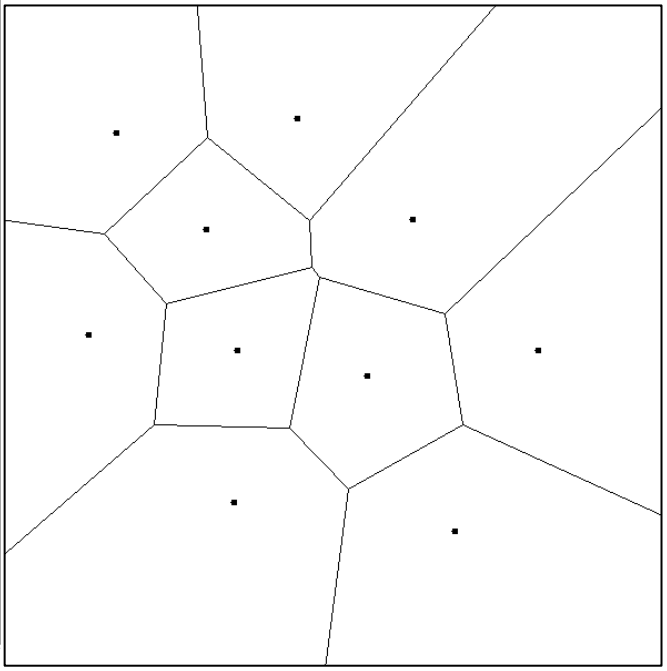
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract_small.txt	**10**	**5,62**	**0,8 – 0,0001**	**0,4 – 0,005**



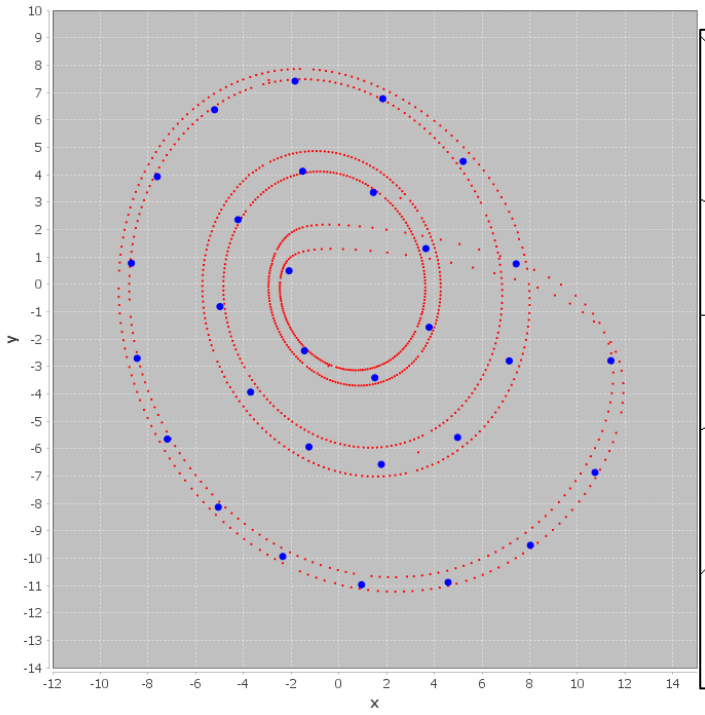


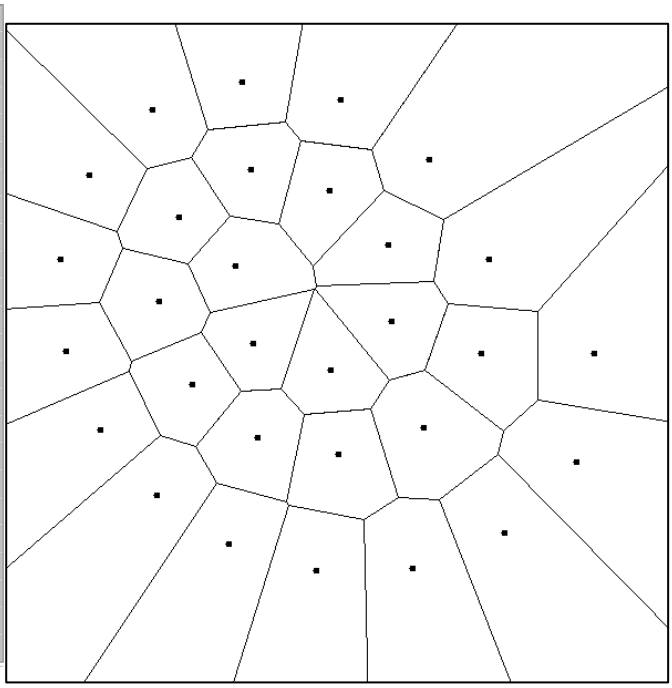
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract.txt	**10**	**5,59**	**0,8 – 0,0001**	**0,4 – 0,005**



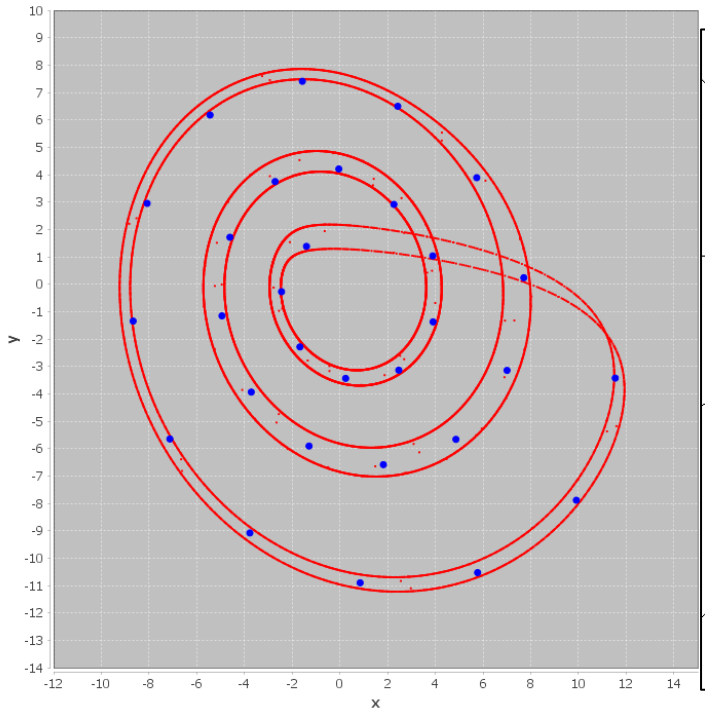


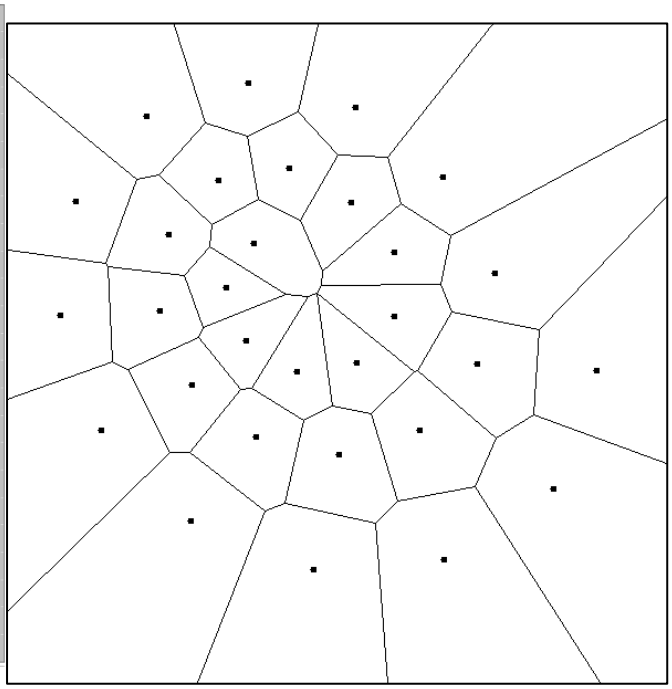
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract_small.txt	**30**	**1,18**	**1,0 – 0,0001**	**0,1 – 0,005**





dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract.txt	**30**	**1,21**	**1,0 – 0,0001**	**0,1 – 0,005**





5.3. Zestawienie wyników

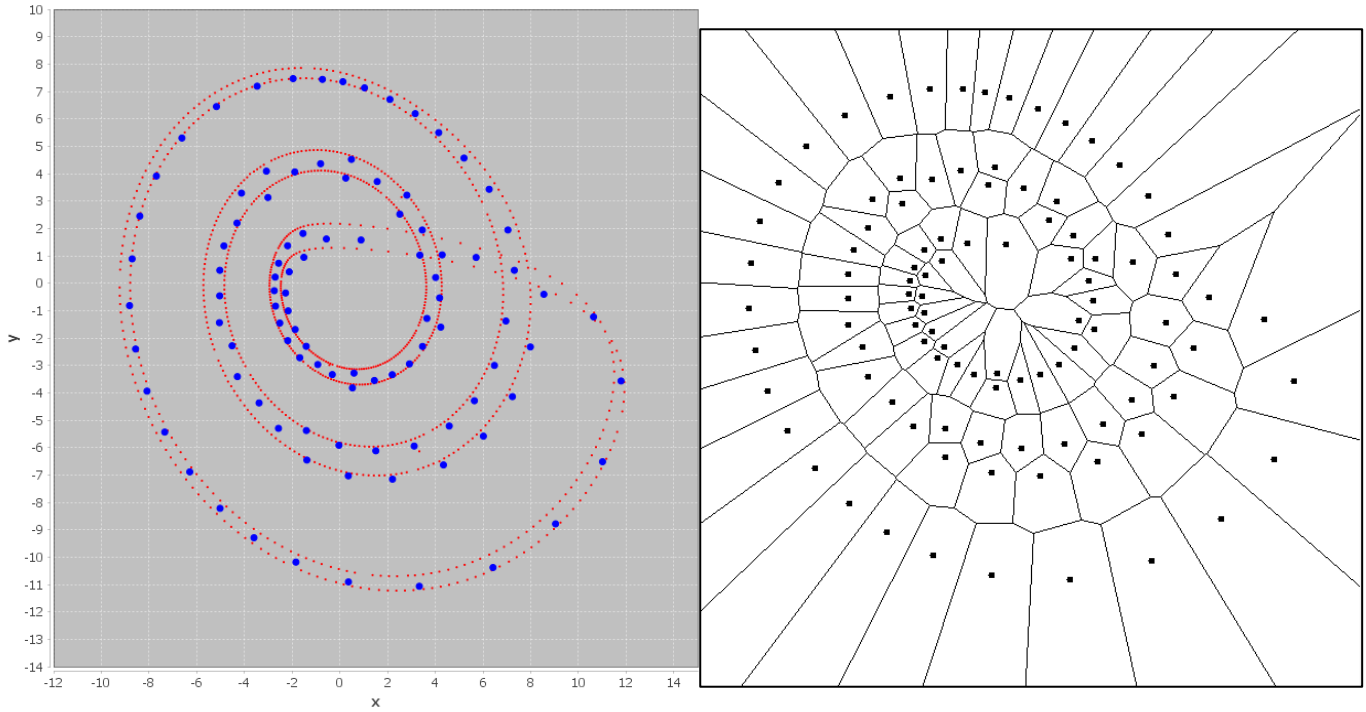
	średni błąd kwantyzacji		
liczba centroid/neuronów	algorytm k-średnich	sieć Kohonena	a. gazu neuronowego
2	27,84 / 27,86	28,12 / 28,14	27,86 / 27,89
4	14,47 / 14,44	14,72 / 14,59	14,55 / 14,46
6	9,99 / 9,85	9,93 / 9,95	9,93 / 9,86
8	7,07 / 7,10	7,21 / 7,16	7,11 / 7,09
10	5,85 / 5,57	5,68 / 5,68	5,62 / 5,59
30	1,20 / 1,21	1,16 / 1,22	1,18 / 1,21

5.4. Grupowanie neuronów uzyskanych w wyniku działania sieci SOM

Pierwszym etapem tej części zadania było pogrupowanie danych przy użyciu dużej liczby neuronów, przyjęto liczbę 100 dla danych attract_small.txt i 150 dla attract.txt. Następnie uzyskane neurony zostały pogrupowane algorytmem k-średnich.

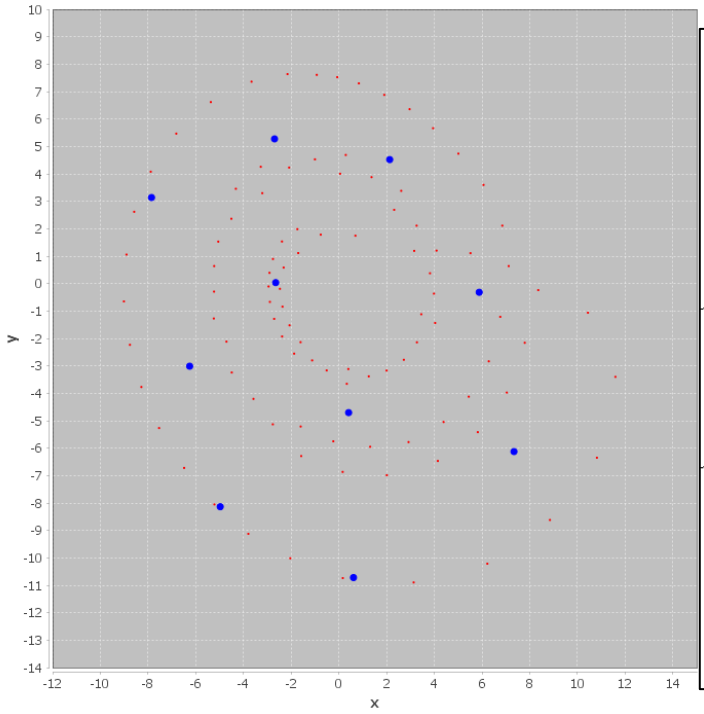
5.4.1. Sieć Kohonena

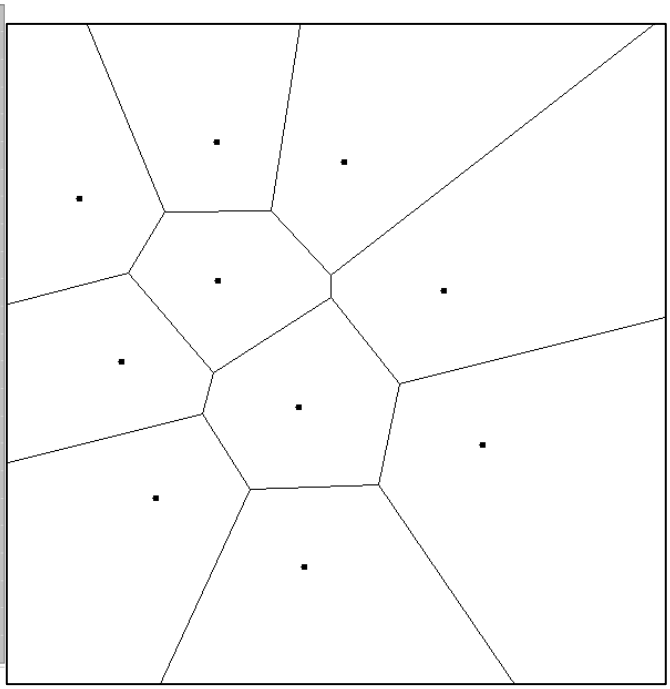
dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract_small.txt	**100**	**0,28**	**0,2 – 0,0001**	**0,2 – 0,01**



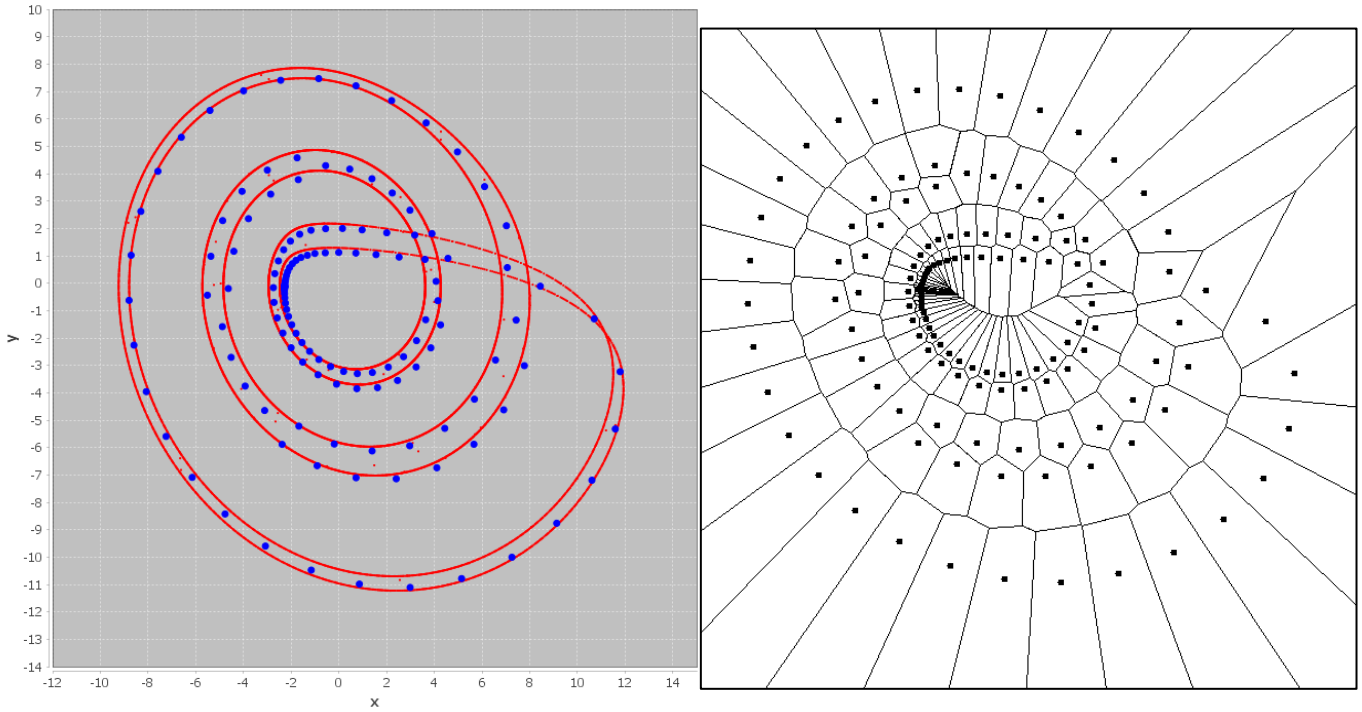
Grupowanie neuronów algorytmem k-średnich

dane	liczba centroid	średni błąd na końcu adaptacji
attract_small.txt	**10**	**5,61**



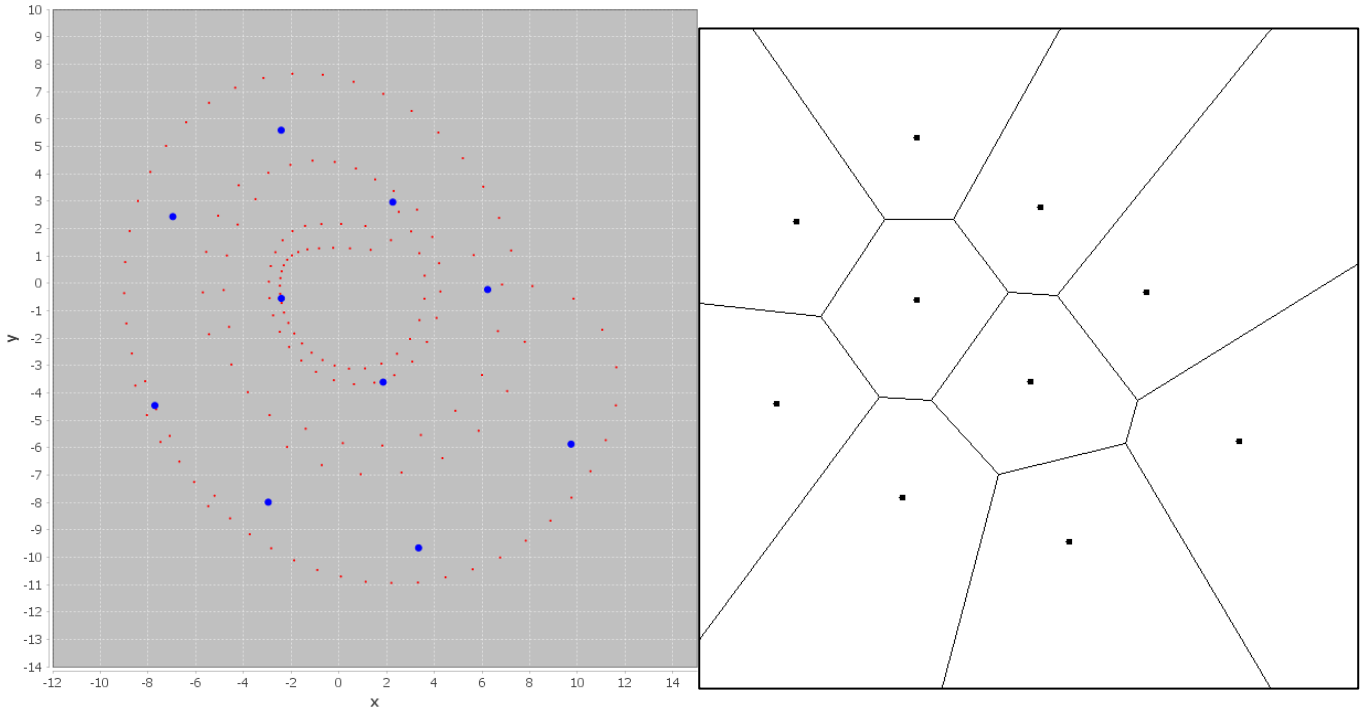


dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract.txt	**150**	**0,16**	**0,15 – 0,0001**	**0,1 – 0,005**



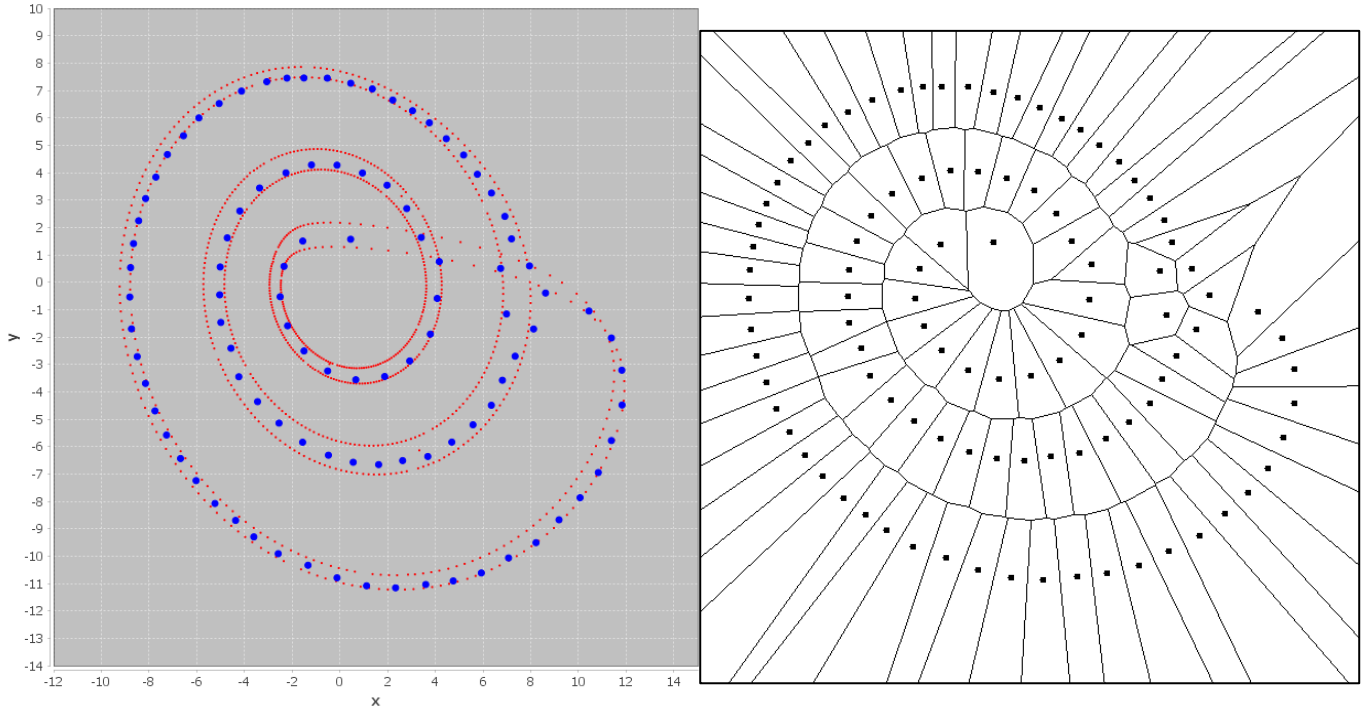
Grupowanie neuronów algorytmem k-średnich

dane	liczba centroid	średni błąd na końcu adaptacji
attract_small.txt	**10**	**5,37**



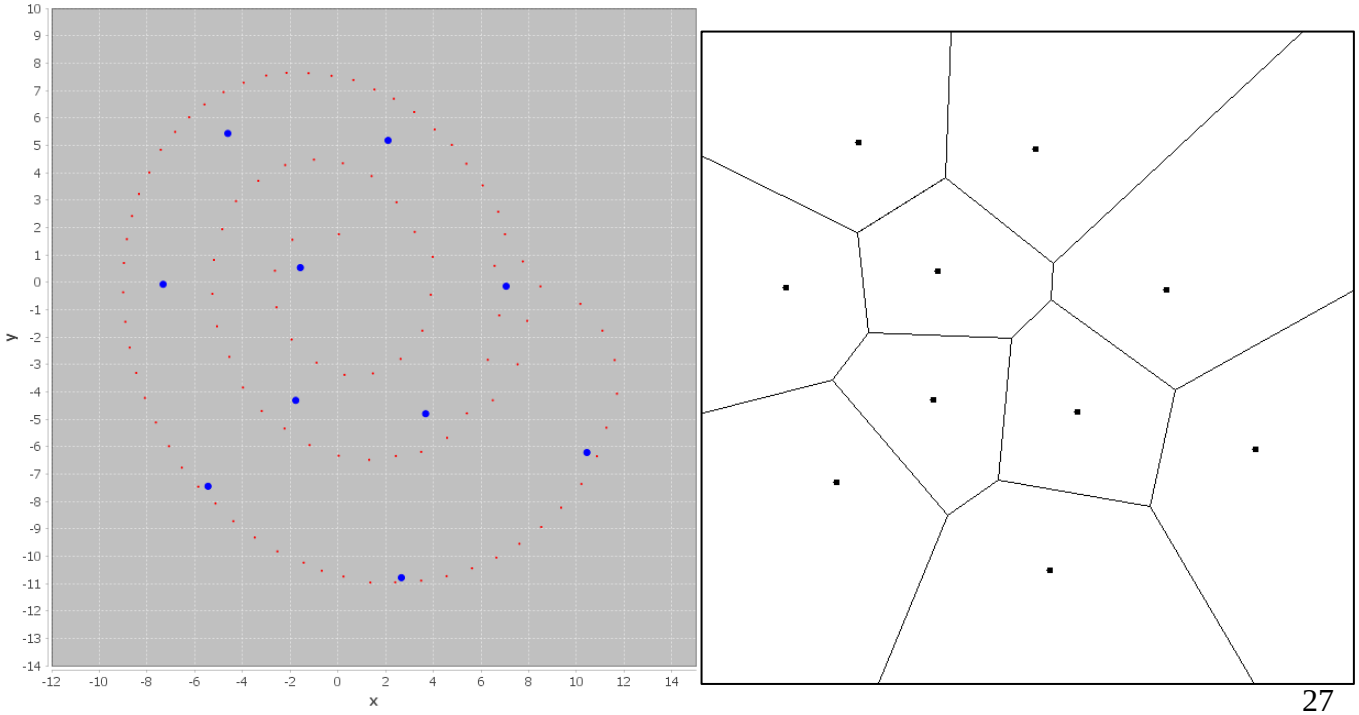
5.4.2. Algorytm gazu neuronowego

dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract_small.txt	100	0,24	2.5 – 0,001	0,2 – 0,005

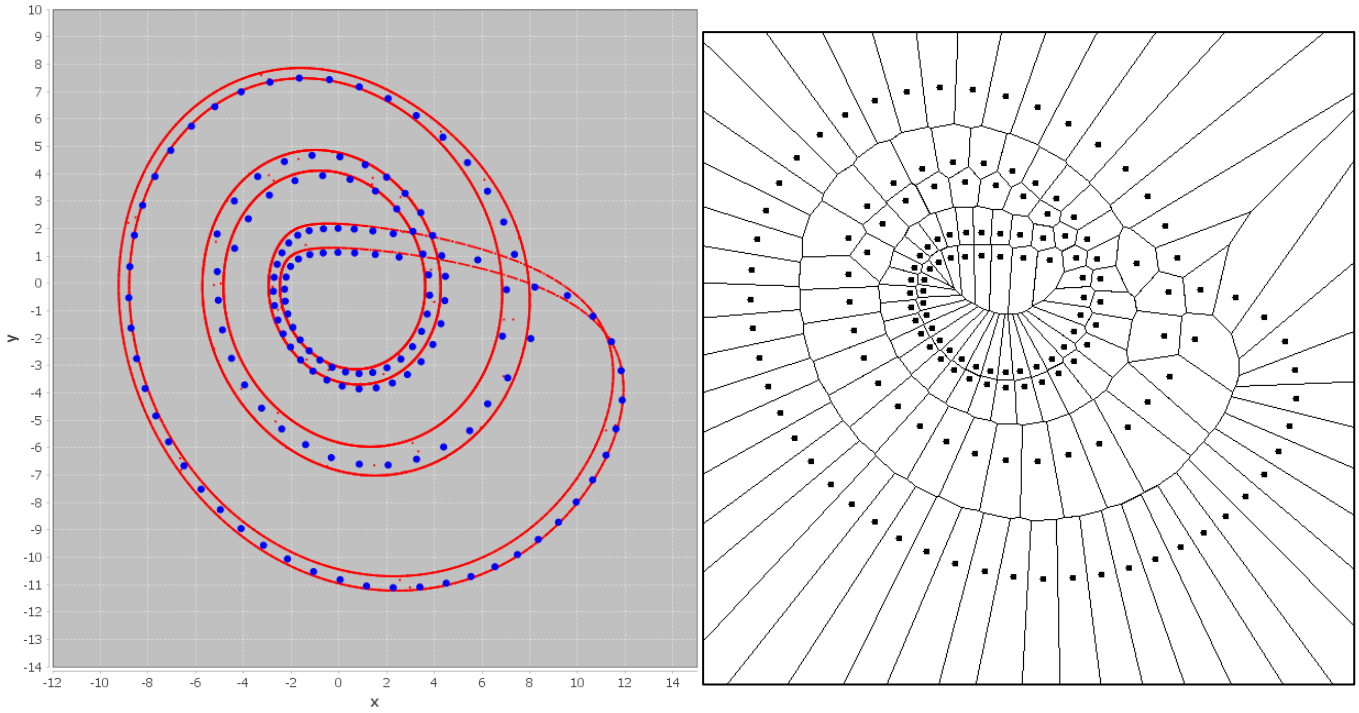


Grupowanie neuronów algorytmem k-średnich

dane	liczba centroid	średni błąd na końcu adaptacji
attract_small.txt	10	6,15

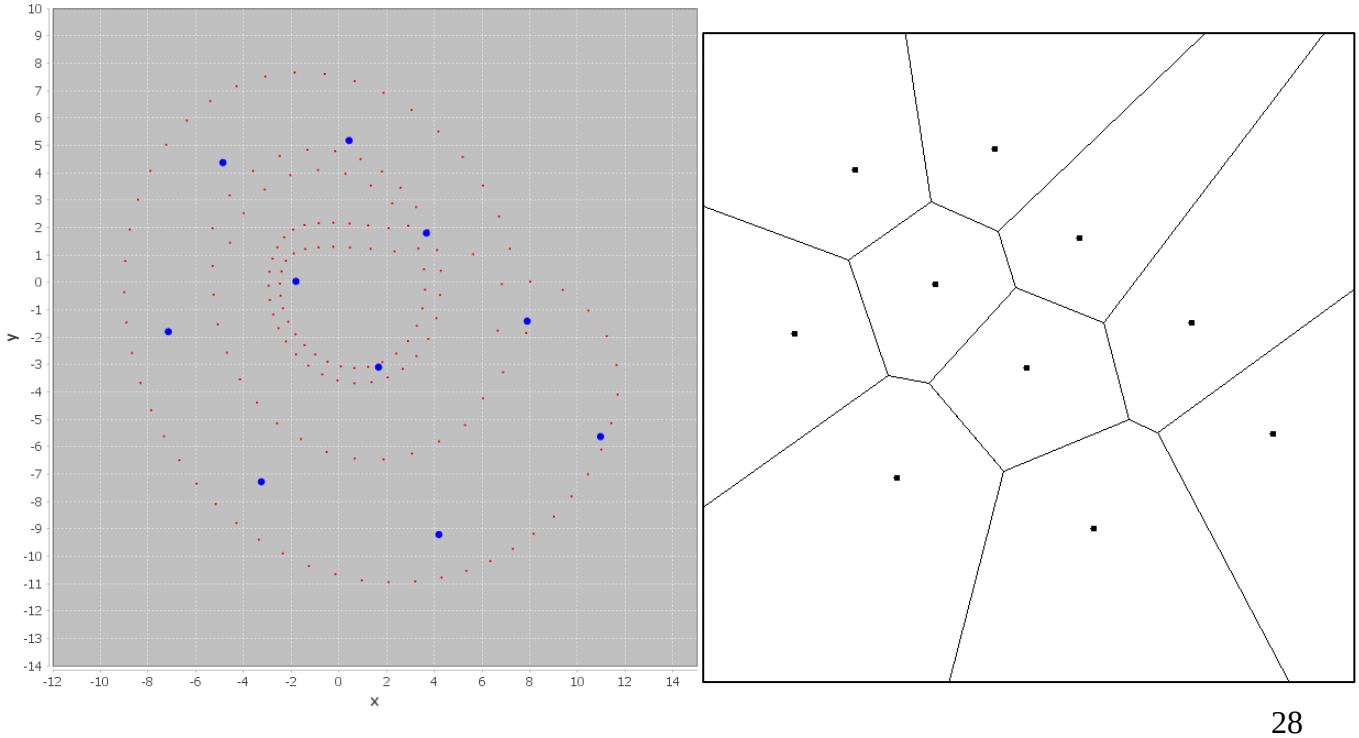


dane	liczba neuronów	średni błąd na końcu adaptacji	zakres parametru λ	zakres parametru η
attract.txt	**150**	**0,16**	**2.5 – 0,001**	**0,2 – 0,005**



Grupowanie neuronów algorytmem k-średnich

dane	liczba centroid	średni błąd na końcu adaptacji
attract.txt	**10**	**5,39**



5.5. Wnioski

Wszystkie testowane algorytmy skutecznie grupowały zbiory danych i dawały porównywalne wartości średnich błędów kwantyzacji, co zrozumiałe zwiększanie liczby grup powoduje obniżenie wartości błędu. Przy czym podczas pracy algorytmu k-średnich już przy liczbie centroid większej niż 8 pojawiają się martwe centroidy, co więcej końcowy błąd w dużym stopniu zależy od położenia początkowego centroid. Natomiast problem martwych neuronów nie występuje w sieciach SOM, z drugiej strony algorytm k-średnich jest zdecydowanie najszybszą metodą.

Nie zaobserwowano znaczących różnic pomiędzy grupowaniem obu badanych zbiorów danych.

W drugiej części badań sprawdzono poprawność grupowania. Grupowanie neuronów uzyskanych z sieci Kohonena algorytmem k-średnich generuje błąd porównywalny z błędem grupowania oryginalnego zbioru danych. Wskazuje to na poprawność zastosowanego algorytmu. Natomiast w przypadku algorytmu gazu neuronowego wyniki nie są jednoznaczne. W wyniku kontroli grupowanie mniejszego zbioru danych 100 neuronami otrzymano wyższy błąd niż w przypadku grupowania zbioru wejściowego (6,15 vs 5,85). Z kolei grupowanie większego zbioru 150 neuronami daje wyniki lepsze niż sieć Kohonena.