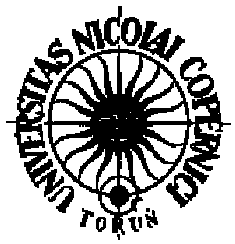
Maciej Orłowski

Wizualizacja działania sztucznych sieci neuronowych dla danych statycznych i dynamicznych

Praca magisterska pod kierunkiem
prof. Włodzisława Ducha



Katedra Metod Komputerowych
Wydział Fizyki, Astronomii i Informatyki Stosowanej
Uniwersytet Mikołaja Kopernika, Toruń 2002

Spis treści

Spis ilustracji

1. Wstęp

Sieci neuronowe dzięki dużemu postępowi technologicznemu w dziedzinie przetwarzania danych mają znaczące zastosowanie w wielu gałęziach nauki i przemysłu. Wśród szerokiego wachlarza zastosowań znaczącą rolę odgrywa klasyfikacja. Pozwala ona na rozpoznawanie ręcznego pisma, twarzy, czy na analizę wielkich zbiorów danych doświadczalnych np. w astronomii albo fizyce cząstek elementarnych. Innym zastosowaniem sieci neuronowych jest predykcja – przewidywanie wielkości makroekonomicznych, kursów walut itp., choć jeszcze nie udał się skonstruować modelu który generowałby konkretne zyski grając na giełdzie. Sztuczne sieci neuronowe wykorzystuje się także do sterowania skomplikowanymi robotami. Są używane również w systemach wspomagania decyzji.

Obecnie każdy przy odrobinie dobrej woli może sobie zamodelować i badać sieć neuronową na domowym komputerze. Dlatego też można się spodziewać dalszego dynamicznego rozwoju tej dziedziny.

Za pomocą obrazu można przekazać najwięcej informacji zdrowemu człowiekowi w porównaniu z innymi nośnikami bodźców. Ponadto obraz jest także łatwo przyswajany przez człowieka. W nowoczesnym marketingu rozwinęła się gałąź, która specjalizuje się w kreowaniu

graficznego wizerunku swoich klientów. Od wyborów prezydenckich w USA, gdzie decydującą rolę wygranej J. Kennedy'ego odegrały pierwsze debaty telewizyjne także politycy nie bagatelizują przekazu wizualnego. Zalety histogramów są od lat wykorzystywane w statystyce do szybkiej oceny ilościowej danych.

Jednym z zastosowań sieci neuronowych jest szeroko pojęta klasyfikacja danych. Zobrazowanie wyników działania klasyfikujących sieci jest celem niniejszej pracy.

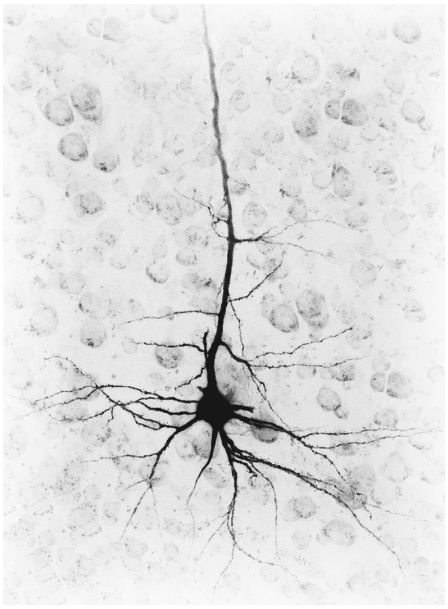
Rozdział drugi przedstawia krótki zarys podstaw teorii sieci neuronowych. Począwszy od pojęcia sztucznego neuronu, poprzez architekturę sieci typu MLP i RBF, na uczeniu skończywszy.

Tematem rozdziału trzeciego jest tytułowa wizualizacja. Zostały przedstawione różne sposoby prezentacji danych wielowymiarowych oraz zaproponowana została metoda wizualizacji danych podzielonych na klasy przez sieć neuronową.

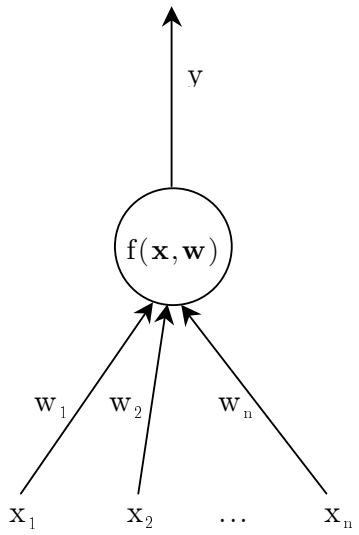
2. Sztuczne sieci neuronowe

2.1. Od neuronu biologicznego do sztucznej sieci neuronowej

Neuron biologiczny widoczny na rysunku 2.1 składa się z czterech zasadniczych części. Są to soma, dendryty, akson i synapsy.



Rysunek 2.1: Neuron biologiczny



Rysunek 2.2: Model sztucznego neuronu

Dendryty odbierają potencjał elektryczny od innych neuronów poprzez synapsy, w których na drodze chemicznej napięcie elektryczne może być

obniżone lub podwyższone. W somie zachodzi sumowanie potencjałów, dostarczonych przez dendryty. Jeśli sumaryczny potencjał przekracza pewien próg, neuron zostaje pobudzony i poprzez akson emituje swój potencjał elektryczny, który dochodzi do synaps innych neuronów. Po wzbudzeniu napięcie somy wraca do ustalonego potencjału spoczynkowego i czeka na następne wzbudzenie.

Sztuczny neuron (rysunek 2.2) posiada kilka wejść x_i i jedno wyjście y. Sygnał wyjściowy jest wynikiem działania na sygnały wejściowe funkcji aktywacji, zwaną też funkcją transferu.

$$y = f(\mathbf{x}, \mathbf{w}) \qquad (2.1)$$

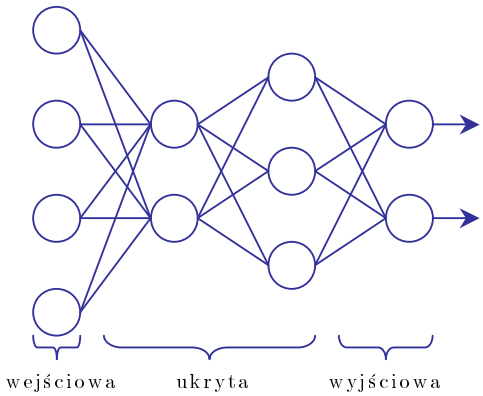
Pierwszy model sztucznego neuronu powstał na bazie swojego biologicznego odpowiednika. Zaproponowany został przez Warrena McCullocha i Waltera Pittsa w 1943 roku. W tym modelu funkcją aktywacji była funkcja progowa:

$$y = \begin{cases} 1 & dla \quad \sum_{i=1}^{n} w_i x_i + \theta \geq 0 \\ 0 & dla \quad \sum_{i=1}^{n} w_i x_i + \theta < 0 \end{cases} \qquad (2.2)$$

gdzie θ oznacza próg wzbudzenia neuronu.

Pierwotny model rozszerzono o funkcje różniczkowalne aby przy uczeniu można było wykorzystać metody gradientowe. Najczęściej spotykanymi są funkcja sigmoidalna, tangens lub funkcja Gaussa.

Neuron staje się bardziej użyteczny w połączeniu z innymi neuronami (węzłami) tworząc sieć neuronową.



Rysunek 2.3: Struktura sieci wielowarstwowej

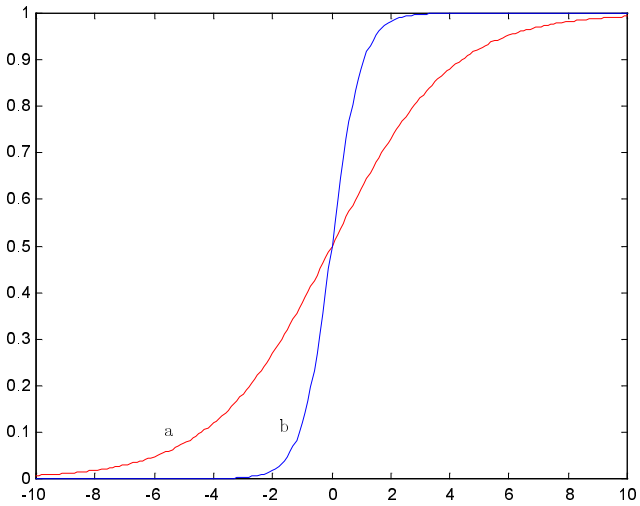
Neurony w sieci ułożone są w warstwy. Węzły warstwy wejściowej nie przetwarzają sygnału, służy ona tylko do wprowadzenia danych. Sygnał zostaje przetwarzany przez kolejne warstwy zwane ukrytymi, by następnie zostać ostatecznie zmodyfikowany przez neurony warstwy wyjściowej połączonych z wyjściem sieci.

Oprócz sieci jednokierunkowych istnieją jeszcze sieci rekurencyjne, w których sygnał z warstwy wyjściowej jest sygnałem wejściowym dla sieci w kolejnym cyklu.

2.2. Sieci MLP (Multi-Layer Perceptron)

Sieć typu MLP posiada co najmniej jedną warstwę ukrytą. Połączenia miedzy neuronami są pełne. Najczęściej stosowaną funkcją transferu jest sigmoida zwana też funkcją logistyczną.

$$y_i = \frac{1}{1 + exp\left(-\beta \sum_{j=1}^{n} w_{ij} x_j + \theta_i\right)} \qquad (2.3)$$



Rysunek 2.4: Wykresy jednowymiarowej funkcji logistycznej

a) $\beta = 0.5$, b) $\beta = 2$

Parametr β sigmoidy steruje jej skosem. Im wyższy tym bardziej stromy jest jej wykres (Rysunek 2.4). W skrajnych przypadkach funkcja

logistyczna przechodzi w funkcję liniową dla β=0 i funkcję skokową dla β>>0. Parametr θ oznacza próg wzbudzenia neuronu.

Liczbę węzłów w warstwie wejściowej określa ilość składowych wektora treningowego. Liczba warstw ukrytych i ich węzłów zależy od postawionego przed siecią zadania klasyfikacyjnego. Działanie pojedynczej funkcji logistycznej rozciąga się od określonego punktu w przestrzeni aż do nieskończoności.

SIGMOIDA 3D

Rysunek 2.5: Wykres dwuwymiarowej funkcji logistycznej

Przy zwiększeniu liczby warstw ukrytych i ich węzłów sieć pozwala na realizację nawet bardzo wyszukanych obszarów decyzyjnych.

Parametr β funkcji logistycznej można uzależnić od wartości wag i progu

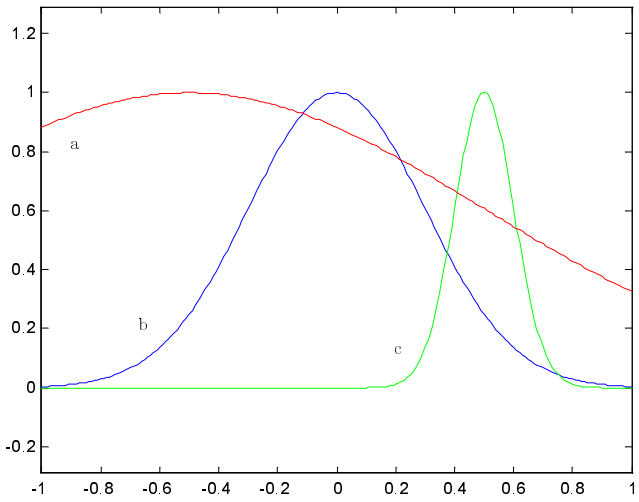
$$\beta' = \mathbf{w}'\mathbf{x} + \theta' \qquad (2.4)$$

więc aby uprościć obliczenia, w procesie uczenia wystarczy optymalizować wagi połączeń i próg przy stałym skosie równym 1.

2.3. Sieci RBF (Radial Basis Function)

Neurony sieci RBF charakteryzują się specyficznymi funkcjami aktywacji. Są to funkcje, których obszar zmienności jest symetryczny względem wybranego centrum. Wielowymiarowe funkcje w takiej postaci wydzielają z przestrzeni hipereliptyczne obszary. Typowym przedstawicielem funkcji radialnych jest funkcja Gaussa:

$$G(\mathbf{x}) = exp\left(-\frac{(\mathbf{x}-\mathbf{c})^2}{2r^2}\right) \qquad (2.5)$$

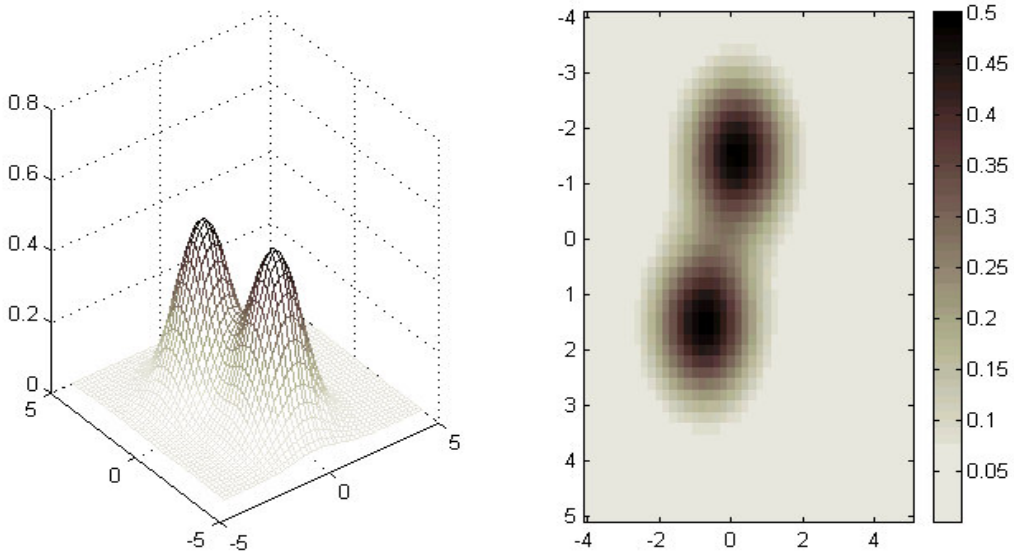


Rysunek 2.6: Wykresy jednowymiarowej funkcji Gaussa

a) r=1 c=-0.5, b) r = 0.3 c=0, c) r=0.1 c=0.5

W punkcie równym *c* funkcja osiąga wartość 1 a parametr *r* określa szerokość charakterystycznego kapelusza.

Warstwa wejściowa podobnie jak w sieciach MLP ma tyle węzłów ile składowych ma wektor treningowy. Ponieważ jedna wielowymiarowa funkcja Gaussa wydziela z przestrzeni zamknięty obszar, przy problemach klasyfikacyjnych wystarczy jedna warstwa ukryta, której n węzłów separuje n klas. Jednak często się zdarza, że obszar wyznaczony przez jedną funkcję nie wystarcza. Dokładność klasyfikacji można podnieść zwiększając liczbę neuronów ukrytych. Wówczas separowane obszary będą kombinacją kilku funkcji Gaussa.



Rysunek 2.7: Obszar decyzyjny wyznaczony przez kombinację dwóch dwuwymiarowych funkcji Gaussa.

Zadaniem warstwy wyjściowej sieci RBF jest ostateczne wagowe sumowanie sygnałów z warstwy ukrytej. Odpowiedzią wytrenowanej sieci na wektor testowy jest pobudzenie jednego z wyjść odpowiadającego danej klasie.

Podczas uczenia dobierane są nie tylko wagi połączeń ale również centra Gaussów i ich szerokości.

2.4. Uczenie sieci

2.4.1. Przygotowanie danych

Przed rozpoczęciem uczenia zwykle dane trzeba wstępnie przetworzyć ponieważ mogą być one niespójne, niejednorodne lub niekompletne.

Pierwszym krokiem jest digitalizacja. Sieć neuronowa wymaga by wszystkie dane były numeryczne. Wartości cech takich jak szerokość czy temperatura są liczbami. Jednak wartością cechy kolor jest np. ‘zielony’. Wszystkim danym symbolicznym muszą być przypisane wartości numeryczne w postaci liczb naturalnych bądź wektorów binarnych. (zielony=1, czerwony=2, niebieski=1; albo zielony=100, czerwony=010, niebieski=001).

Z rozmaitych powodów dane mogą być niekompletne. Niektórych cech nie można wyznaczyć, może zawieść aparatura pomiarowa o czynniku ludzkim nie wspominając. Sieć neuronowa wymaga jednak aby wektory wejściowe miały wszystkie składowe określone. Gdy liczba niekompletnych wektorów stanowi niewielką część całego zbioru można je odrzucić. Inną metodą jest obliczenie średniej na podstawie wszystkich wartości cech należących do danej klasy. Wykorzystując

bardziej zaawansowaną statystykę można znaleźć wartość najbardziej prawdopodobną w oparciu o znane wektory.

Dane opisują różne cechy, których wartości mogą mieścić się w różnych zakresach. W algorytmach klasyfikujących, w których prawdopodobieństwo jest określane n podstawie odległości taka niejednorodność danych prowadzi do złych rezultatów. Aby ujednolicić dane najczęściej stosuje się normalizację i standaryzację. W wyniku normalizacji nowe wartości cech należą do przedziału <0,1>. Niech **x** oznacza wektor wartości pewnej cechy o n składowych. Wówczas znormalizowaną wartość wyznacza się ze wzoru:

$$x_i^N = \frac{x_i - min(\mathbf{x})}{max(\mathbf{x}) - min(\mathbf{x})} \qquad (2.6)$$

Standaryzacja w odróżnieniu od normalizacji zachowuje rozkład wartości.

$$x_i^S = \frac{x_i - \bar{x}}{\sigma_x} \qquad (2.7)$$

gdzie

$$\bar{x} = \frac{1}{n}\sum_j^n x_j \qquad (2.8)$$

$$\sigma_x^2 = \frac{1}{n-1}\sum_j^n \left(x_j - \bar{x}\right) \qquad (2.9)$$

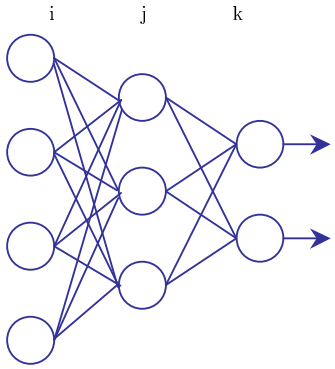
Po standaryzacji średnia nowych wartości cechy jest równa 0 a odchylenie standardowe $\sigma_x=1$.

2.4.2. Inicjalizacja

Wagi a w sieciach RBF dodatkowo parametry funkcji radialnych to parametry adaptacyjne. W kolejnych krokach uczenia parametry adaptacyjne są tak modyfikowane aby zapewnić jak najlepszą zgodność wartości wyjściowych z wartościami żądanymi, zawartymi w zbiorze treningowym. Uczenie zostaje przerwane gdy parametry te przestają się znacząco zmieniać co wcale nie świadczy o tym że ich wartości są optymalne, ponieważ proces mógł utknąć w minimum lokalnym. Wartości początkowe parametrów adaptacyjnych mogą więc wpływać na końcowy wynik. Aby wyeliminować takie sytuacje najczęściej stosuje się losowy wybór wag a uczenie powtarza się, co powoduje startowanie z różnych punktów. W sieciach RBF parametry funkcji radialnych można wstępnie dobrać analizując statystyczny rozkład wartości wejściowych.

2.4.3. Uczenie sieci MLP – algorytm propagacji wstecznej

Dla uproszczenia notacji przyjmijmy, że sieć ma jedną warstwą ukrytą. Niech indeks *i* oznacza węzły warstwy wejściowej, *j* warstwy ukrytej, *k* warstwy wyjściowej.



Rysunek 2.8: Sieć z jedną warstwą ukrytą

Do węzłów warstwy ukrytej dochodzi sygnał x_j i wychodzi sygnał y_j.

$$x_j = \sum_i w_{ij} x_i \qquad y_j = f(\sum_i w_{ij} x_i) \tag{2.10}$$

Całkowity sygnały wyjściowy z sieci y_K można więc zapisać:

$$x_k = \sum_j w_{jk} y_j \qquad y_k = f(\sum_j w_{jk} f(\sum_i w_{ij} x_i)) \tag{2.11}$$

Minimalizujemy funkcję celu E, która ma postać błędu średniokwadratowego:

$$E = \frac{1}{2}\sum_k (y_k - d_k)^2 = \frac{1}{2}\sum_k \left[f(\sum_j w_{jk} f(\sum_i w_{ij} x_i)) - d_k \right]^2 \tag{2.12}$$

gdzie **y** jest wektorem wyjściowym sieci a **d** wektorem wartości żądanych. Pochodna po wagach warstwy wyjściowej wynosi:

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial x_k}\frac{\partial x_k}{\partial w_{jk}} = (y_k - d_k) f'(x_k) y_j = \delta_k y_j \qquad (2.13)$$

gdzie δ interpretuje się jako sygnał błędu. Za pomocą obliczonej pochodnej można skorygować wagi warstwy wyjściowej:

$$w_{ij} \leftarrow w_{ij} - \eta \delta_k y_j \qquad (2.14)$$

η oznacza stałą uczenia, czyli czynnik regulujący jak bardzo ma się zmienić nowa waga w stosunku do poprzedniej wartości.

Pozostały jeszcze wagi warstwy ukrytej, do których poprawkę wyznacza się podobnie:

$$\frac{\partial E}{\partial w_{ij}} = \sum_k \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial x_k}\frac{\partial x_k}{\partial y_j}\frac{\partial y_j}{\partial x_j}\frac{\partial x_j}{\partial w_{ij}} = \sum_k (y_k - d_k) f'(x_k) w_{jk} f'(x_j) x_i = \delta_j x_i \qquad (2.15)$$

gdzie:

$$\delta_j = \sum_k w_{jk} \delta_k f'(x_j) \qquad (2.16)$$

W (2.15) widać zależność sygnału błędu warstwy j zależy od błędu warstwy k. Pokazuje to że sygnał błędu rozchodzi się odwrotnie do sygnału pobudzającego neurony sieci.

Powyższy wywód można uogólnić na dowolną liczbę warstw i zapisać wzorem:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \eta \nabla E(\mathbf{w}) \qquad (2.17)$$

gdzie k oznacza krok uczenia.

Sieć jest uczona wzorcami ze zbioru treningowego. Po prezentacji całego zbioru treningowego czyli po jednej epoce oblicza się całkowity sumując błędy poszczególnych wzorców (2.11). Uczenie zostaje przerwane gdy zmiany całkowitego błędu kolejnych epok są mniejsze od zadanego progu.

Algorytm propagacji wstecznej jest najprostszym z całej rodziny metod gradientowych. Uczenie może zostać przerwane po osiągnięciu minimum lokalnego funkcji celu. Algorytm jest także wolno zbieżny. Dlatego też używa się raczej bardziej złożonych i bardziej efektywnych metod.

2.4.4. Uczenie sieci RBF

Dla p wzorców uczących i warstwie ukrytej o m węzłach funkcja celu dla sieci RBF ma postać

$$E = \frac{1}{2}\sum_{i=1}^{p}\left(\sum_{j=1}^{m} w_j G_j(\mathbf{x}) - d_i\right)^2 + \sum_{j=1}^{m}\lambda_j w_j^2 \qquad (2.18)$$

gdzie λ jest parametrem regularyzacyjnym, który zapewnia gładkość funkcji błędu.

Wszystkie parametry adaptacyjne można wyznaczyć przez optymalizację funkcji (2.17) ze względu na wagi, centra i rozmycia funkcji bazowych.

Jednak takie podejście jest dość złożone obliczeniowo. W uczeniu sieci dla potrzeb wizualizacji zastosowano następująca metodę.

Wartości centrów funkcji radialnych są inicjalizowane przy użyciu niewielkiej liczby iteracji algorytmu K-średnich. Następnie są one optymalizowane algorytmem EM (Expectation Maximization), który dokładnie jest omówiony w pracy [11].

Rozmycia Gaussów są wyznaczane jako maksymalna kwadratowa odległość między centrami.

Wagi są wyznaczane przy znanych parametrach funkcji radialnych. Niech funkcje radialne mają postać (2.3), wektor **d** oznacza wartości żądane. Jak pokazano w [5] minimalizacja funkcji celu (2.17) prowadzi do układu równań liniowych, którego rozwiązaniem jest wektor wag **w**.

$$\mathbf{A}\mathbf{w} = \mathbf{H}^T\mathbf{d} \tag{2.19}$$

gdzie

$$\mathbf{H} = \begin{bmatrix} G_1(\mathbf{x}_1) & G_2(\mathbf{x}_1) & \cdots & G_m(\mathbf{x}_1) \\ G_1(\mathbf{x}_2) & G_2(\mathbf{x}_2) & \cdots & G_m(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ G_1(\mathbf{x}_p) & G_2(\mathbf{x}_p) & \cdots & G_m(\mathbf{x}_p) \end{bmatrix} \tag{2.20}$$

oraz

$$\mathbf{A} = \mathbf{H}^T\mathbf{H} + \mathbf{\Lambda} \tag{2.21}$$

Macierz Λ jest macierzą diagonalną z wartościami parametrów regularyzacyjnych na przekątnej.

2.4.5. Ocena jakości działania sieci

Dane składają się z par: wzorców $\mathbf{x}$ i wartości żądanych $\mathbf{d}$. Przed uczeniem sieci dane dzieli się na zbiór treningowy T i zbiór testowy C. Podczas uczenia na podstawie wartości żądanych $\mathbf{d}_T$ optymalizuje się parametry adaptacyjne. Jedną z zalet sieci neuronowych jest ich zdolność do uogólniania więc dobrze nauczona sieć powinna prawidłowo klasyfikować wzorce testowe, które wcześniej nie były jej prezentowane. Wytrenowana sieć generuje odpowiedzi na podane wzorce $\mathbf{x}_C$. Ocena klasyfikacji polega na porównaniu tych odpowiedzi z wartościami żądanymi $\mathbf{d}_C$ zawartych w zbiorze testowym.

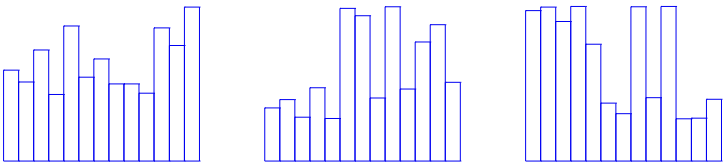
Jedną z technik pozwalających na ocenę jakości klasyfikacji jest kroswalidacja. W jednym kroku kroswalidacji dokonuje się losowego podziału bez powtórzeń na dane uczące i testowe. Po zakończeniu procesu uczenia wyznaczany jest na podstawie danych testowych błąd klasyfikacji. Kroswalidację zwykle powtarza się kilkunastokrotnie aby otrzymać średni wynik.

3. Wizualizacja

3.1. Sposoby wizualizacji bezpośredniej

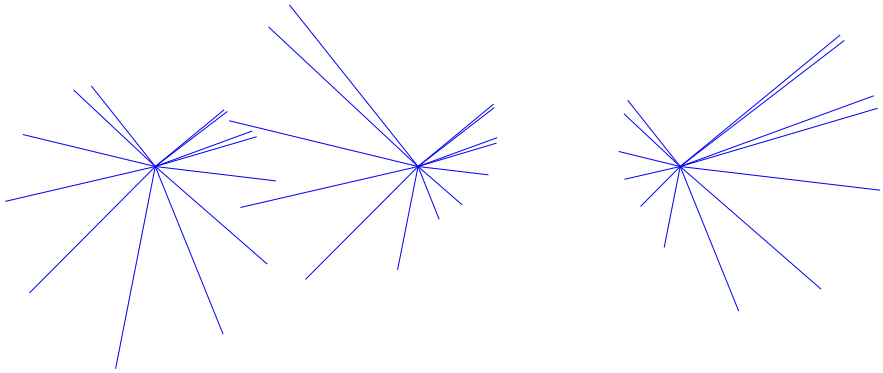
W wyniku wizualizacji bezpośredniej otrzymuje się wykres bądź figurę których poszczególne elementy przedstawiają wartości z każdego wymiarów.

Najprostszą metodą jest przedstawienie profilu wektora wielowymiarowego gdzie wartości kolejnego wymiaru pokazuje odpowiedni słupek.



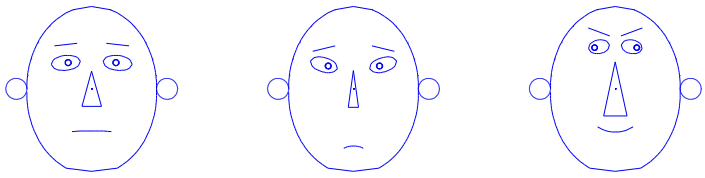
Rysunek 3.1: Profile

Innym sposobem jest przedstawienie wartości wielowymiarowych jako długości promieni wychodzących z jednego punktu.



Rysunek 3.2: Promienie

Jeszcze inną metodę zaproponował w 1973 roku Herman Chernoff. Wartości z wielu wymiarów wyznaczają rozmieszczenie, kształt i wielkość różnych części twarzy. Została tu wykorzystana naturalna zdolność do rozpoznawania szczegółów ludzkiej twarzy przez człowieka jako szybka metoda analizy i zapamiętywania wielowymiarowych danych.

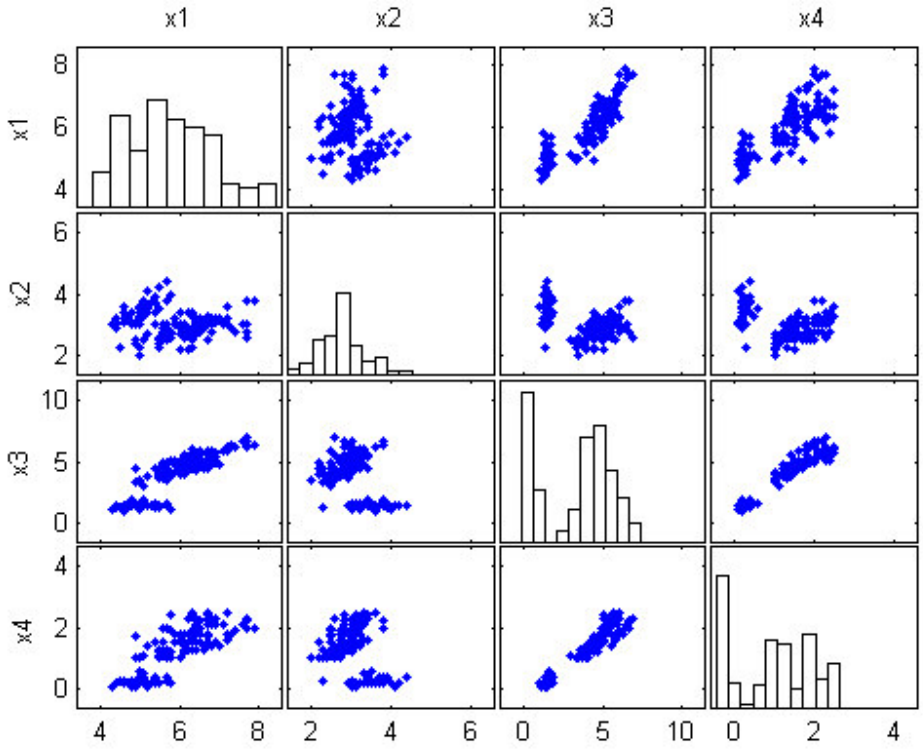


Rysunek 3.3: Twarze Chernoffa

Na powyższych przykładach zostały przedstawiono po 3 wektory 13 wymiarowe z bazy „Wine recognition data", należące do 3 różnych klas. Wektory zostały poddane standaryzacji. Obrazy otrzymano przy pomocy programu Systat [15].

Przedstawione dotychczas przykłady metod wizualizacji są przydatne do wizualizacji w sensie zestawienia pojedynczych przypadków. Trudno sobie bowiem wyobrazić analizę np. 1000 twarzy Chernoffa.

Innym podejściem jest metoda wykresu rozproszonego (scatter plot), gdzie może być przedstawiony nawet pokaźny zbiór danych wielowymiarowych. Wykres rozproszony przedstawiony na rysunku (3.4) przedstawia 150 wektorów z bazy „Iris Plants Database". Dodatkowo przekątna zawiera histogramy wartości z odpowiednich wymiarów.



Rysunek 3.4: Wykres rozproszony (scatter plot)

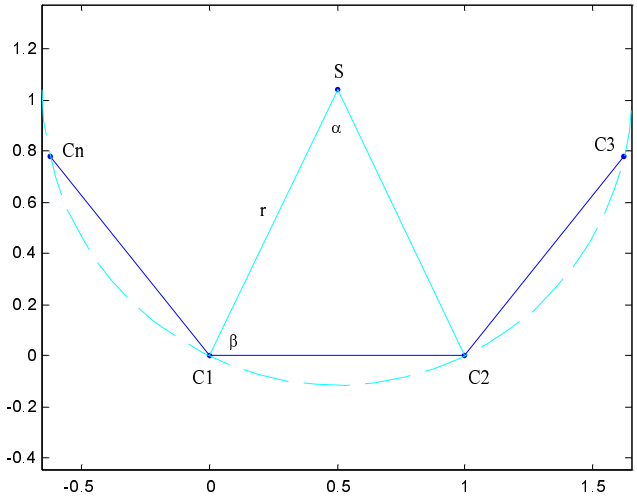
Na takim wykresie można łatwo zaobserwować zależności między wartościami poszczególnych wymiarów. Ciężko jednak wyciągnąć ogólne wnioski co do rozróżnienia całych wektorów.

3.2. Projekcja na 2 wymiary

Metody wizualizacji bezpośredniej mimo pewnych zalet trudno wykorzystać do prezentacji dużych zbiorów wielowymiarowych. Z pomocą przychodzą metody statystyczne takie jak skalowanie wielowymiarowe (multidimensional scaling) [4] czy grupowanie (klasteryzacja).

Wiedzę o danych można też pozyskać przez wizualizację działania sieci neuronowych. Wynikiem dobrze klasyfikującej sieci neuronowej jest wielowymiarowy wektor. Jeśli poszczególne klasy były zakodowane jako binarny wektor wartości żądanych, składowe wektora wyjściowego odpowiadają prawdopodobieństwom zakwalifikowania prezentowanego przypadku do odpowiedniej klasy. Czyli numer składowej bliskiej 1 odpowiada numerowi klasy do jakiej został zaliczony, podczas gdy reszta składowych jest bliska 0. Zbiór takich wektorów można przedstawić na płaszczyźnie jako punkty zgrupowane wokół wierzchołków wielokąta foremnego, gdzie każdy wierzchołek odpowiada jednej klasie.

Niech klasa 1 ma swój wierzchołek w punkcie (0,0), a następne klasy zgodnie z rysunkiem. Bok wielokąta jest równy 1.



Rysunek 3.5: Szkic do wyprowadzenia równań na wierzchołki wielokąta

Stosując prostą trygonometrię można uzyskać wzory na współrzędne wierzchołków n-kąta dla dowolnego n.

Wprowadzając oznaczenia:

$$\alpha = \frac{2\pi}{n}, \qquad \beta = \frac{\pi}{2} - \frac{\alpha}{2}, \qquad \varphi = -\frac{\pi}{2} - \frac{\alpha}{2},$$
$$r = \frac{1}{2\cos(\beta)}, \qquad x_S = \frac{1}{2}, \qquad y_S = \frac{1}{2}tg(\beta) \qquad (3.1)$$

otrzymujemy ogólne wzory:

$$x_i = x_S + r\cos(\varphi + i\alpha)$$
$$y_i = y_S + r\sin(\varphi + i\alpha) \qquad (3.2)$$
$$i = 0, 1, \ldots, n-1$$

Punkt (x_S, y_S) jest środkiem okręgu opisanego na wielokącie, a r promieniem. Kąt φ oznacza przesunięcie zapewniające aby pierwszy wierzchołek był w punkcie (0,0).

Rzutowanie n wymiarowego wektora Y_{nD} na płaszczyznę jest transformację liniową:

$$Y_{2D} = WY_{nD} + B \qquad (3.3)$$

Macierz W ma wymiar nx2, macierz B 2x1. Elementy macierzy W i B są rozwiązaniem układu 2n+2 równań liniowych:

$$A \cdot X = P$$

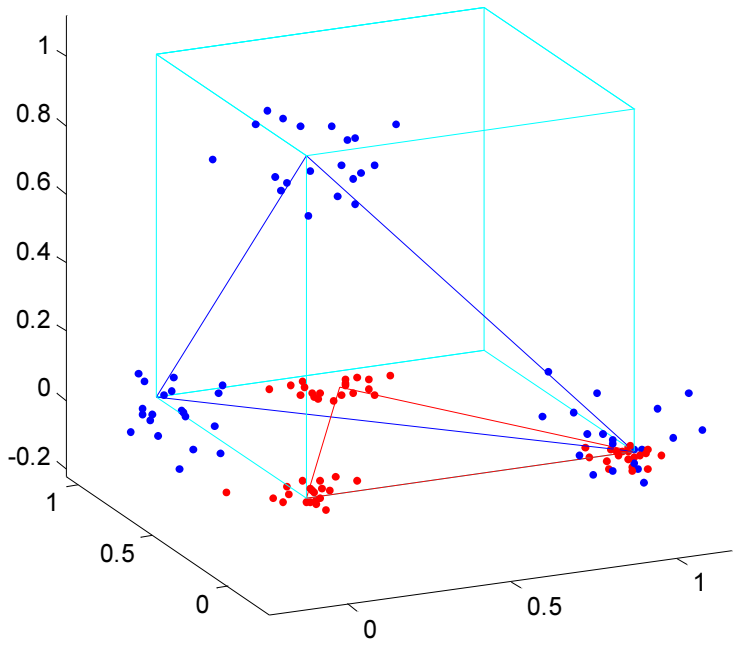
$$\left[\begin{array}{cccc|cccc|cc} 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ & & \ddots & & & & \ddots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & 1 & 0 \\ \hline 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & 0 & 1 \\ & & \ddots & & & & \ddots & & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & 0 & 1 \\ \hline 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 & 0 & 1 \end{array}\right] \cdot \begin{bmatrix} w_{11} \\ w_{12} \\ \vdots \\ w_{1n} \\ w_{21} \\ w_{22} \\ \vdots \\ w_{2n} \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ y_1 \\ y_2 \\ \vdots \\ y_n \\ x_S \\ y_S \end{bmatrix} \qquad (3.4)$$

Wyróżnione 4 podmacierze mają wymiar nxn, więc wymiar całej macierzy A wynosi 2n+2.

Równania od 1 do 2n powstały z rzutowania wierzchołków hipersześcianu odpowiadających klasom na wierzchołki wielokąta w

przestrzeni dwuwymiarowej. Dwa ostatnie odpowiadają rzutowaniu przeciwległego wierzchołka hipersześcianu do początku układu współrzędnych (wszystkie składowe równe 1) na środek dwuwymiarowego wielokąta.

Zastosowanie transformacji dla danych trójwymiarowych przedstawia rysunek (3.6).



Rysunek 3.6: Projekcja danych trójwymiarowych na płaszczyznę

3.3. Wizualizacja danych statycznych

Dane na których testowano wizualizację pochodzą z różnych źródeł. Wszystkie jednak przeszły proces wstępnego przygotowania. Pierwszym zabiegiem była zamiana numerów klas, bądź i etykiet na wektory binarne, gdzie numer składowej równej 1 oznaczał klasę. Niektóre ze zbiorów danych wymagały normalizacji. W przypadku brakujących danych niepełne wektory zostały usunięte.

Strukturę sieci dobrano testując różne architektury. Wybór najlepszej architektury zależał od wyników 10-krotnej kroswalidacji. W każdym jej kroku zbiory były dzielone losowo na wektory treningowe i testowe w stosunku 2/3, 1/3.

Następnie sieć o optymalnej strukturze jednokrotnie uczono a wyniki klasyfikacji poddano transformacji (3.3) i wykreślono różne klasy zaznaczając kolorami.

Aby zaznaczyć granicę poszczególnych klas, do wektorów wejściowych dodano niewielki szum o rozkładzie normalnym. Na tak zmodyfikowanych wektorach uczono sieć analogicznie jak w przypadku rzeczywistych wektorów. Wyniki zaznaczono na wykresach mniejszymi punktami.

Do uczenia sieci wykorzystano pakiet Netlab [14]. Neurony ukryte sieci MLP miały sigmoidalne funkcje aktywacji a RBF funkcje Gaussa.

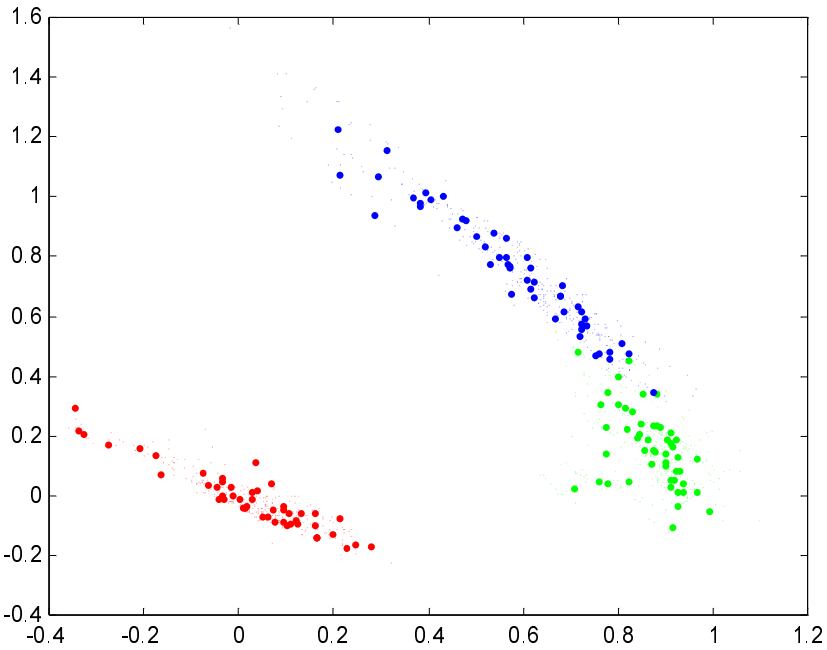
W procesie uczenia wagi sieci MLP były optymalizowane algorytmem sprzężonych gradientów [10]. W sieciach RBF metodą opisaną w podrozdziale 2.4.4.

3.3.1. Dane „Iris Plants Database"

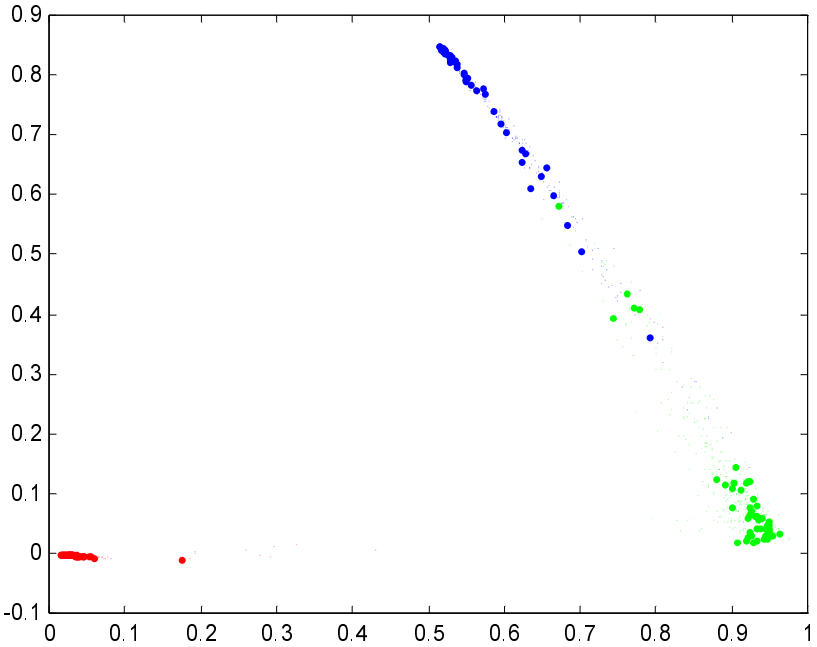
Dane zawierają pomiary trzech gatunków irysa: Iris Setosa, Iris Versicolour, Iris Virginica. Każdy wektor zawiera opis 4 cech: długość liścia, szerokość liścia, długość płatka i szerokość płatka podanych w centymetrach. 150 wektorów po 50 z każdej klasy.

Struktura i wyniki sieci	RBF	MLP
Wejścia	4	4
Neurony ukryte	6	5
Wyjścia	3	3
Poprawnie sklasyfikowane wektory	98%	98%

Z wykresów widać, że klasa 2 i 3 częściowo się nakładają. Klasa 1 jest bardzo dobrze oddzielona. Po analizie tak zaprezentowanych wyników działania sieci można stwierdzić, że gatunek Iris Setosa różni się znacznie wyglądem od dwóch pozostałych, które mają więcej cech podobnych.



Rysunek 3.7: „Iris Plants Database”, RBF



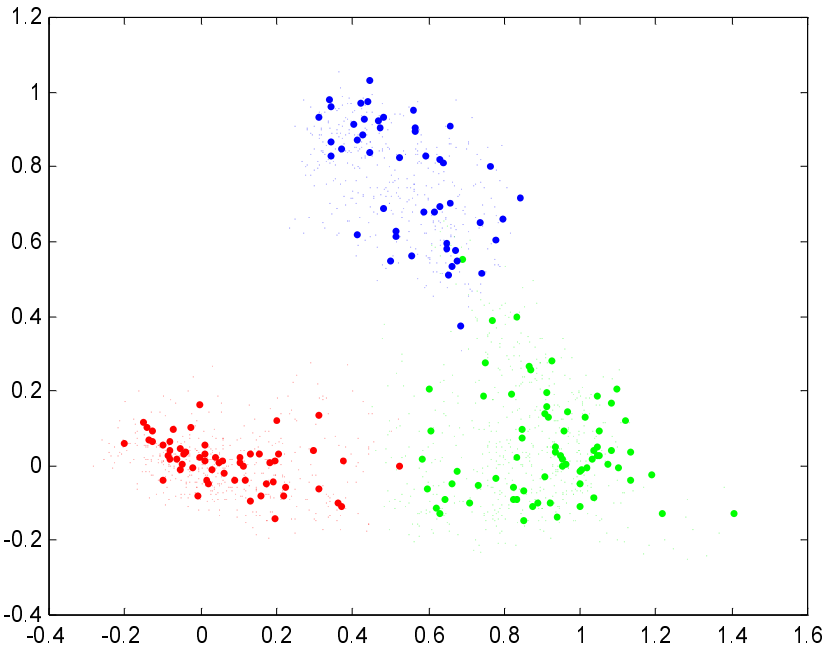
Rysunek 3.8: „Iris Plants Database”, MLP

3.3.2. Dane „Wine recognition data”

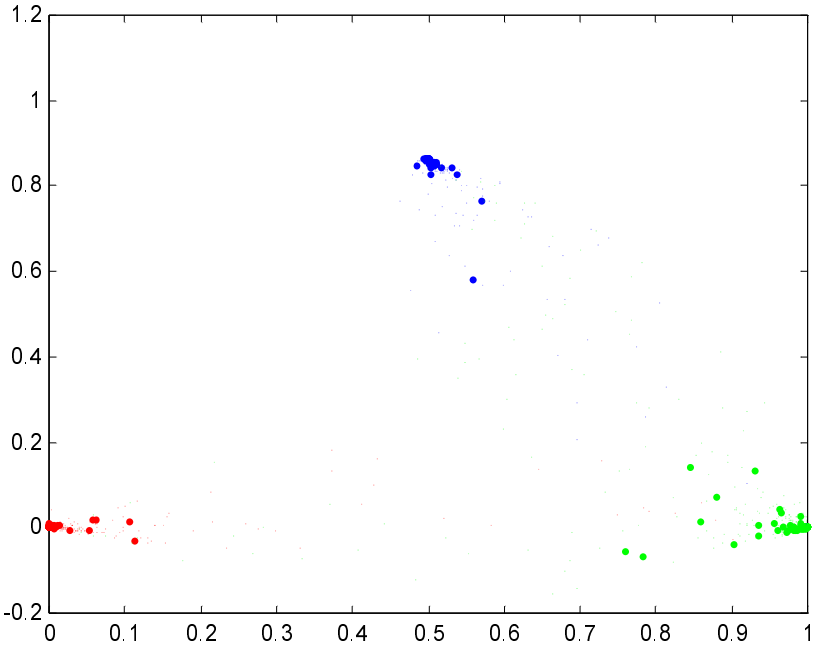
Dane pochodzą z analizy chemicznej 13 składników wina pochodzącego z różnych upraw z jednego regionu Włoch. Zawierają 175 wektorów podzielonych na 3 klasy.

Struktura i wyniki sieci	RBF	MLP
Wejścia	13	13
Neurony ukryte	5	6
Wyjścia	3	3
Poprawnie sklasyfikowane wektory	99%	100%

Wektory są bardzo dobrze oddzielone, na wykresach widać wyraźne granice między klasami.



Rysunek 3.9: „Wine recognition data”, RBF



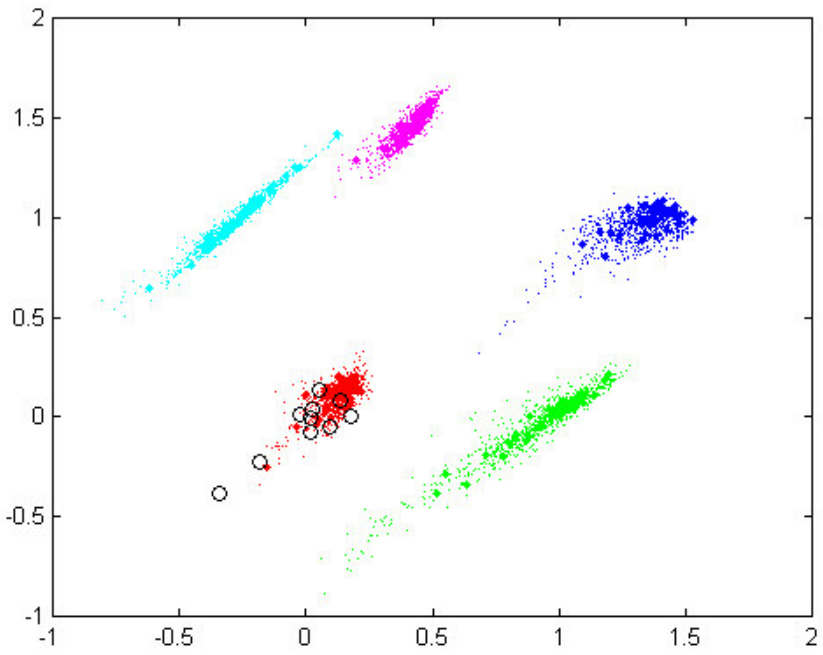
Rysunek 3.10: „Wine recognition data”, MLP

3.3.3. Dane sztuczne

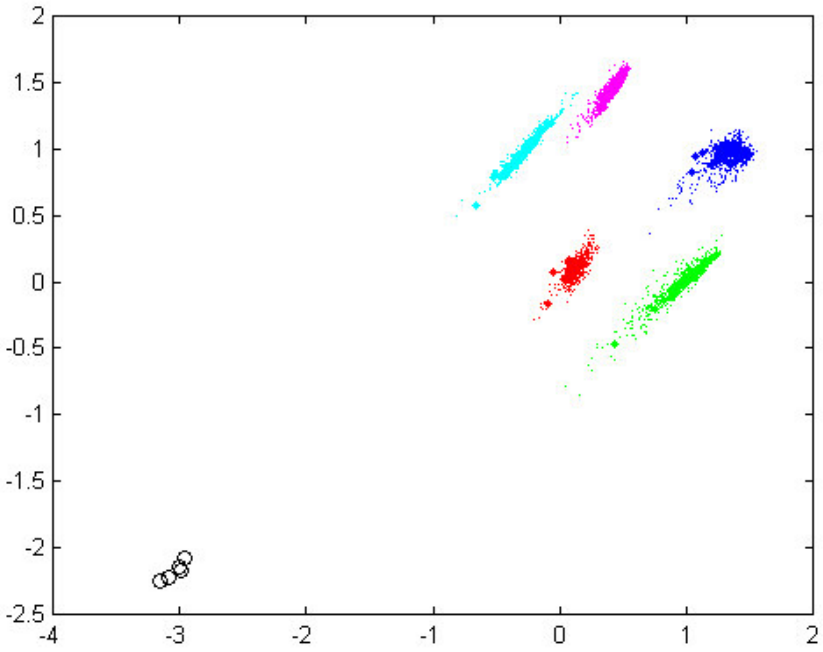
Sztuczne dane powstały z generacji losowych liczb z rozkładem normalnym. Wygenerowano 5 klas punktów należących do hipersfer umieszczonych w różnych miejscach przestrzeni czterowymiarowej, o promieniach z zakresu (0.2, 1). Zbiór zawierał 500 wektorów po 100 dla każdej z klas.

Struktura i wyniki sieci	RBF	MLP
Wejścia	4	4
Neurony ukryte	5	7
Wyjścia	5	5
Poprawnie sklasyfikowane wektory	99%	100%

Po projekcji wyników działania sieci na dwa wymiary poszczególne klasy rozlokowały się wokół wierzchołków pięciokąta. Po zakończeniu uczenia sieci dodatkowo zaprezentowano kilka wektorów należących do jednej z klas (3.11a) i kilka zdecydowanie nie należących do żadnej klasy (3.11b). W pierwszym przypadku zgodnie z oczekiwaniami wektory trafiły w obszar swojej klasy. W drugim przypadku punkty zdecydowanie opuściły obszar wielokąta.

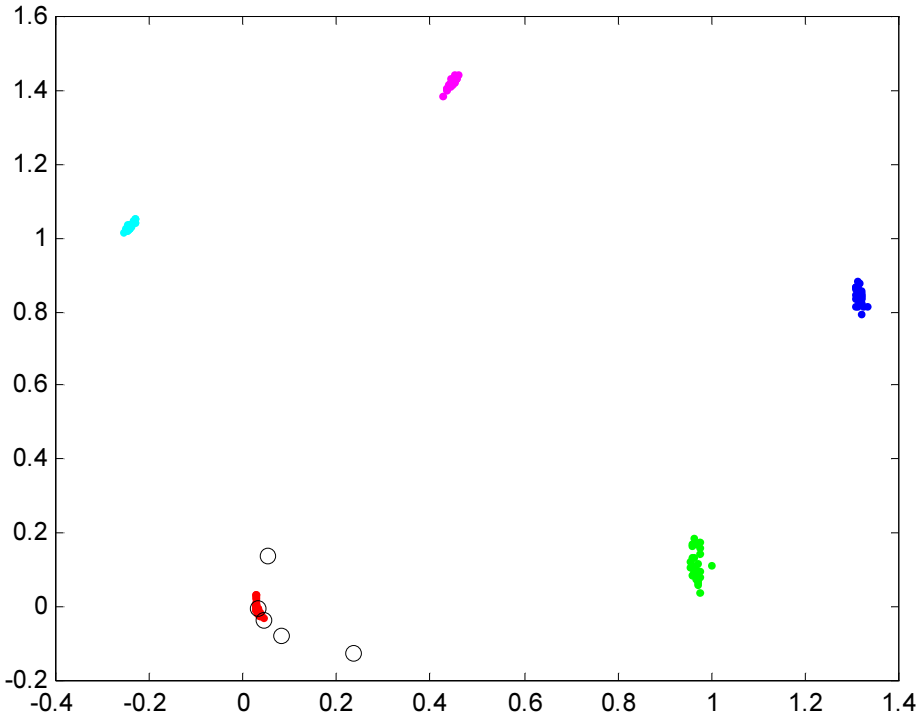


a)



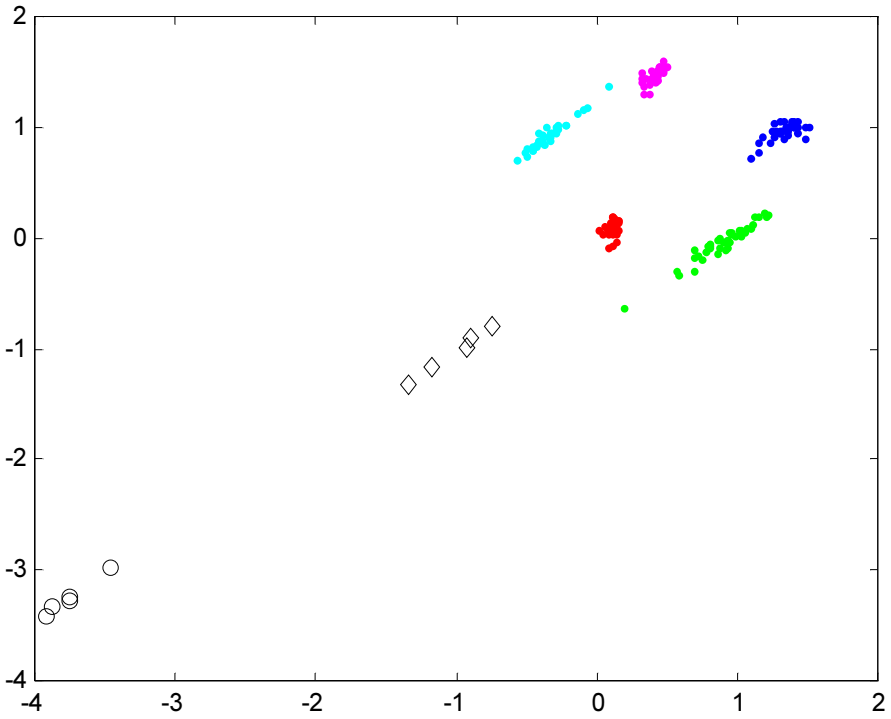
b)

Rysunek 3.11: Sztuczne dane, RBF, czarnymi kółkami zaznaczone nowe wektory a) należące do pierwszej klasy, b) nie należące do żadnej z klas



Rysunek 3.12: Sztuczne dane, MLP,
czarnymi kółkami zaznaczone nowe wektory należące do pierwszej klasy

Dla sieci MLP zrezygnowano z pokazania drugiego przypadku ponieważ zastosowano sieć o jednej warstwie ukrytej. Niezerowa wartość funkcji sigmoidalnej rozciąga się od określonego punktu w przestrzeni aż do nieskończoności więc dowolny wektor znacznie różniący się od reszty i tak znajdował się zawsze w obszarze którejś z klas. Zastosowanie sieci o większej liczbie warstw ukrytych powinno rozwiązać ten problem.



Rysunek 3.13: Sztuczne dane, RBF, dwie nowe klasy

Po nauczeniu, sieci RBF zaprezentowano dwa zestawy nowych wektorów. Na wykresie (3.13) widoczne są dwa nowe skupiska, które można interpretować jako nowe klasy nie wyodrębnione w procesie uczenia. Ma to znaczenie przy nie wystarczającej ilości wektorów treningowych.

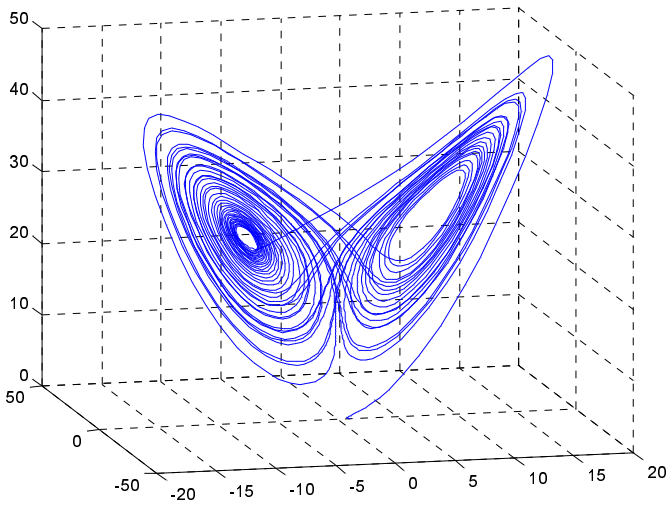
3.4. Wizualizacja danych dynamicznych

3.4.1. Atraktor Lorentza

Dane zostały uzyskane z rozwiązania równań [9]:

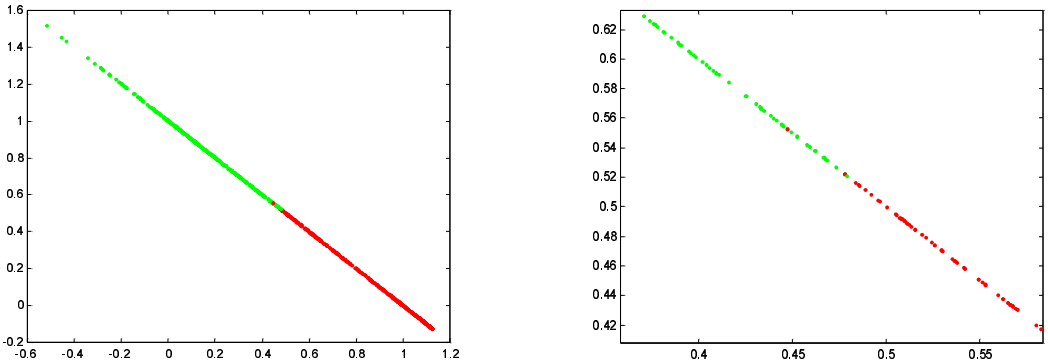
$$\dot{X} = \sigma(Y - X)$$
$$\dot{Y} = rX - Y - XZ$$
$$\dot{Z} = XY - bZ$$

z parametrami r = 28, σ = 10, b = 8/3. Rozwiązaniem równania jest trójwymiarowa trajektoria, która wykonuje nieregularnie pętle po lewej i po prawe stronie osi z.



Wygenerowano ponad 2000 wektorów, wśród których łatwo wyróżnić dwie 2 klasy.

Struktura i wyniki sieci	RBF	MLP
Wejścia	3	3
Neurony ukryte	2	3
Wyjścia	2	2
Poprawnie sklasyfikowane wektory	98%	99%



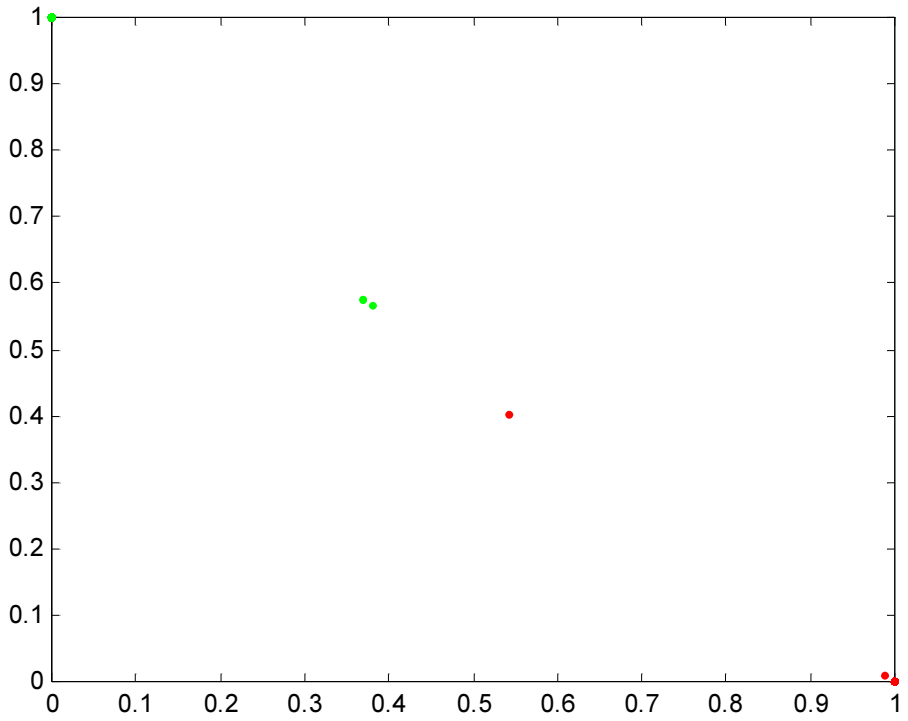
Rysunek 3.14: Lorentz, RBF, na prawym wykresie powiększenie

W przypadku danych dwuwymiarowych nie było potrzeby stosować projekcji na płaszczyznę. Zostały więc wykreślone wartości jakie zwraca sieć na wyjściu czyli prawdopodobieństwo, że wektor należy do danej klasy.

W przypadku sieci RBF (3.13) gdyby nie kolory nawet na powiększeniu nie dałoby się odróżnić przynależności danego wektora. Jest to spowodowane dużym zagęszczeniem wartości na granicy klas.

Ilość poprawnie klasyfikowanych wektorów wyznaczano przez porównanie maksymalnego wzbudzenia i odpowiadającej mu wartości żądanej stąd wyniki klasyfikacji 98%.

Natomiast sieć MLP bardzo wyraźnie podzieliła prezentowane wektory. Poza kilkoma przypadkami znakomita większość z ponad 2000 została zakwalifikowana z prawdopodobieństwem równym 1.



Rysunek 3.15: Lorentz, MLP

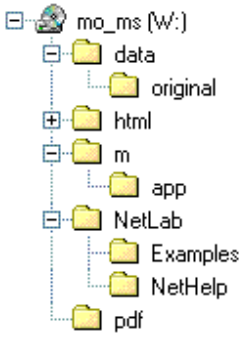
4. Podsumowanie

Na podstawie przedstawionych badań można stwierdzić, że zaproponowana metoda projekcji na dwa wymiary nadaje się do wizualizacji działania sieci neuronowej zarówno typu MLP jak i RBF. Transformację na 2D można przeprowadzać z dowolnej liczby wymiarów. Wyniki działania dobrze nauczonej sieci neuronowej można prezentować nawet na monochromatycznym wykresie nie tracąc informacji klasyfikacyjnych ponieważ wektory poszczególnych klas skupiają się wokół wierzchołków wielokąta. Można też łatwo odróżnić wektory niesklasyfikowane, które są zaznaczane jako punkty poza wielokątem. Są też zauważalne ewentualne nowe klasy nie wyodrębnione w procesie uczenia sieci RBF.

Metodę wizualizacji można rozwinąć o wyznaczenie granic klas.

Częścią pracy są zaimplementowane funkcję do programu Matlab [13] umożliwiające wizualizację wyników działania sieci neuronowej.

Dodatek A



Zawartość CD-ROM

/data	- użyte dane (przystosowane do obliczeń)
/data/original	- dane oryginalne
/html	- dokumentacja zaimplementowanych funkcji
/m	- implementacje na potrzeby wizualizacji
/m/app	- przykłady zastosowań
/NetLab	- NetLab © Ian Nabney, Christopher Bishop
/Netlab/NetHelp	- dokumentacja pakietu NetLab
/pdf	- praca w wersji elektronicznej (PDF)

Bibliografia

Literatura

[1] R. Adamczak: „Zastosowanie sieci neuronowych do klasyfikacji danych doświadczalnych", Praca doktorska, Uniwersytet Mikołaja Kopernika, Toruń, 2001.

[2] S. Cohen, N. Intrator: „Global Optimization of RBF Networks", Tel-Aviv University, URL: http://www.math.tau.ac.il/~nin

[3] S. Kaski: „Data exploration using self-organising maps", Praca doktorska, University of Technology, Helsinki, 1997,
URL: http://www.cis.hut.fi/~sami/thesis.ps

[4] A. Naud: „Neural and Statistical Methods for the Visualization of Multidimensional Data", Praca doktorska, Uniwersytet Mikołaja Kopernika, Toruń, 2000.

[5] M. J. L. Orr: „Introduction to Radial Basis Function Networks", University of Edinburgh, Edinburgh, 1996,
URL: http://www.anc.ed.ac.uk/~mjo/papers/intro.ps

[6] M. J. L. Orr: „Recent Advances in RBF Networks", University of Edinburgh, Edinburgh, 1999, URL: http://www.anc.ed.ac.uk/~mjo/papers/recad.ps

[7] S. Osowski: „Sieci neuronowe w ujęciu algorytmicznym", Wydawnictwo Naukowo-Techniczne, Warszawa, 1996.

[8] W. S. Sarle: „Neural Networks and Statistical Models", SAS Institute, 1994.

[9] H. G. Schuster: „Chaos deterministyczny", PWN, Warszawa, 1995, s. 110.

[10] J. R. Shewchuk: „An introduction to the Conjungate Gradient Method Without the Agonizing Pain", School of Computer Science, Carnegie Mellon University, Pittsburgh, 1994.

[11] M. Titsias, A. Likas: „A Probablistic RBF Network for Classification”, University of Ioannina

[12] D. R. Tveter: „Backpropagator's Review”, 2001, URL: http://www.dontveter.com/bpr/bpr.html

Użyte oprogramowanie

[13] Matlab, © Mathworks, URL: http://www.mathworks.com

[14] Netlab, © Ian Nabney and Christopher Bishop, URL: http://www.ncrg.aston.ac.uk/netlab

[15] Systat, © SYSTAT Software Inc., URL: http://www.systat.com