Sprawozdanie do projeku:

Wykorzystanie probabilistycznych sieci neuronowych do klasyfikacji danych o zmiennym charakterze

Grzegorz Ryniak, Rafał Szęszoł

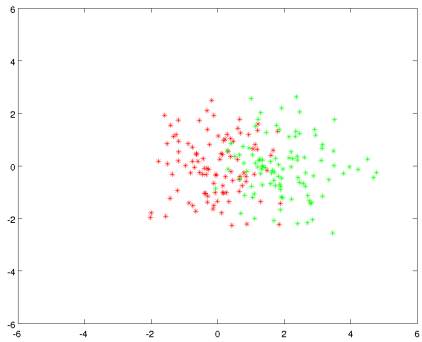
Kraków, czerwiec 2018

1 Wstęp

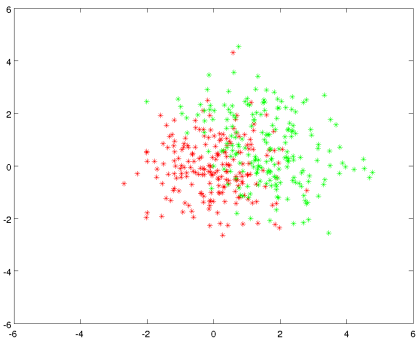
W świecie rzeczywistym często spotykamy się ze źródłami informacji, które z czasem zaczynają przesyłać zmodyfikowane dane. Przykładem może być elektroniczny czujnik grubości - odpowiednik czujnika zegarkowego. W skutek zużycia np. końcówek pomiarowych, z czasem otrzymywane wartości mogą być zaniżone. Duży problem pojawia się kiedy odbierane dane poddawane są klasyfikacji. Klasyfikator widząc przesunięcie danych może uznać, że należą one do innej klasy. Z czasem tych klas mogło by powstać coraz więcej. Ze względu na ograniczone zasoby i łatwość interpretacji często wymaga się, aby liczba klas była stała. Celem tego projektu było stworzenie klasyfikatora opartego o probabilistyczne sieci neuronowe, który byłby w stanie na bieżąco dostosowywać się do zmiennych danych.

2 Doświadczenia

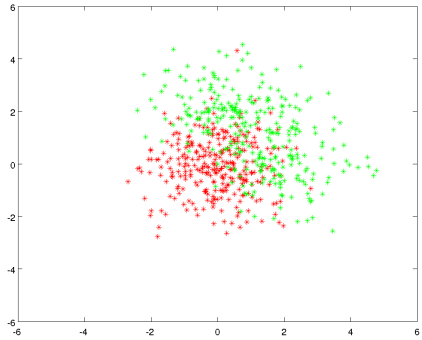
Wszystkie symulacje zostały przeprowadzone z wykorzystaniem środowiska MATLAB 2012b. W pierwszej kolejności stworzono generator danych, który symulował by rzeczywiste źródło informacji podlegające zużyciu. Stworzone zostały 2 zestawy danych. Generowane były po 2 klasy dwu-wymiarowych danych. Dane takie łatwo da się przedstawić na układzie współrzędnych w postaci punktów. W pierwszym zestawie pierwsza klasa składała się z punktów o współrzędnych generowanych z rozkładu Gaussa o średniej wartości (0;0) i odchyleniu 1, natomiast druga klasa również była generowana na podstawie rozkładu Gaussa o średnim odchyleniu 1 ale średnie wartości były punktami leżącymi na okręgu o promieniu 2 i środku w punkcie (0;0). Kąt pomiędzy osią x a promieniem zwiększał się z czasem w kierunku przeciwnym do wskazówek zegara. Co 90 stopni ruch na chwilę się zatrzymywał, aby sprawdzić, czy sieć nie przyzwyczaiła się do ruchu. Generowanie danych kończy się w momencie kiedy druga klasa „okrąży" pierwszą dookoła. Kolejne etapy generowania danych pokazane są na rysunku 1.



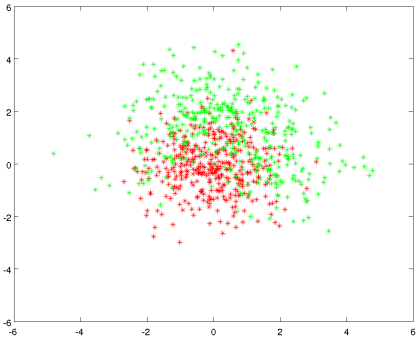
(a) Krok 1



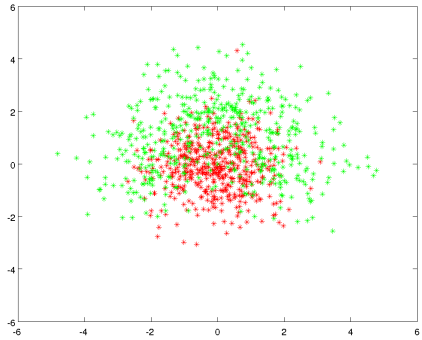
(b) Krok 2



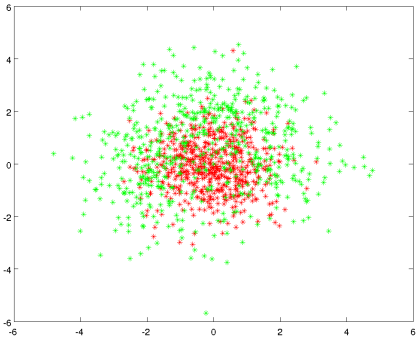
(c) Krok 3



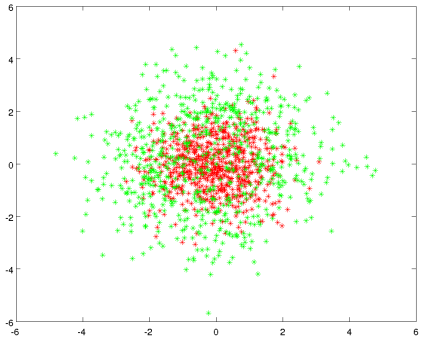
(d) Krok 4



(e) Krok 5



(f) Krok 6



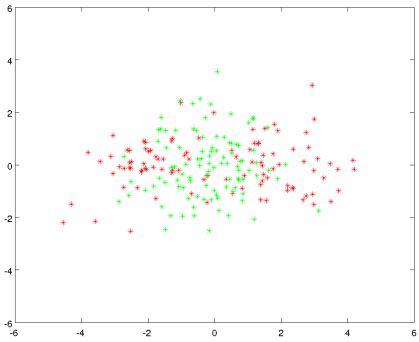
(g) Krok 7

Rysunek 1: Poszczególne etapy generowania danych dla zestawu pierwszego

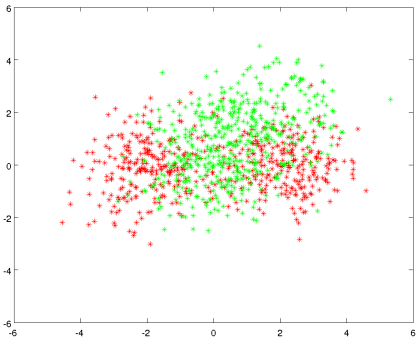
Drugi zestaw składał się również z dwóch dwu wymiarowych klas. Pierwsza klasa składała się z punktów rozsianych wg rozkładu Gaussa z dwoma punktami centralnymi (-2,0) i (2,0). Druga klasa przemieszczała

się po "ósemce- dwóch połączonych okręgach wokół klasy pierwszej. Promień okręgu wynosił 4. Odchylenie standardowe przy rozkładzie Gaussa w obu klasach wyniosło 1. Kolejne etapy generowania danych dla zestawy 2 pokazane są na rysunku 2.

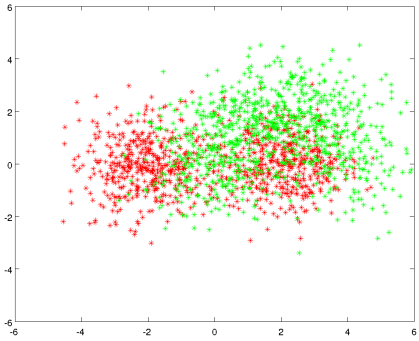
1.



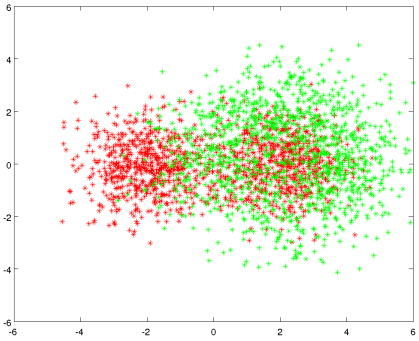
(a) Krok 1



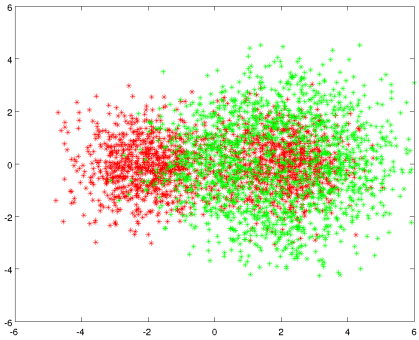
(b) Krok 2



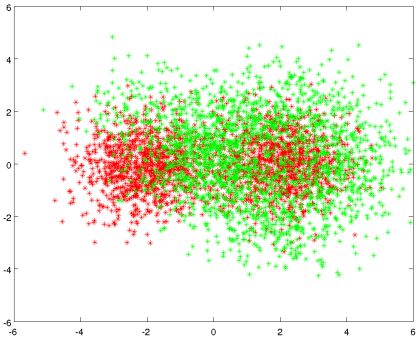
(c) Krok 3



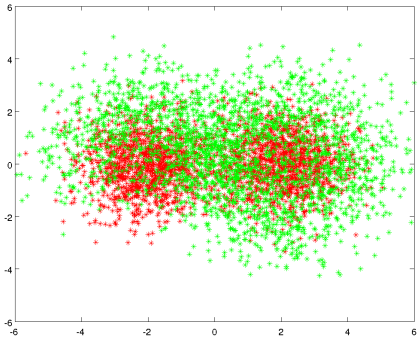
(d) Krok 4



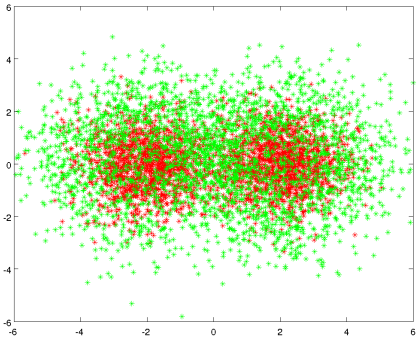
(e) Krok 5



(f) Krok 6



(g) Krok 7



(h) Krok 8

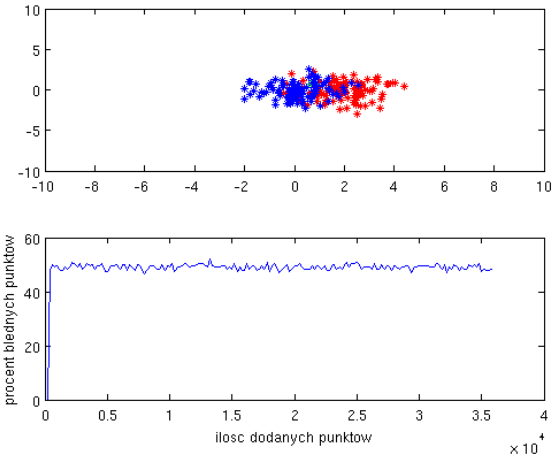
Rysunek 2: Poszczególne etapy generowania danych dla zestawu drugiego

Liczbę punktów generowanych w każdym kroku można było zmieniać. Co określoną ilość dodanych punktów sieć neuronowa była uczona nowych danych. Podczas ponownego uczenia sieci, zmianom podlegał parametr wygładzania. Algorytm wyznaczania tegoż parametru został dostarczony przez prowadzącego zajęcia. Podczas symulacji wykonywany był wykres jak w czasie zmienia się procent błędnie sklasyfikowanych punktów oraz mapa, gdzie te punkty się znajdowały. Wartość błędu była ułamkiem wyrażonym w procentach oznaczającym ilość błędnych klasyfikacji w obrębie całego okna na ilość dodanych nowych punktów, zatem wartość ta może przekroczyć 100 %.

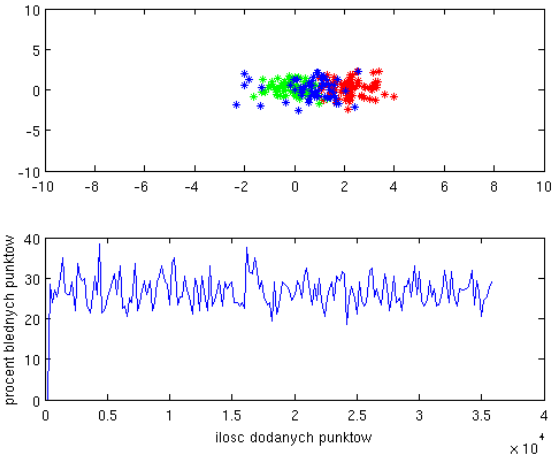
2.1 Metoda stałego okna

W metodzie kolejkowej dane, na których uczyła się sieć znajdowały się w kolejce. Kiedy do kolejki wchodził nowy punkt, najstarszy zostawał z niej usuwany. Długość kolejki mogła być regulowana.

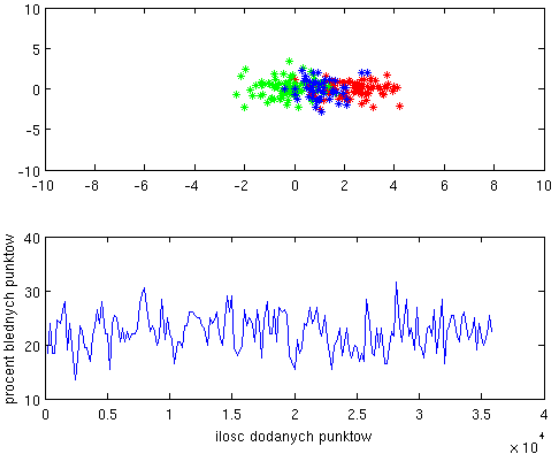
Na początku przeprowadzono testy dla stałego parametru wygładzania (wartość spread) na zestawie pierwszym. Liczbę wszystkich generowanych punktów ustawiono na 9000. Rozmiar okna wynosił 200 punktów. Symulacje przeprowadzono dla spread równego 0,001; 0,01; 0,1; 1, 10 oraz 15. Dla każdego testu sporządzona został mapa z błędnymi punktami w ostatniej fazie, wykres zależności błędu od czasu oraz średnia wartość błędu. Wyniki przedstawia rysunek 3 oraz tabela 1.



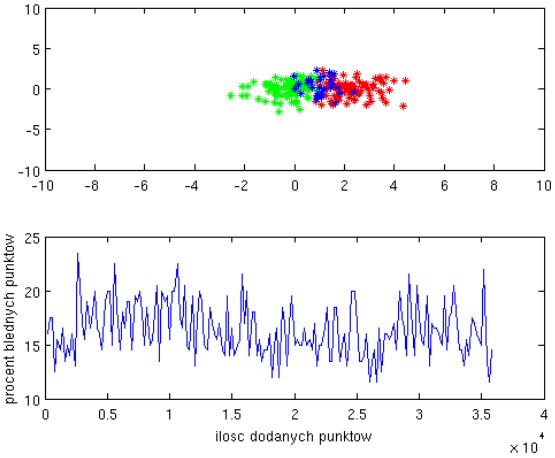
(a) h = 0,001



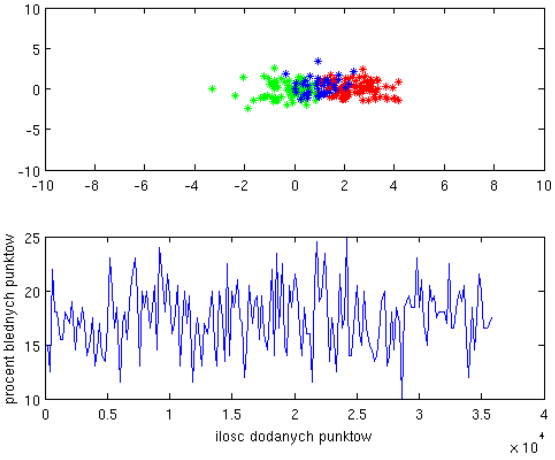
(b) h = 0,01



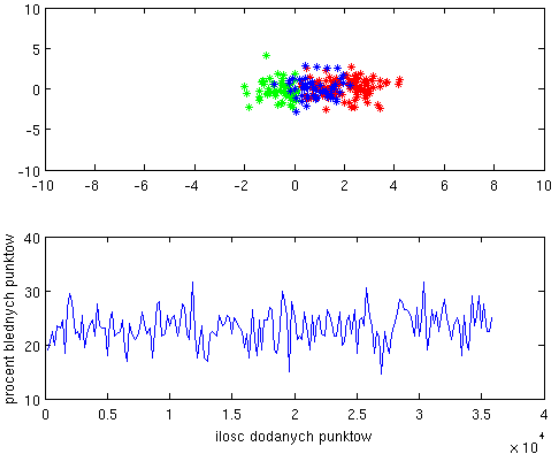
(c) h = 0,1



(d) h = 1



(e) h = 10



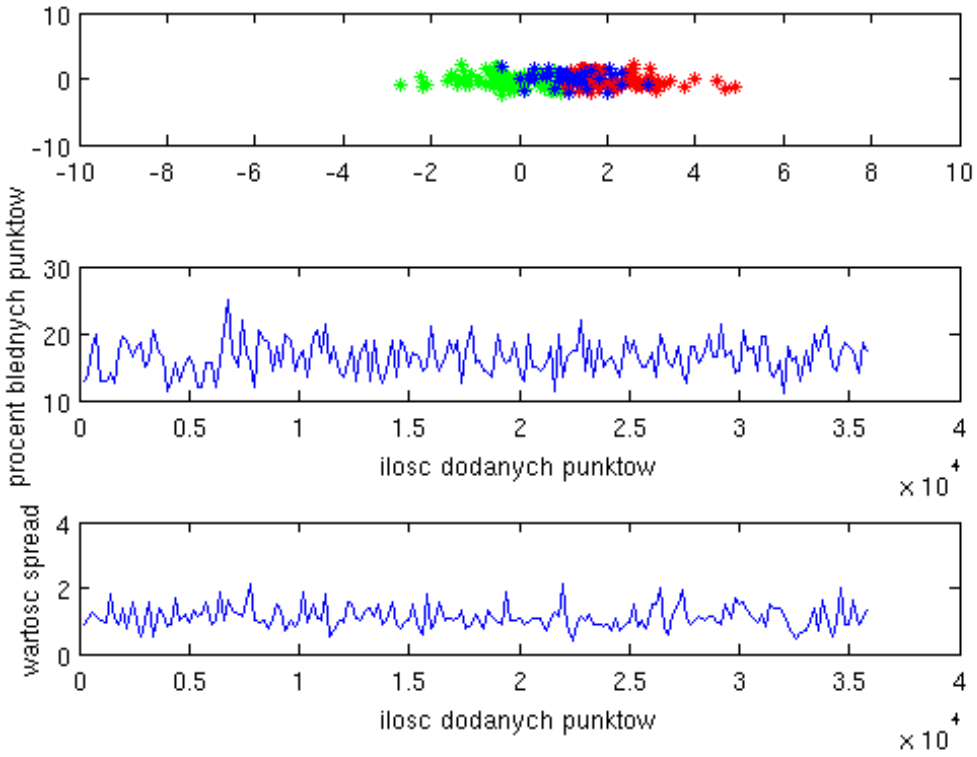
(f) h = 15

Rysunek 3: Test zestawu 1 dla różnych h

spread	błąd
0,001	48%
0,01	26%
0,1	22%
1	16%
10	18%
15	23%

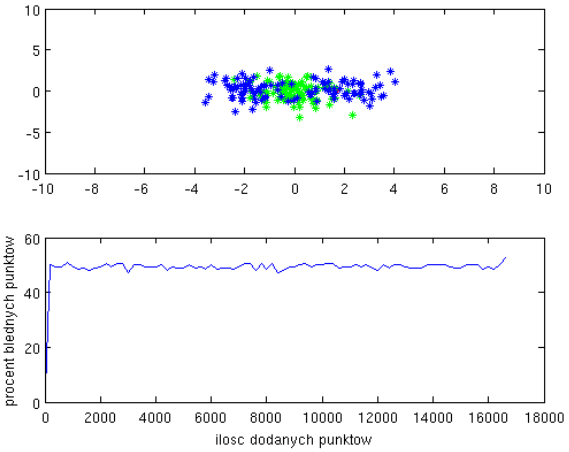
Tablica 1: Zestawienie wyników testów dla stałych wartości spread dla zestawu 1.

Następnie przeprowadzono symulację dla h - wartości spread liczonej automatycznie.
Parametry danych były jak w poprzednim przypadku. Rysunek **??** przedstawia otrzymane wyniki. Sporządzone wyniki zostały uzupełnione o wartość współczynnika wygładzenia zmieniającego się w czasie.

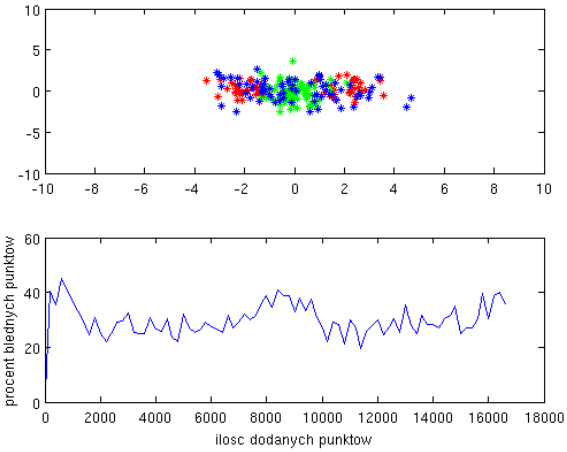


Rysunek 4: Symulacja dla spread liczonego automatycznie.

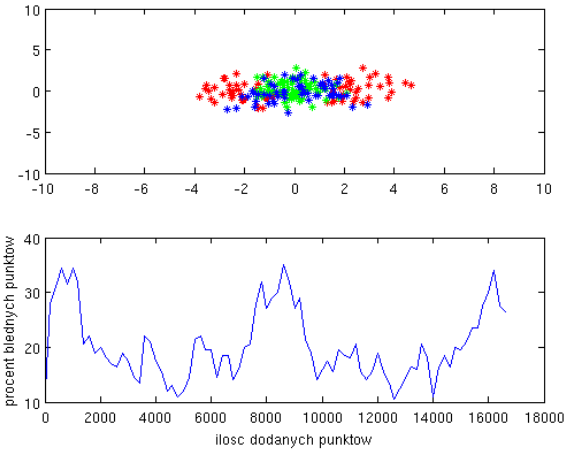
Średni błąd wyniósł 40 %.
Analogiczny test został przeprowadzony dla zestawu nr 2. Wyniki przedstawia rysunek 5 oraz tabela 2.



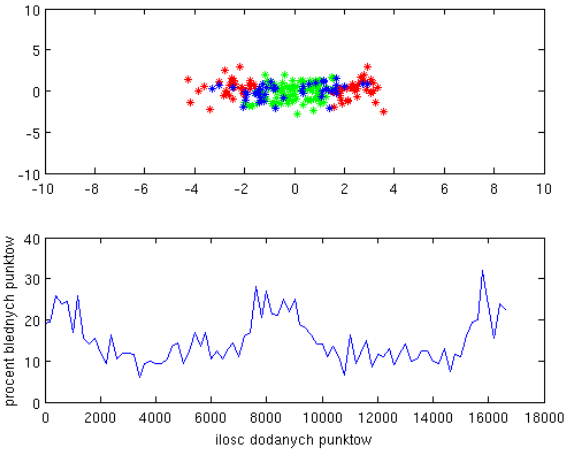
(a) h = 0,001



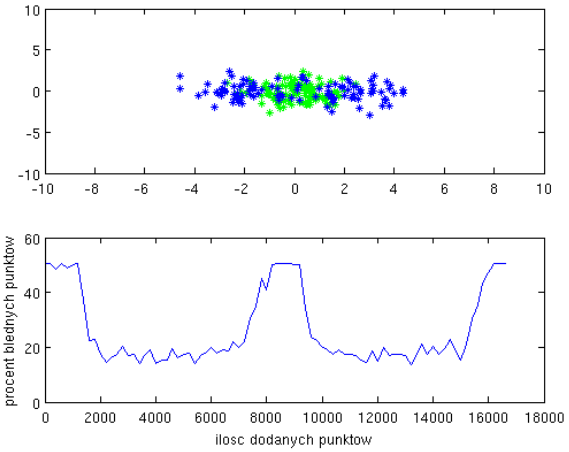
(b) h = 0,01



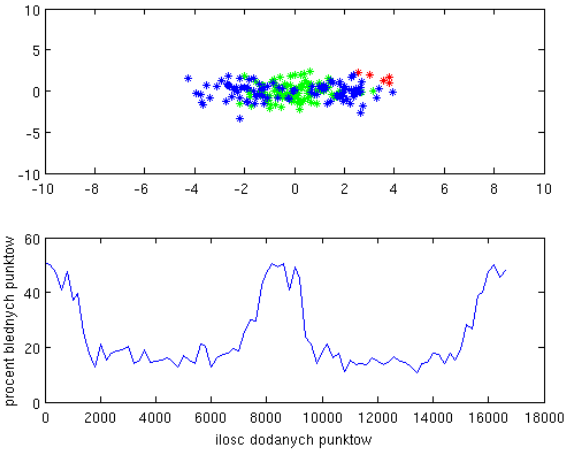
(c) h = 0,1



(d) h = 1



(e) h = 10



(f) h = 15

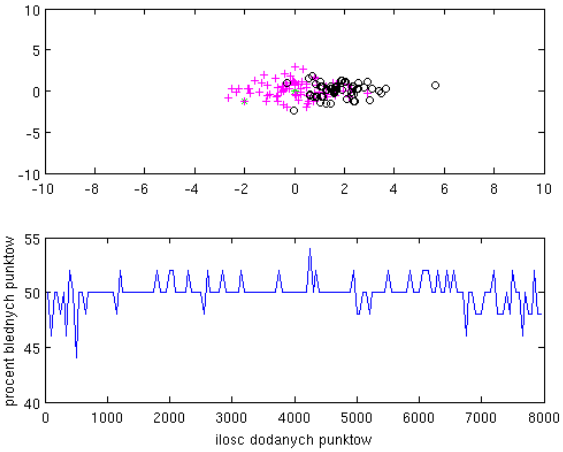
Rysunek 5: Test zestawu 2 dla różnych h

spread	błąd
0,001	48%
0,01	29%
0,1	20%
1	15%
10	26%
15	23%

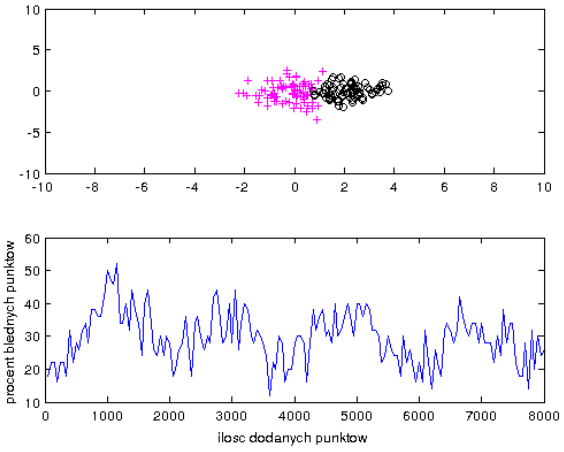
Tablica 2: Zestawienie wyników testów dla stałych wartości spread dla zestawu 2.

2.2 Metoda priorytetów

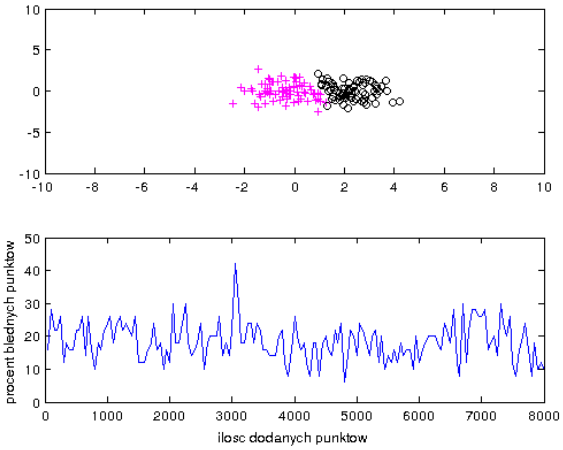
Analogiczne testy przeprowadzono dla metody priorytetowej. Polegała ona na tym, że punkty służące do uczenia sieci w kolejnych krokach były trzymane w buforze o stałej wielkości. Kiedy przychodził nowy punkt, liczony był jego priorytet na podstawie prawdopodobieństwa przynależności do danej klasy oraz jego wieku. Dla punktów w buforze aktualizowano wartość priorytetu, ponieważ punkty w buforze się starzały. Nowy punkt był dodawany do bufora. Następnie odbywało się sortowanie wg priorytetu i usuwany był punkt z najniższą jego wartością. Dla obu zestawów przeprowadzono symulacje dla h równego 0,001, 0,01; 0,1; 1, 10 i 100. W drugim zestawie wprowadzono modyfikację polegającą na rozbiciu klasy stacjonarnej na dwie klasy (o centrach w pkt. (-2,0) i (2,0). Szerokość okna wynosiła 50. Wyniki dla zestawu pierwszego przedstawia rysunek 6 oraz tabela 3.



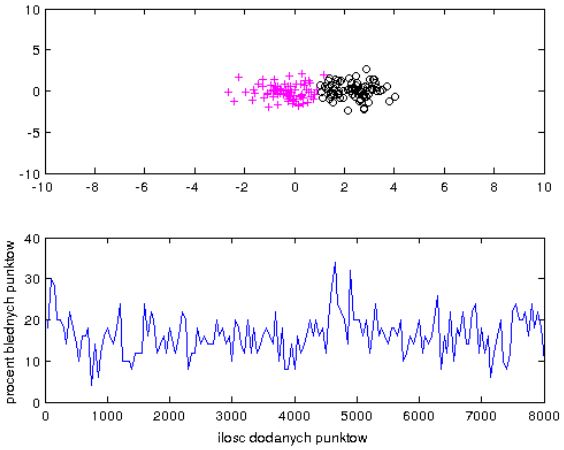
(a) h = 0,001



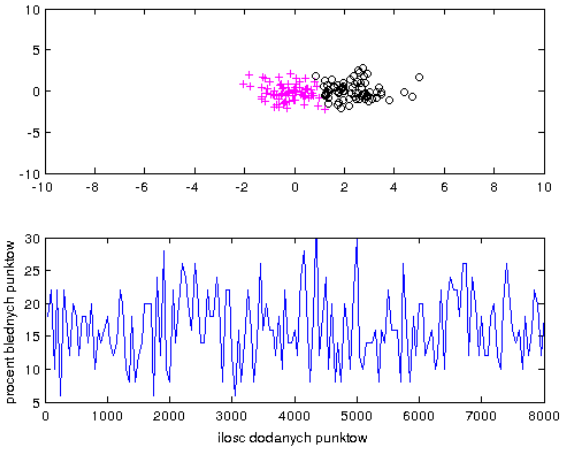
(b) h = 0,01



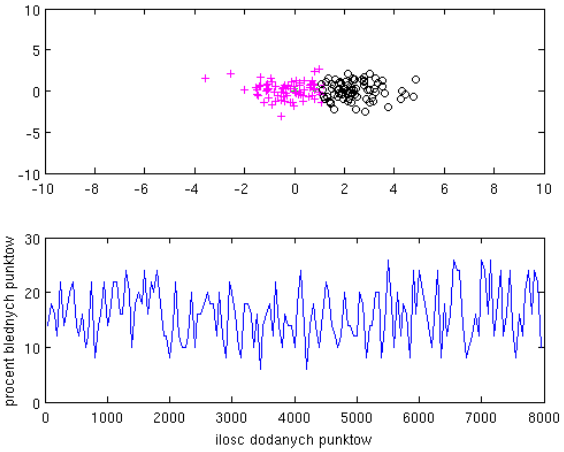
(c) h = 0,1



(d) h = 1



(e) h = 10



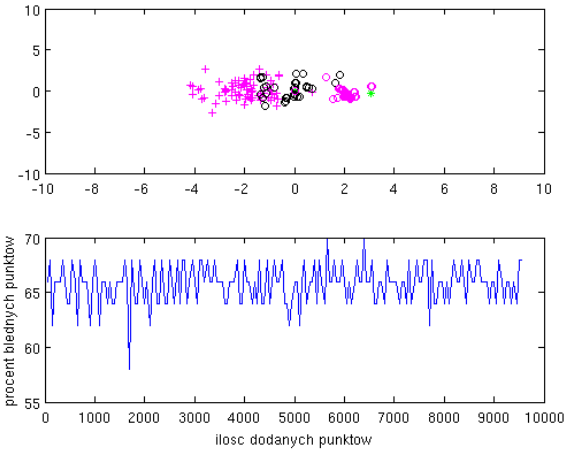
(f) h = 100

Rysunek 6: Test zestawu 1 dla różnych h

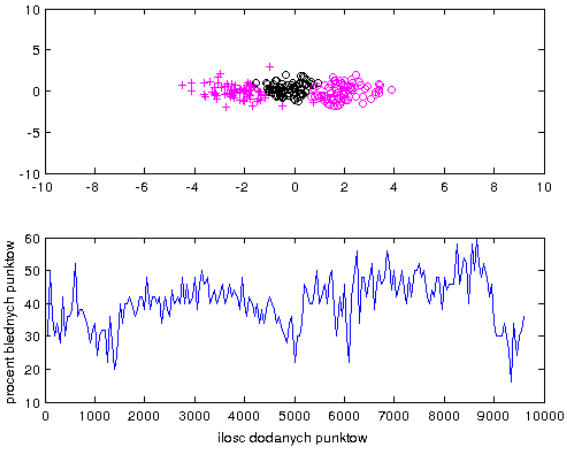
spread	błąd
0,001	50%
0,01	29%
0,1	18%
1	16%
10	17%
100	16%

Tablica 3: Zestawienie wyników testów dla stałych wartości spread dla zestawu 1.

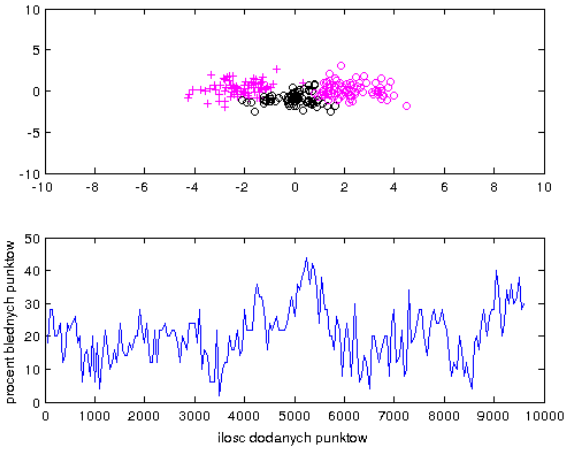
Wyniki dla zestawu drugiego przedstawia rysunek 7 oraz tabela 4.



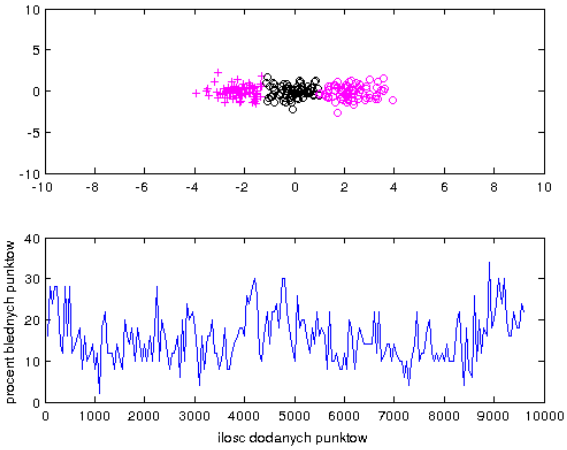
(a) h = 0,001



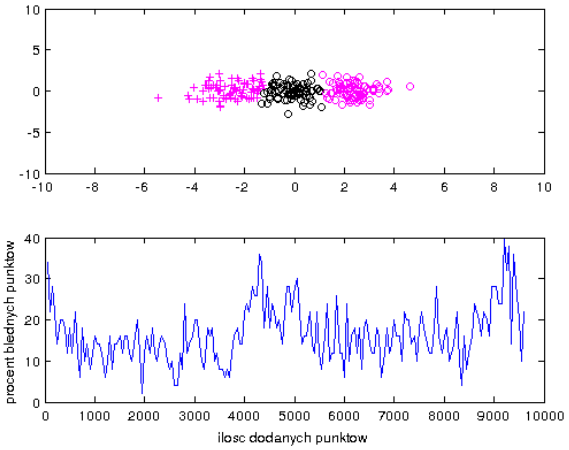
(b) h = 0,01



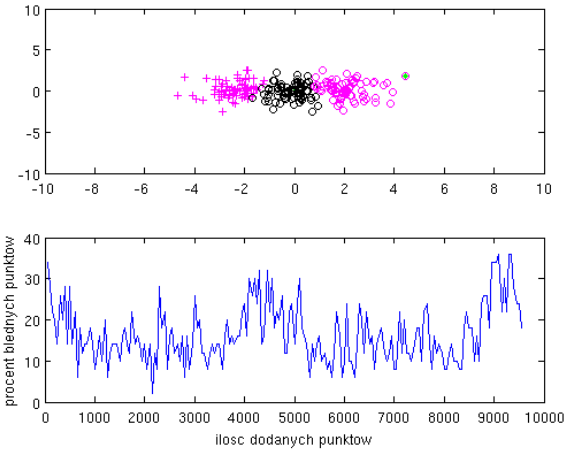
(c) h = 0,1



(d) h = 1



(e) h = 10



(f) h = 100

Rysunek 7: Test zestawu 2 dla różnych h

spread	błąd
0,001	66%
0,01	40%
0,1	20%
1	15%
10	17%
100	17%

Tablica 4: Zestawienie wyników testów dla stałych wartości spread dla zestawu 2.

3 Wnioski

Spoglądając na wyniki dla różnych wartości spread można zauważyć ciekawe zjawiska. Dla bardzo małej wartości błąd utrzymywał się na poziomie 50%. Wynika to z faktu, że sieć rozpoznawała tylko punkty nauczone w poprzedniej iteracji, gdyż zasięg funkcji bazowej był praktycznie zerowy. W miarę zwiększania parametru wygładzania, błąd się zmniejszał, by osiągnąć minimalną wartość dla spread równego 1. Kiedy wartość była dalej zwiększana, błąd rósł, ale tylko dla metody oknowej, ponieważ zasięg funkcji bazowych był tak duży, że nawet punkty wyuczone w poprzedniej iteracji były błędnie klasyfikowane. W metodzie priorytetowej ustawienie wartości spread nawet na bardzo duże wartości nie powodowało znaczącego pogorszenia wyniku. Było to spowodowane faktem, że metoda ta opierała się na priorytetach, więc nawet dla dużych wartości parametru wygładzania punkty leżące bliżej skupiska danej klasy miały większy priorytet. W czasie trwania symulacji nie zauważono znaczącego wpływu przemieszczania się punktów drugiej klasy w pierwszym zestawie na błędne sklasyfikowane punkty. W drugim zestawie dla obydwu metod wyło widać wyraźny wzrost błędu w momencie, kiedy krążąca klasa przechodziła przez środek układu współrzędnych, gdzie "było"wąsko. Należy przy tym zwrócić uwagę, że względny wzrost był dużo mniejszy przy metodzie priorytetowej. Działo się tak, ponieważ ta metoda wybierała tylko te punkty, co do których ma największą pewność - punkty leżące w środkowej części klasy nie zmieszane z dwoma pozostałymi.

Symulacja z automatycznym liczeniem współczynnika wygładzania działa podobnie jak dla optymalnie wybranego parametru h. Współczynnik oscylował w okolicach wartości 1.

Podsumowując, wiele zależy od danych, które otrzymujemy. Jeśli obie klasy były by wystarczająco oddalone od siebie, poziom błędów byłby dużo mniejszy. Na stworzonym przykładzie poziom błędów nie zmieniał się znacznie w czasie, co dobrze świadczyło o adaptacji sieci neuronowe. W kwestii współczynnika spread można albo ustawić metodami prób i błędów wartość na sztywno i liczyć, że z czasem wartość tego parametru będzie nadal aktualna lub użyć funkcji obliczającej automatycznie ten parametr i mieć pewność, że jakość działania sieci będzie bardziej adaptatywna.

4 Kod

Plik **method1_example1.m** generuje symulację dla metody kolejkowej i zestawu 1.
Plik **method1_example1_dynamic_spread.m** generuje symulację dla metody kolejkowej i zestawu 1 z automatycznym h.
Plik **method1_example2.m** generuje symulację dla metody kolejkowej i zestawu 2.
Plik **method1_example2_dynamic_spread.m** generuje symulację dla metody kolejkowej i zestawu 2 z automatycznym h.
Plik **method2_example1.m** generuje symulację dla metody priorytetowej i zestawu 1.
Plik **method2_example2.m** generuje symulację dla metody priorytetowej i zestawu 2.

W metodzie priorytetowej współczynnik spread ustawia się w pliku **considerPoint.m** zmienną sigma