Uniwersytet Mikołaja Kopernika
Wydział Fizyki, Astronomii i Informatyki Stosowanej
Katedra Informatyki Stosowanej

Helena Jurkiewicz
numer albumu: 177622

Praca magisterska na kierunku fizyka komputerowa

Nieeuklidesowe sieci neuronowe

Opiekun pracy dyplomowej
prof. dr hab. Włodzisław Duch
Uniwersytet Mikołaja Kopernika

Toruń 2009

Pracę przyjmuję i akceptuję

_____

data i podpis opiekuna pracy

Potwierdzam złożenie pracy dyplomowej

_____

data i podpis pracownika dziekanatu

Spis treści

Rozdział 1

Wstęp

1.1 Sztuczne sieci neuronowe - informacje ogólne

1.1.1 Biologiczne i sztuczne sieci neuronowe

Inspiracją dla powstania sztucznych sieci neuronowych była niewątpliwie biologia. Biologiczna sieć neuronowa to system złożony z połączonych ze sobą pojedynczych komórek nerwowych - neuronów. Jeden neuron skojarzony jest średnio z 10^4 innymi neuronami [2]. Sygnały w sieci przekazywane są pomiędzy komórkami nerwowymi za pomocą impulsów elektrycznych. Neuron reaguje na podstawie wartości wszystkich wzbudzeń, które dotarły do niego w określonym przedziale czasu. Aktywacja neuronu i przekazanie sygnału dalej, następują jeśli potencjał pobudzenia jego błony komórkowej przekroczy pewien progowy poziom.

Model sztucznej sieci neuronowej można opisać jako strukturę złożoną z pewnych jednostek, zdolną do zdobywania, przetwarzania i wykorzystywania wiedzy. Sieć potrafi uogólniać zdobytą wiedzę na nowe przypadki. Budowa i zasady działania sztucznych sieci neuronowych były wzorowane na właściwościach sieci biologicznych. Poniżej wymieniono niektóre cechy sieci naturalnych, których odpowiedniki można odnaleźć w sieciach sztucznych [18]:

- Architektura sieci: sieć złożona jest z wielu, połączonych ze sobą, jednostek przetwarzających - neuronów.
- Budowa neuronu: dendryty i akson w węźle sieci biologicznej odpowiadają wielu wejściom i wyjściu sztucznego neuronu.
- Możliwość odbierania i przetwarzania wielu sygnałów jednocześnie.
- Sygnał wyjściowy - jest generowany na podstawie wartości sumy ważonej sygnałów wejściowych, może zostać przekazany dalej wielu węzłom sieci.
- Właściwości synapsy: sygnały wejściowe są modyfikowane poprzez nadanie wag odpowiednim synapsom neuronu, wagi te ulegają zmianie wraz z doświadczeniem.

- Działanie neuroprzekaźników: pobudzające lub hamujące.

Choć model sztucznego neuronu powstał w oparciu o neuron biologiczny, a pomiędzy sztucznymi i naturalnymi sieciami neuronowymi zauważyć można wiele analogii, to sieci wykorzystywane w zadaniach inteligencji obliczeniowej nie są w stanie naśladować skomplikowanych funkcji sieci naturalnych. Model sztucznych sieci jest zbyt prosty by odtworzyć złożoną strukturę i działanie sieci stworzonych przez przyrodę. Pomimo niemożności wymodelowania "syntetycznego mózgu", sztuczne sieci neuronowe wykorzystywane są w wielu dziedzinach. W następnym podrozdziale wymieniono niektóre najważniejsze przykłady zastosowań takich sieci.

1.1.2 Przykłady zastosowań sieci neuronowych

Sieci neuronowe mają wiele zastosowań, jednym z najbardziej przydatnych jest klasyfikacja. Prawidłowa segregacja złożonych wielowymiarowych danych często jest zadaniem przekraczającym możliwości człowieka, podczas gdy sztuczne sieci neuronowe radzą sobie z tym całkiem dobrze. Innymi odpowiednimi dla sieci zadaniami są: przetwarzanie sygnałów, aproksymacja, modelowanie matematyczne. Sztuczne sieci neuronowe mogą znaleźć zastosowanie w każdej dziedzinie, w której potrzebne jest rozwiązanie któregoś z powyższych zagadnień.

Sieci wykorzystuje się zarówno w biznesie, przemyśle jak i nauce. Znakomicie sprawdzają się w przetwarzaniu i analizie różnego rodzaju danych: fizycznych, geologicznych czy też medycznych, często są jednym ze składników systemów ekspertowych. W medycynie mogą służyć na przykład jako pomoc przy stawianiu diagnoz. Także ekonomia korzysta z możliwości sieci neuronowych w zagadnieniach takich jak modelowanie i prognoza zjawisk ekonomicznych. Za pomocą sieci bada się na przykład zachowania giełdy i rynków, podejmowane są próby badania zależności prowadzących do upadku przedsiębiorstw [16]. Kolejną dziedziną, w której można wykorzystać sztuczne sieci, jest astronomia. Sieci wykorzystywane mogą być między innymi do klasyfikacji galaktyk i w rozpoznawaniu typów widmowych gwiazd. Więcej o wykorzystaniu sztucznych sieci neuronowych w astronomii można przeczytać na przykład w [19],[17],[3].

Powyższe przykłady stanowią jedynie niewielką część całego spektrum problemów przy rozwiązywaniu których znajdują zastosowanie sztuczne sieci neuronowe.

1.2 Cel pracy

Niniejsza praca zajmuje się wielowarstwowymi jednokierunkowymi sieciami neuronowymi. Jej celem jest zbadanie działania i możliwości wynikających z zamiany funkcji aktywacji w takich sieciach ze standardowej na aktywację opartą o funkcję odległości, implementacja tak zmodyfikowanej sieci i porównanie jej działania ze zwykłą siecią wielowarstwową.

Ponieważ sieci w niniejszej pracy testowane są poprzez porównanie jakości dokonywanych przez nie klasyfikacji, w drugim rozdziale omówiono zagadnienie klasyfikacji oraz sposoby porównywania błędów klasyfikacji.

W następnej części pracy przedstawiono podstawowe pojęcia związane z sieciami wielowarstwowymi. Opisano: budowę neuronu, architekturę sieci oraz proces nauki.

Rozdział czwarty przedstawia podstawowy podział funkcji wyjścia i aktywacji wykorzystywanych w sieciach neuronowych oraz wprowadza termin nieeuklidesowych sieci neuronowych, jako szczególnego rodzaju sieci wykorzystujących funkcje oparte na odległości (sieci D-MLP). Dalsza część rozdziału prezentuje pokrótce kształty obszarów decyzyjnych możliwych do wygenerowania przez sieci D-MLP.

W kolejnym rozdziale omówiono szczegóły implementacji sieci, między innymi przedstawiono wykorzystane funkcje transferu, podano sposoby wstępnego przetwarzania danych.

Część szósta prezentuje wyniki działania nieeuklidesowej sieci neuronowej dla kilku zbiorów danych. Wyniki te porównano z analogicznymi rezultatami dla zwykłych sieci wielowarstwowych. W rozdziale podane są opisy zbiorów danych, tabelki porównawcze błędów oraz przykładowe granice decyzji.

Rozdział ostatni zamyka i podsumowuje pracę.

Rozdział 2

Zagadnienie klasyfikacji

2.1 Podstawowe pojęcia

Pod pojęciem klasyfikacji można rozumieć zagadnienie przydzielania obiektów do pewnych grup na podstawie ustalonych kryteriów. Poszczególne dane poddawane klasyfikacji nazywa się obrazami lub obiektami. Są one wektorami w wielowymiarowej przestrzeni cech (atrybutów), określających właściwości obiektu. Do danego obiektu na podstawie jego atrybutów może zostać przypisana etykieta określająca klasę. Klasa jest zbiorem wektorów o podobnych właściwościach. Elementy jednej klasy mogą się między sobą różnić, jednak własności na podstawie których dokonuje się klasyfikacji powinny być podobne w obrębie elementów tej samej klasy. Systemem klasyfikującym nazywa się pewien model teoretyczny, którego zadaniem jest odnalezienie jak najlepszego odwzorowania z wektora danych $\mathbf{x}$ na klasę c.

Zadaniem klasyfikatorów jest więc przydzielenie punktów do odpowiednich klas. Zazwyczaj klasyfikator ma do dyspozycji dwa zbiory danych: pierwszy, z informacją do której z klas należą poszczególne wektory, tzw. zbiór treningowy oraz drugi, zawierający nieprzypisane wektory, nazywany zbiorem testowym. Na podstawie informacji uzyskanych ze zbioru uczącego, klasyfikator dokonuje przypisania obiektów ze zbioru testowego do istniejących klas. Proces, podczas którego model klasyfikacyjny wykorzystuje informacje zawarte w zbiorze treningowym i na tej podstawie modyfikuje pewne swoje parametry tak, aby jak najlepiej klasyfikować dane, nazywa się nauką z nadzorem lub nauką z nauczycielem. Należy jeszcze wspomnieć o klasteryzacji, rozważanej jako klasyfikacja bez nadzoru. O klasteryzacji można mówić, gdy istnieje próbka niepogrupowanych danych i nie ma informacji o klasach na jakie mogłyby one zostać podzielone. Zadaniem klasyfikatora jest wtedy wyodrębnienie klas na jakie można podzielić te obiekty. W niniejszej pracy pojawi się jedynie pierwszy rodzaj klasyfikacji, a jako model klasyfikujący zostaną wykorzystane modele sieci neuronowych: MLP i DMLP.

2.2 Generalizacja

Bardzo ważnym i pożądanym przymiotem systemów klasyfikacyjnych jest zdolność do generalizacji. Generalizacja to umiejętność formułowania ogólnych prawidłowości na podstawie przeanalizowanych przykładów. Podczas nauki system może zbytnio "dopasować się" do danych treningowych, co często skutkuje niewystarczająco dobrym rozpoznawaniem klas w zbiorze testującym. Klasyfikator z dobrą generalizacją potrafi zaklasyfikować nie tylko obiekty ze zbioru treningowego, ale także, korzystając z informacji zdobytych podczas treningu, jest w stanie z zadowalającą dokładnością zaklasyfikować dane nie uczestniczące w procesie uczenia.

2.3 Granice decyzji

Wektor danych $\mathbf{x_i}$ o atrybutach $[x_{i1}...x_{in}]$ opisuje punkt w n-wymiarowej przestrzeni cech. System klasyfikujący dokonuje mapowania wektora $\mathbf{x_i}$ na odpowiadającą mu klasę c_l, wybierając jedną ze zbioru możliwych klas $\{c_1...c_d\}$. Klasyfikator określa zbiór prawdopodobieństw "a posteriori": $P\left(c_j|\mathbf{x_i}\right)$ $j = 1...d$, dla wektora $\mathbf{x_i}$. Prawdopodobieństwo "a posteriori" $P\left(c_j|\mathbf{x_i}\right)$ oznacza prawdopodobieństwo przynależności wektora $\mathbf{x_i}$ o określonych atrybutach $[x_{i1}...x_{in}]$, do klasy c_j. W idealnym przypadku jedno z prawdopodobieństw równe jest jedności, reszta przyjmuje wartość zerową. Najczęściej jednak rozkłady danych w przestrzeni cech dla różnych klas są takie, że dla danego wektora prawdopodobieństwa "a posteriori" przynależności do różnych klas przyjmują wartości niezerowe. Optymalny klasyfikator przypisuje wtedy obiekt klasie, dla której prawdopodobieństwo "a posteriori" dla obiektu jest największe. Prawdopodobieństwo przynależności obiektu do klasy realizowane jest w klasyfikatorach poprzez pewne funkcje nazywane funkcjami dyskryminacyjnymi $f_l(\mathbf{x})$, $l = 1, 2...d$. Można zapisać, że klasyfikator przydzieli obiekt $\mathbf{x_i}$ do klasy c_j jeśli $f_j(\mathbf{x_i}) > f_k(\mathbf{x_i})$ dla $j, k = 1, 2...d$ i $j \neq k$.

Obszarem decyzyjnym klasy c_j nazywa się taki fragment przestrzeni cech, dla którego każdy położony wewnątrz niego obiekt x_i przynależy do klasy c_j. Obszary decyzyjne oddzielone są od siebie hiperpowierzchniami zwanymi granicami decyzyjnymi. Granica pomiędzy obszarami decyzyjnymi klas c_m i c_n leży na powierzchni, której punkty z równym prawdopodobieństwem należą do klas c_m i c_n. Równanie takiej granicy decyzyjnej ma postać:

$$f_m(\mathbf{x}) - f_n(\mathbf{x}) = 0 \tag{2.1}$$

2.4 Jakość klasyfikacji

2.4.1 Miara błędu

Celem procesu klasyfikacji jest jak najdokładniejsze przydzielenie danych obiektów do odpowiednich klas. Jako miernik poprawności dokonanej klasyfikacji stosuje się zazwyczaj

współczynnik będący stosunkiem ilości poprawnie sklasyfikowanych punktów do ogólnej liczby wektorów danego zbioru:

$$P = \frac{N_p}{N} \tag{2.2}$$

Współczynnik błędu liczony jest jako stosunek liczby błędnie zaklasyfikowanych danych do ilości wszystkich obiektów w zbiorze:

$$B = \frac{N_b}{N} \tag{2.3}$$

2.4.2 Porównywanie i ocena modeli klasyfikujących

Często zachodzi potrzeba porównania i oceny różnych systemów klasyfikacji. Powszechnym sposobem komparacji takich systemów jest zestawianie i ocena błędów popełnianych przez nie podczas procedury klasyfikacji danych. Pamiętać przy tym trzeba, że porównywanie jakichkolwiek klasyfikatorów ma sens jedynie gdy porównuje się ich działanie na określonych rozkładach danych. Dany model dla pewnego problemu może dawać znakomite rezultaty, podczas gdy dla innych zagadnień będzie zawodził. Ponieważ ważną i pożądaną cechą modeli klasyfikacyjnych jest, opisywana wcześniej zdolność do generalizacji, ich testowanie powinno odbywać się na danych, które nie uczestniczyły w procesie nauki. Istnieje możliwość, że podczas treningu system zostanie "przeuczony" i, pomimo że bardzo dobrze rozpozna klasy punktów w zbiorze na którym się uczył, to generalizacja będzie niewystarczająca. Zastosowanie oddzielnych zbiorów do nauki i testowania, pozwala ocenić model klasyfikacyjny pod względem zdolności do generalizacji. Innym niebezpieczeństwem może być, np. nikła reprezentatywność zbioru treningowego w stosunku do ogólnego problemu. Błąd otrzymany podczas testowania modelu zależy zarówno od właściwości danego modelu, jak też od wad zbioru uczącego. Dobrze jest więc wykonać testy modelu na wielu podziałach i jako miernik jakości modelu uznać, np. średnią po wszystkich błędach. Wielokrotne testowanie jest zalecane także w przypadku, gdy sam proces nauki może zależeć od jakichś losowych czynników. Na przykład w standardowym algorytmie propagacji wstecznej w sieci neuronowej, dla której wagi inicjalizowane są losowo, funkcja kosztu podczas nauki niekoniecznie musi zmierzać do globalnego minimum. Przy takich samych pozostałych parametrach sieci, różna inicjalizacja wag może znacznie wpływać na końcowy wynik nauki, a więc także na wielkość popełnianego przez sieć błędu.

Istnieje wiele metod pomocnych w określaniu jakości działania klasyfikatorów. Część z nich zostanie wymieniona i krótko scharakteryzowana w dalszej części podrozdziału [18],[2].

Prosty test Czasami do klasyfikacji dostaje się zbiór danych, który został już wcześniej podzielony na część testową i treningową. Wystarczy wtedy wykorzystać podzbiór treningowy do nauki, a test przeprowadzić na przeznaczonej do tego części danych. Dla lepszego

oszacowania poprawności klasyfikacji dobrze jest powtórzyć naukę i testy wielokrotnie, a błąd policzyć jako średnią.

Kroswalidacja Kroswalidacja, zwana jest także sprawdzianem krzyżowym lub walidacją krzyżową. Zbiór danych dzieli się na m rozłącznych, możliwie równych podzbiorów. Każdy z nich staje się zbiorem testowym dla klasyfikatora trenowanego na zbiorze uczącym, złożonym z $m-1$ pozostałych części. Błąd modelu określa się licząc średnią błędów z tych m próbek. Im większe m, tym lepiej oszacować można jakość klasyfikatora, ale jednocześnie zbiory testowe stają się coraz mniejsze.

Leave-one-out Szczególnym przypadkiem kroswalidacji jest tzw. "leave-one-out", sytuacja gdy m równe jest liczbie wektorów danych N. Dostaje się wówczas N zbiorów treningowych zawierających $N-1$ obiektów i N próbek testujących, każdą składającą się tylko z jednego wektora. Leave-one-out stosuje się zazwyczaj do zestawów danych o małej liczbie obrazów.

Kroswalidacja stratyfikowana Kroswalidacją stratyfikowaną nazywa się kroswalidację, w której podczas podziału proporcje liczebności wektorów należących do poszczególnych klas w oryginalnym zbiorze są zachowane w każdej próbce. Jest to szczególnie ważne w sytuacji gdy w opracowywanym zestawie istnieją znaczne dysproporcje w liczebności poszczególnych klas. Stratyfikacji z oczywistych względów nie stosuje w leave-one-out.

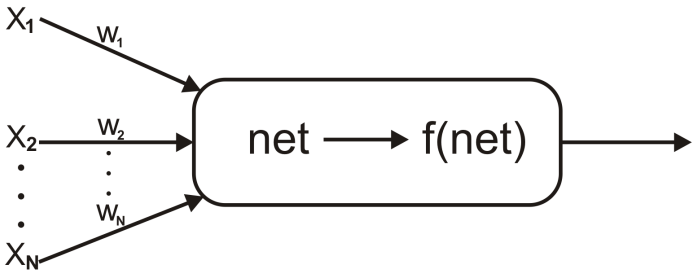
Rozdział 3

Sieci MLP

Sieć MLP (Multi Layer Perceptron) to wielowarstwowa, jednokierunkowa sieć neuronowa. W dalszej części pracy poprzez termin sieć neuronowa będzie rozumiana sieć MLP.

3.1 Neuron

Podstawowymi elementami z których składa się sieć neuronowa są neurony. Poniżej przedstawiono zaproponowany przez McCulloch'a i Pitts'a matematyczny model budowy i działania sztucznego neuronu.



Rysunek 3.1: Model sztucznego neuronu

Do "ciała" neuronu dochodzi wiele wejść ($\mathbf{x}$). Dane wchodzące na wejścia, mnożone są przez współczynniki skalujące ich wartość - wagi ($\mathbf{w}$). Wagi są parametrami adaptacyjnymi sieci, stanowią analogię do czułości biologicznego neuronu na impulsy pochodzące z poszczególnych wejść. Iloczyny wag i sygnałów wejściowych są sumowane i tworzą sygnał pobudza-

jący oznaczony jako *net*:

$$net = \sum_{i=0}^{N} w_i \cdot x_i \tag{3.1}$$

Gdzie N określa liczbę wejść neuronu, uzależnioną od wymiaru wektora danych wejściowych.

Do aktywacji neuronu często dodaje się tzw. próg , nazywany też biasem, oznaczony jako θ. Można go porównać do potencjału pobudzenia neuronu biologicznego. Po uwzględnieniu biasu pobudzenie ma postać:

$$net = \sum_{i=0}^{N} w_i \cdot x_i + \theta \tag{3.2}$$

Bias można zrealizować dodając do neuronu jeszcze jedno wejście i ustalając wartość sygnału wchodzącego na to wejście na 1, wtedy pobudzenie ma znów postać (3.1). W ciele neuronu pobudzenie *net* jest przetwarzane, dając w rezultacie sygnał wyjściowy. Do przetwarzania *net* wykorzystywana jest pewna funkcja, zwana funkcją wyjścia:

$$y = f(net) \tag{3.3}$$

W najprostszym przypadku funkcja ta może być tożsamościowa i dawać jak owynik aktywację *net*, częściej jednak wykorzystywane są bardziej skomplikowane funkcje.

3.2 Funkcje wyjścia

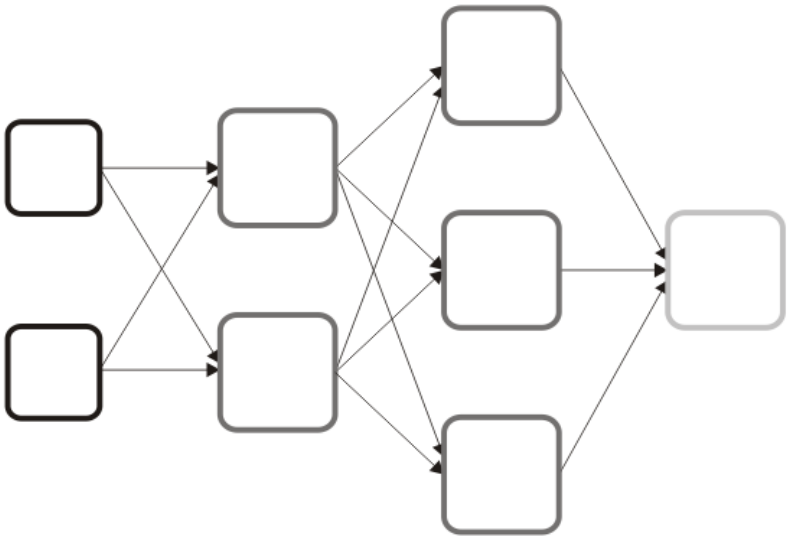
Funkcja wyjścia to funkcja określająca zależność między pobudzeniem neuronu *net*, a jego wyjściem y. W klasycznym modelu perceptronu McCullocha i Pittsa wykorzystywana była funkcja skokowa w postaci:

$$f(net) = \begin{cases} 1, & \text{gdy } net > 0 \\ 0, & \text{gdy } net \leqslant 0 \end{cases} \tag{3.4}$$

Obecnie jako funkcje wyjścia stosowane są najczęściej funkcje ciągłe, przyjmujące wartości pośrednie z zakresu $[0, 1]$. Przewagą tych funkcji nad (3.4) jest między innymi ich różniczkowalność, umożliwiająca wykorzystanie algorytmów gradientowych w nauce sieci. Typowym kształtem takiej funkcji jest sigmoida, a klasycznym przykładem - sigmoidalna funkcja unipolarna zwana też funkcją logistyczną. Jej wzór (5.5) oraz wykres (Rys. 5.1) znajdują się w dalszej części opracowania, w podrozdziale poświęconym wykorzystanym w pracy funkcjom transferu. Liczone od aktywacji (3.1) funkcje typu sigmoidalnego używane w sieciach MLP zaliczane są do tzw. funkcji nielokalnych. Taka pojedyncza funkcja dzieli przestrzeń danych wejściowych na dwa nieskończone obszary.

Funkcje transferu używane w sieciach MLP, stanowią rodzaj omawianych wcześniej funkcji dyskryminacyjnych. Ciągłe sigmoidy dzielą przestrzeń wejściową na obszary o określonym stopniu przynależności do danej klasy, binarne funkcje progowe dzielą przestrzeń kategorycznie, na część należącą lub nie do danej klasy, bez stopniowania przynależności.

3.3 Warstwy sieci neuronowej



Rysunek 3.2: Schemat sieci neuronowej: warstwa wejściowa reprezentująca wektor danych, 2 warstwy ukryte i jeden neuron jako warstwa wyjściowa.

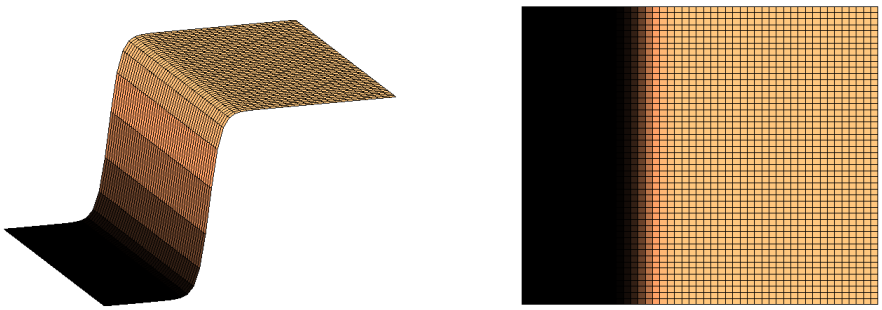
3.3.1 Budowa sieci

Architektura sieci określona jest poprzez liczbę i sposób połączenia składających się na nią neuronów. Najprostszym i szczególnym przykładem sieci neuronowej jest sieć utworzona z pojedynczego neuronu. Taka "sieć ", jest w stanie poradzić sobie z przydzieleniem danych do dwóch liniowo separowalnych klas, nie nadaje się jednak do rozwiązywania bardziej złożonych zagadnień. Większe możliwości dają sieci wielowarstwowe, gdzie wyjścia neuronów jednej warstwy połączone są z wejściami neuronów warstwy następnej. Każdy z neuronów danej warstwy ma ten sam wektor danych wejściowych. Wagi są indywidualne dla pojedynczego neuronu. Neurony należące do jednej warstwy nie są ze sobą połączone.
Zazwyczaj rozróżnia się warstwy :

- Wejściową - warstwa ta nie wykonuje obliczeń, a jedynie dostarcza sygnały pierwszej warstwie ukrytej.
- Wewnętrzną - może zawierać jedną lub więcej warstw ukrytych, o różnej ilości neuronów, wszystko zależy od złożoności rozważanego problemu.
- Zewnętrzną - tworzy wyjście sieci. Zazwyczaj ilość neuronów tej warstwy odpowiada ilości klas rozpoznawanych w danym zadaniu: każdy neuron odpowiada jednej klasie danych.

3.3.2 Kształty obszarów decyzyjnych realizowane przez MLP

Sieć składająca się z jednego neuronu tworzy hiperpłaszczyznę dzielącą przestrzeń na 2 części (Rys.4.8). Dodawanie do sieci nowych warstw zwiększa zakres odwzorowań, które sieć jest w stanie zrealizować. Neurony pierwszej warstwy ukrytej dzielą przestrzeń wejściową na pewne proste podprzestrzenie, kolejne warstwy tworzą bardziej skomplikowane fragmenty, w końcu warstwa zewnętrzna konstruuje finalne obszary decyzyjne sieci. W zasadzie sieć z dwoma warstwami ukrytymi, o odpowiednio wysokiej liczbie neuronów jest w stanie wygenerować obszary decyzyjne o dowolnym kształcie. Na rysunkach Rys. 3.4 i Rys. 4.10 można zobaczyć przykładowe obszary decyzyjne realizowane przez sieć z jedną warstwą ukrytą:



Rysunek 3.3: Obszar decyzyjny realizowany przez jeden neuron.

Rysunek 3.4: Przykładowy obszar decyzyjny dla sieci składającej się z dwóch neuronów i wyjścia.

Rysunek 3.5: Obszar decyzyjny możliwy do zrealizowania przez sieć składającą się z czterech neuronów w warstwie ukrytej oraz wyjścia.

3.4 Uczenie sieci neuronowych

Nauka sieci polega na minimalizacji funkcji kosztu (zwanej dalej także funkcją błędu). Minimalizacja ta jest realizowana poprzez odpowiednie modyfikacje wag neuronów. W zależności od wymagań, można używać różnych funkcji błędu. Powszechnie stosowaną i wykorzystaną w niniejszej pracy funkcją kosztu, jest funkcja sumująca kwadraty różnic pomiędzy wartościami wyjściowymi obliczonymi przez sieć, a wartościami wzorcowymi. Podczas uczenia, w każdej iteracji redukowana jest funkcja błędu dla kolejnych, poszczególnych obrazów wejściowych. Łączny błąd dla całego zbioru danych uczących, dla jednej epoki można przedstawić jako:

$$E = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=0}^{m} \left(y_j(\mathbf{x_i}) - d_{ji} \right)^2 \tag{3.5}$$

Gdzie n jest liczbą wektorów wejściowych uczących sieć, m liczbą wyjść sieci, $y_j(\mathbf{x_i})$ oznacza odpowiedź obliczoną przez sieć na j-tym wyjściu od wektora wejściowego $\mathbf{x_i}$, natomiast d_{ji} to oczekiwana odpowiedź neuronu. Do nauki sieci najczęściej wykorzystywane są różne procedury oparte na metodach gradientowych. Podstawową metodą nauki dla MLP jest metoda propagacji wstecznej błędu.

3.5 Metoda propagacji wstecznej błędu

Metoda wstecznej propagacji błędu zaliczana jest do metod gradientowych. Minimalizacja funkcji błędu wiąże się z liczeniem gradientów po parametrach adaptacyjnych (wagach) i zmianie wag w kierunku przeciwnym do obliczonego gradientu. Obliczanie gradientu wymaga, aby funkcja błędu, a co za tym idzie także funkcje transferu dla neuronów były różniczkowalne.

3.5.1 Reguła delta

Poniżej wyjaśniono sposób nauki oparty o tzw. regułę delta dla sieci jednowarstwowej. Wagi w procesie uczenia modyfikowane są iteracyjnie. Wzór na współczynnik o jaki dokonuje się korekcja wag ma postać:

$$\mathbf{\Delta w_i} = -\eta \bigtriangledown E_l(\mathbf{w_i}) \tag{3.6}$$

Gdzie η to współczynnik uczenia - parametr dobierany w zależności od problemu, E_l to błąd jednego (l-tego) obrazu wejściowego, $\mathbf{w_i}$ to wektor wag dla i-tego neuronu. Rozpisując k-tą składową gradientu i uwzględniając to, że błąd zależy od wag poprzez aktywację, można dojść do poniższego wzoru:

$$\frac{\partial E_l}{\partial w_{ik}} = \frac{\partial E_l}{\partial net_i}\frac{\partial net_i}{\partial w_{ik}} \tag{3.7}$$

Gdzie w_{ik} jest wagą k-tego wejścia, i-tego neuronu w danej warstwie, a net_i to aktywacja i-tego neuronu. Błąd jest zależny od aktywacji poprzez funkcję transferu. Rozwijając pierwszą część prawej strony wzoru (3.7), dostaje się wyrażenie określane jako δ:

$$-\delta_{li} = \frac{\partial E_l}{\partial net_i} = \frac{1}{2}\frac{\partial}{\partial net_i}\left(f_{li} - d_{li}\right)^2 = (f_{li} - d_{li})f'(net_i) \tag{3.8}$$

Indeks l przy d_{li} i f_{li} ma podkreślać, że są to wartości dla l-tego obrazu wejściowego. $f'(net_i)$ oznacza pochodną funkcji transferu po pobudzeniu i-tego neuronu. Nie jest to zaznaczone wprost, ale oczywiście pobudzenie jest w tym wypadku liczone od wartości l-tego obrazu. Drugi człon prawej strony równania (3.7) równy jest wartości k-tej składowej wektora wejściowego:

$$\frac{\partial net_i}{\partial w_{ik}} = x_k \tag{3.9}$$

Podsumowując: wzór na korekcję k-tej wagi, i-tego neuronu, w j-tym kroku, dla jednego wektora wejściowego, w jednej warstwie można zapisać:

$$w_{ik}^{(j+1)} = w_{ik}^{(j)} - \eta\left(f_i - d_i\right) f'(net_i)x_k \tag{3.10}$$

Powyższy przykład można łatwo uogólnić dla bardziej rozbudowanej sieci. Na regule delta bazuje, mający zastosowanie dla sieci wielowarstwowych, algorytm propagacji wstecznej błędu.

3.5.2 Algorytm propagacji wstecznej błędu

Wzory dla sieci wielowarstwowej są analogiczne do wzorów dla sieci jednowarstwowej. Różnica polega na tym, że funkcja błędu liczona jest tylko dla ostatniej warstwy, gdyż wzorcowe wartości sygnału znane są tylko dla wyjścia sieci, nie zaś dla wyjść poszczególnych warstw wewnętrznych. Nazwa propagacja wsteczna błędu oznacza, że do obliczenia błędu (w sensie δ) danej warstwy trzeba najpierw określić błąd warstwy, która znajduje się przed nią. Czyli najpierw liczy się wartości na wyjściu sieci, a na ich podstawie określa błędy warstw wcześniejszych cofając się aż do warstwy wejściowej.

Niech $\delta_i^{(s)}$ oznacza sygnał delty dla i-tego neuronu s-tej warstwy, a cała sieć kończy się na warstwie o numerze s+1, wzory są odpowiednie dla jednego, danego wektora wejściowego(w poprzedniej sekcji zaznaczał to indeks $_l$, teraz dla czytelności pominięty):

$$-\delta_i^{(s)} = \frac{\partial E}{\partial net_i} = \frac{\partial E}{\partial f_i^{(s)}} \frac{\partial f_i^{(s)}}{\partial net_i} = \frac{\partial E}{\partial f_i^{(s)}} f'(net_i) \tag{3.11}$$

$f_i^{(s)}$ określa wartość wyjścia i-tego neuronu s-tej warstwy i jest tym samym jedną ze składowych wektora wejścia $\mathbf{f^{(s)}}$ dla warstwy s+1.
Rozwijając (3.11) dalej dostaje się:

$$\frac{\partial E}{\partial f_i^{(s)}} = \frac{1}{2} \frac{\partial}{\partial f_i^{(s)}} \sum_{j=0}^{m} \left[f^{(s+1)} \left(net_j(\mathbf{f^{(s)}}) \right) - d_j \right]^2 = \tag{3.12}$$
$$= \sum_{j=0}^{m} \left[(f_j^{(s+1)} - d_j) f'(net_j) \frac{\partial net_j}{\partial f_i^{(s)}} \right] = - \sum_{j=0}^{m} \delta_j^{(s+1)} w_{ji}$$

Delta używana do obliczania poprawki dla warstwy s wykorzystuje deltę liczoną w warstwie s+1, tak samo delta dla warstwy s-1 korzysta z delty warstwy s itd. Ostatecznie delta dla i-tego neuronu warstwy s wynosi:

$$\delta_i^{(s)} = f'(net_i) \sum_{j=0}^{m} \delta^{(s+1)} w_{ji}^{(s+1)} \tag{3.13}$$

Zmiana wagi k-tego wejścia i-tego neuronu:

$$\Delta w_{ik}^{(s)} = -\eta f_k^{(s-1)} \delta_i^{(s)} \tag{3.14}$$

Algorytm nauki polega na iteracyjnym poprawianiu wag dla neuronów sieci przy pomocy powyższych wzorów. W jednej iteracji do nauki wykorzystywane są wszystkie obrazy zbioru uczącego. Dobrze jest gdy wektory danych podawane są na wejście w kolejności losowej. Po wykonanej iteracji można obliczyć błąd całościowy ze wzoru (3.5).
Istnieją różne kryteria określające moment zakończenia nauki sieci, np.:

- Nauka jest przerywana, gdy wartość funkcji kosztu dla zbioru uczącego osiągnie wartość mniejszą od wcześniej zadanego parametru: $|E| < \epsilon$
- Nauka jest przerywana gdy zmiana całościowej funkcji błędu jest mniejsza od zadanego parametru: $|\Delta E| < \epsilon$
- Nauka jest przerywana gdy spada zdolność sieci do generalizacji. Można wydzielić ze zbioru testowego podzbiór walidacyjny i sprawdzać na nim błąd klasyfikacji podczas nauki. Trening przerywa się w momencie gdy wraz ze spadkiem funkcji kosztu błąd zbioru walidacyjnego zaczyna rosnąć. Strategia ta ma zapobiec zbytniemu przeuczeniu sieci.
- Naukę przerywa się po określonej liczbie epok.

Wadami metody wstecznej propagacji błędu są między innymi wolna zbieżność oraz możliwość utykania na minimach lokalnych.

Rozdział 4

DMLP i Nieeuklidesowe sieci neuronowe

Poza aktywacją w formie iloczynu skalarnego wektora wag i wejścia omówioną w poprzednim rozdziale, w sieciach wielowarstwowych można stosować aktywację opartą o miarę odległości. Aktywacja taka używana jest między innymi w sieciach z radialnymi funkcjami bazowymi (sieci RBF wykorzystujące jako aktywację odległość od centrum funkcji bazowej). Sieci MLP, używające jako pobudzenia funkcji opartej o odległość, nazwać można D-MLP(Distance-based MLP)[8]. Sieci D-MLP w których wykorzystana została miara odległości inna niż euklidesowa noszą miano nieeuklidesowych sieci MLP. Zastąpienie ważonej aktywacji funkcją bazującą na odległości, powoduje znaczną zmianę kształtów obszarów decyzyjnych wyznaczanych przez sieć.

4.1 Formy aktywacji neuronu

Realizacja skomplikowanych obszarów decyzyjnych może być osiągnięta poprzez rozbudowywanie architektury sieci: dodanie kolejnej warstwy ukrytej i zwiększanie ilości neuronów. Wiąże się to jednak ze zwiększeniem złożoności sieci. Alternatywnym podejściem może być stosowanie odpowiednich funkcji transferu, pozwalających uzyskać obszary decyzyjne podobne do realizowanych przez standardowe sieci MLP, przy jednocześnie dość prostej budowie samej sieci. Poniżej krótko przedstawiono różne rodzaje funkcji wyjścia i aktywacji oraz omówiono ich wpływ na realizowane przez neuron granice decyzji. W podrozdziale tym korzystano głównie z prac: [10],[11].

4.1.1 Rodzaje aktywacji

Można wyróżnić trzy podstawowe rodzaje aktywacji:

- Standardowa aktywacja będąca iloczynem skalarnym określona równaniem: (3.1) lub

równoważnie wzorem: $I(\mathbf{x};\mathbf{w}) \propto \mathbf{w}^T\mathbf{x}$.

- Aktywacja oparta o funkcję odległości: $D(\mathbf{x};\mathbf{w}) \propto ||\mathbf{x}-\mathbf{w}||$, określająca stopień podobieństwa pomiędzy $\mathbf{x}$ a prototypowym wektorem $\mathbf{w}$. Jako $D(\mathbf{x};\mathbf{w})$ może być wzięta dowolna funkcja odległości np. Minkowskiego, Czebyszewa, Mahalanobiusa. Dla danych symbolicznych możliwe jest użycie normy probabilistycznej np. Modified Value Difference Metric (MVDM).

- Kombinacja dwóch typów aktywacji, mająca ogólną formę:
  $A(\mathbf{x};\mathbf{w},\mathbf{t}) \propto \alpha\mathbf{w}^T\mathbf{x} + \beta||\mathbf{x}-\mathbf{t}||$.

4.1.2 Funkcje wyjścia

Funkcje wyjścia można podzielić na dwie podstawowe grupy: skokowe (np. (3.4)) i ciągłe. Wśród funkcji ciągłych wyróżnia się następujące trzy rodziny:

- Funkcje sigmoidalne.

- Funkcje zlokalizowane wokół pojedynczego centrum, np. funkcje RBF.

- Funkcje posiadające kilka punktów centralnych.

Właściwy dobór funkcji wyjścia i aktywacji prowadzi do utworzenia funkcji transferu pozwalającej na wydzielenie przez sieć odpowiedniego obszaru decyzyjnego. W tej części pracy przedstawiona zostanie kombinacja sigmoidalnych funkcji wyjścia z aktywacją wykorzystującą funkcję odległości. Pełny podział i opis funkcji transferu można znaleźć w [10].

4.2 DMLP

W niniejszym rozdziale szerzej omówione zostaną właściwości sieci z funkcją transferu powstałą z połączenia aktywacji opartej na odległości oraz sigmoidalnej nielokalnej funkcji wyjścia. Funkcję aktywacji neuronu z postaci (3.1) da się przekształcić tak, aby stała się funkcją zawierającą odległość [9] :

$$\mathbf{w}\cdot\mathbf{x} = 0.5\left(\|\mathbf{w}\|^2 + \|\mathbf{x}\|^2 - \|\mathbf{w}-\mathbf{x}\|^2\right) \tag{4.1}$$

Drugim możliwym przekształceniem jest:

$$\mathbf{w}\cdot\mathbf{x} = 0.25\left(\|\mathbf{w}+\mathbf{x}\|^2 - \|\mathbf{w}-\mathbf{x}\|^2\right) \tag{4.2}$$

Uogólniając, funkcję wyjścia liczyć można od aktywacji w postaci:

$$net' = d_0 - D(\mathbf{w}, \mathbf{x}) \tag{4.3}$$

Gdzie d_0 ma wartość ustaloną (np. dla $d_0 = \|\mathbf{w}\|^2 + \|\mathbf{x}\|^2$ jeśli wektory $\mathbf{x}$ i $\mathbf{w}$ mają określoną stałą normę), bądź jest parametrem adaptacyjnym (zawierającym bias), a $D(\mathbf{w}, \mathbf{x})$ to funkcja odległości. Jako funkcję odległości najprościej jest przyjąć dowolną normę Minkowskiego:

$$D_M(\mathbf{w}, \mathbf{x}, \alpha, \gamma) = \left(\sum_{i=1}^{N} |w_i - x_i|^\alpha \right)^{\frac{1}{\gamma}} \tag{4.4}$$

Dla $\alpha = \gamma = 2$ dostaje się normę euklidesową. Zmiana parametrów α i γ pozwala uzyskać inne rodzaje normy Minkowskiego. Poza metryką Minkowskiego możliwe jest wykorzystanie innych funkcji odległości.
Implementację nieeuklidesowej sieci MLP można wykonać na dwa sposoby. Prezentowana w niniejszej pracy metoda polega na zastąpieniu aktywacji *net* poprzez net' i wymusza pewną modyfikację w algorytmie nauki:

Wzór na korekcję k-tej wagi i-tego neuronu (3.10) przyjmuje postać:

$$w_{ik}^{(j+1)} = w_{ik}^{(j)} - \eta \left(f_i - d_i \right) f'(net_i') \frac{\partial net_i'}{\partial w_{ik}} \tag{4.5}$$

Algorytm propagacji wstecznej błędu zakłada istnienie pochodnej funkcji aktywacji, a co za tym idzie w D-MLP musi istnieć pochodna funkcji odległości. Jest to spełnione dla miary Minkowskiego. Bez problemu da się dla niej wyznaczyć gradient:

$$\frac{\partial D_M(\mathbf{w}, \mathbf{x}, \alpha, \gamma)}{\partial w_k} = \frac{1}{\gamma} \left(\sum_{i=1}^{N} |w_i - x_i|^\alpha \right)^{\frac{1}{\gamma} - 1} \alpha |w_k - x_k|^{\alpha - 1} \frac{\partial |w_k - x_k|}{\partial w_k} \tag{4.6}$$

Możliwe jest też zaimplementowanie nieeuklidesowej sieci MLP bez zmian w algorytmie nauki, a tylko poprzez odpowiednią transformację danych wejściowych. Metodę tą opisano w artykule [9]. Zostanie ona pobieżnie omówiona w następnej sekcji.

4.3 Implementacja nieeuklidesowych sieci neuronowych przez renormalizację

Zakłada się stałą normę wektorów wejściowych $\mathbf{x}$. Aby ustalenie stałej normy nie łączyło się ze stratą informacji wymagane jest dodanie jednej lub więcej cech do wektorów wejściowych. Dodatkową składową wektora danych może być wartość $x_R = \sqrt{R^2 - \|\mathbf{x}\|^2}$ gdzie

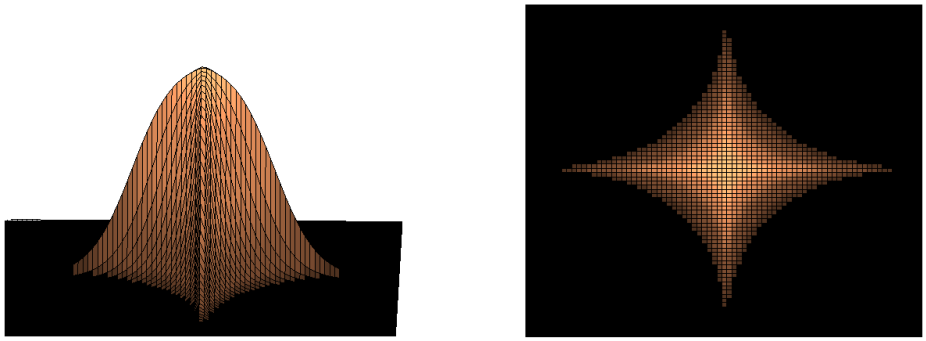
$R \geqslant max_{\mathbf{x}}\|\mathbf{x}\|$. Powyższe działania skutkują umieszczeniem danych na sferze o promieniu R. Wektor w postaci: $(\mathbf{x}, x_R)$ może zostać znormalizowany do $\|(\mathbf{x}, x_R)\|_D = 1$ przy użyciu norm opartych o różne funkcje odległości, np. odpowiedniej normy Minkowskiego. Bardziej szczegółowe opracowanie, wraz z przykładem zastosowania w [9].

4.4 Obszary decyzyjne generowane przez sieci D-MLP

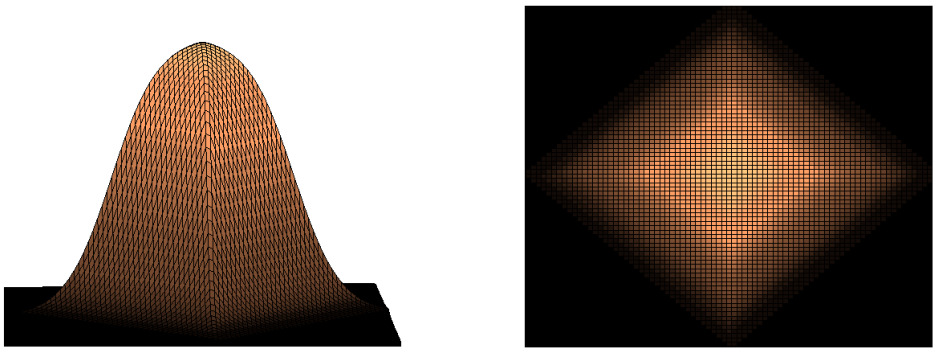
Zastąpienie aktywacji w funkcji wyjścia z postaci (3.1) na (4.3), powoduje zmianę kształtu realizowanych przez taką funkcję obszarów decyzyjnych. Zamknięty obszar decyzyjny do którego wyznaczenia w sieci MLP musiałyby być wykorzystane co najmniej trzy neurony warstwy ukrytej i jeden neuron wyjścia, w sieciach D-MLP może być zrealizowany za pomocą zaledwie jednego neuronu.

4.4.1 Kształty generowane przez normę Minkowskiego

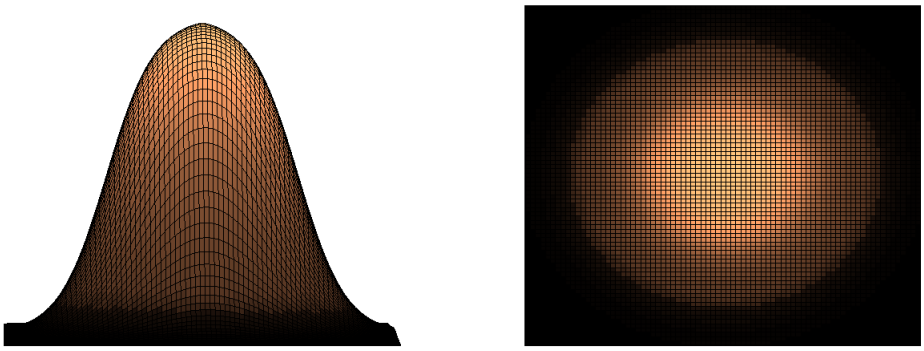
Na rysunkach: 4.1, 4.2, 4.3, 4.4 przedstawiono wykresy funkcji sigmoidalnej liczonej od aktywacji opartej o normę Minkowskiego dla różnych parametrów γ i α, z warunkiem $\alpha = \gamma$ i dla ustalonego d_0. W zależności od wartości α powierzchnie decyzyjne mogą przyjmować różne kształty: od hypocykloidy dla $\alpha \in (0, 1)$, która przy przejściu do $\alpha = 1$ daje równoległobok, poprzez okrąg dla $\alpha = 2$, przekształcający się w prostokąt dla coraz większych wartości α. Bardziej skomplikowane obrazy decyzyjne mogą zostać uzyskane, poprzez wykorzystanie odpowiednio większej liczby neuronów.



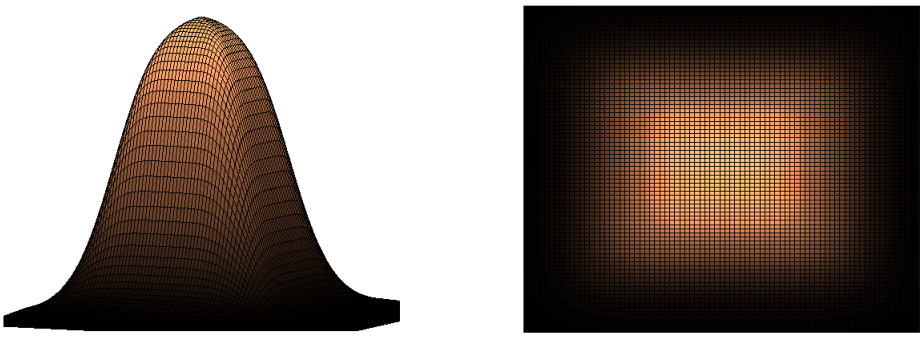
Rysunek 4.1: Obszar decyzyjny realizowany przez jeden neuron dla $\gamma = \alpha = 0.5$.



Rysunek 4.2: Obszar decyzyjny realizowany przez jeden neuron dla $\gamma = \alpha = 1$.



Rysunek 4.3: Obszar decyzyjny realizowany przez jeden neuron dla $\gamma = \alpha = 2$.



Rysunek 4.4: Obszar decyzyjny realizowany przez jeden neuron dla $\gamma = \alpha = 5$.

4.4.2 Obszary decyzyjne generowane przez D-MLP

W tym podrozdziale zaprezentowany zostanie przegląd kształtów jakie generować może pojedynczy neuron ze standardową sigmoidalną funkcją wyjścia o wzorze (5.5) i aktywacją w formie:

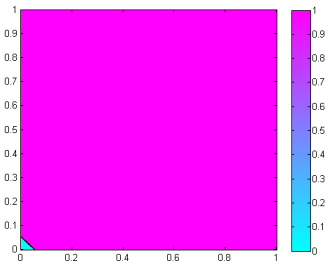
$$K_1 = 0.5 \left(\left(\sum_{i=1}^{N} |x_i|^{\alpha} \right)^{\frac{1}{\gamma}} + \left(\sum_{i=1}^{N} |w_i|^{\alpha} \right)^{\frac{1}{\gamma}} - \left(\sum_{i=1}^{N} |w_i - x_i|^{\alpha} \right)^{\frac{1}{\gamma}} - \theta \right) \tag{4.7}$$

Gdzie za θ w większości przypadków brana była średnia wartość $K_1(\theta = 0)$, a zakres danych wejściowych neuronu ograniczony został do obszaru $[0, 1]$ na $[0, 1]$.

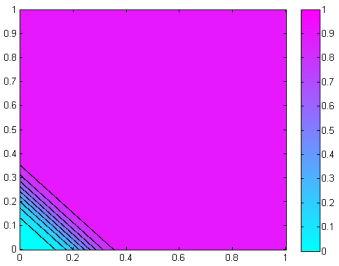
Dla aktywacji w formie iloczynu skalarnego wektorów wag i wejścia lub równoważnej mu formuły (4.1) dostaje się powierzchnię decyzyjną o równoległych poziomicach, zagęszczających się wraz ze wzrostem wartości wag. Dla K_1, dla parametrów: α różnego od 2 i γ różnego od 1, wpływ wag nie jest już tak oczywisty. Najciekawsze obszary dostaje się dla małych wag, których wartość nie przekracza 1, a nawet jest mniejsza od 0.4. Dla wag przyjmujących wartości 1 i większe, zależność kształtu γ jest analogiczna dla różnych α. Dla małych γ dostaje się podobne do opisywanych wcześniej równoległe poziomice zagęszczone tym bardziej, im większe wagi i mniejsze γ. Wraz z rosnącym γ skos się zmniejsza i poziomice zaczynają się zakrzywiać w zależności od α: wklęśle dla $\alpha < 1$, romboidalnie dla $\alpha = 1$, koliście dla $\alpha = 2$ i coraz bardziej "prostokątnie" dla $\alpha > 2$. Przykładowy przebieg zmian w kształcie funkcji wraz ze wzrostem parametru γ, od $\gamma = 0.1$ do $\gamma = 10$, dla $\alpha = 2$ i $w_1 = w_2 = 1$ przedstawiono na rysunku Rys. 4.5. Ewolucja zarysu poziomic dla innych wartości α i większych wag wygląda podobnie.

Sytuacja dla małych wag jest nieco bardziej skomplikowana, również można tu zauważyć pewne prawidłowości, chociażby takie, że dla γ zbliżającego się lub przekraczającego wartość α, zaczynają się tworzyć zamknięte obszary o kształtach zależnych od α. Przykładowe zarysy poziomic dla aktywacji (4.7) dla różnych wartości parametrów, pogrupowane według wartości α przedstawiono na rycinach: Rys. 4.6 i Rys. 4.7 dla $\alpha = 0.5$, Rys. 4.8 dla $\alpha = 1$, Rys. 4.9 dla $\alpha = 2$, Rys. 4.10 dla $\alpha = 4$. Typowe kształty funkcji zostały zebrane i uporządkowane według najważniejszych parametrów w tabeli 4.1.

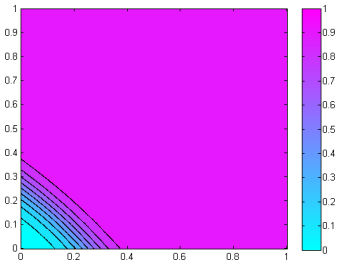
Powyższe obserwacje można wykorzystać, na przykład w inicjalizacji sieci z aktywacją (4.7). Wartości początkowe wag powinny być niewielkie, mniejsze od 0.5, choćby dlatego, że dla niewielkich wag niezerowy gradient zazwyczaj rozciąga się na dość dużej części obszaru, co jest ważne zwłaszcza w początkowej fazie uczenia sieci.



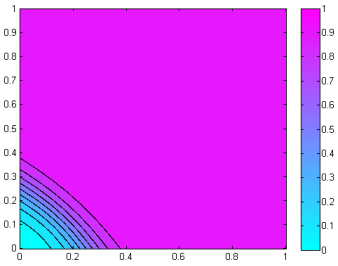
(a) $\gamma = 0.1$



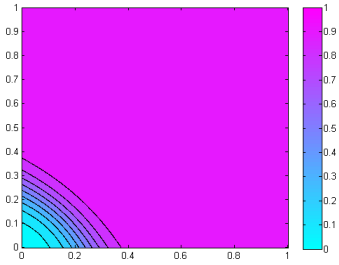
(b) $\gamma = 1$



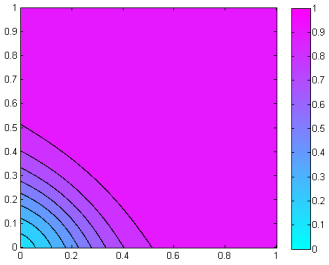
(c) $\gamma = 1.25$



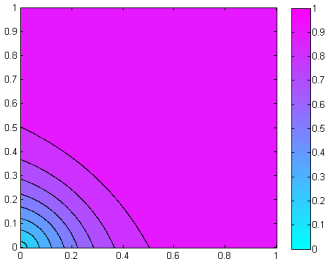
(d) $\gamma = 1.5$



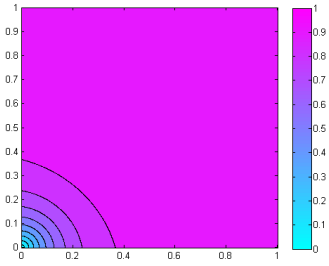
(e) $\gamma = 1.75$



(f) $\gamma = 2$

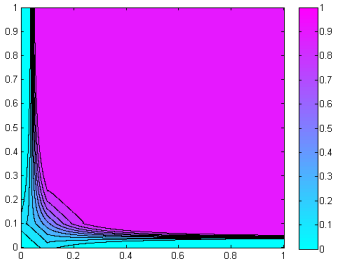


(g) $\gamma = 3$

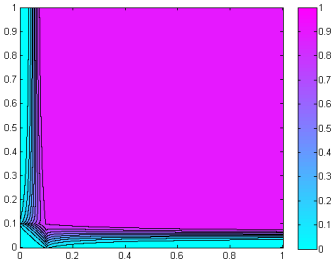


(h) $\gamma = 10$

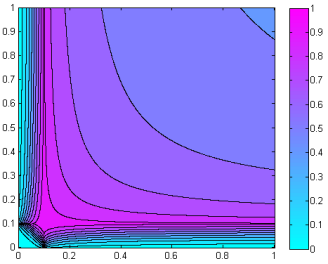
Rysunek 4.5: Ewolucja kształtu obszaru decyzyjnego realizowanego przez neuron z funkcją aktywacji (4.7), wraz ze zmianą parametru γ, dla wag: $w_1 = w_2 = 1$ i parametru $\alpha = 2$.



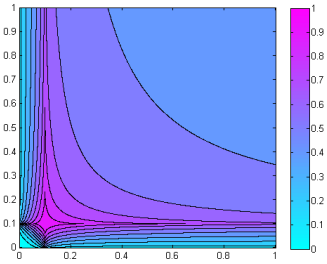
(a) $\gamma = 0.25$



(b) $\gamma = 0.5$

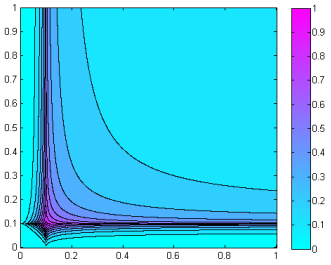


(c) $\gamma = 1$

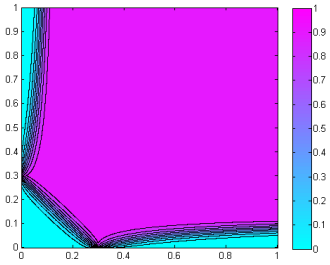


(d) $\gamma = 2$

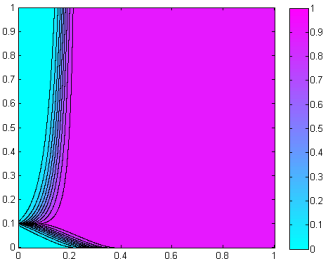
Rysunek 4.6: Kształty obszaru decyzyjnego realizowanego przez neuron z funkcją aktywacji (4.7), dla $w_x = w_y = 0.1$, $\alpha = 0.5$ i różnych γ.



(a) $\gamma = 1$, $w_1 = w_2 = 0.1$, $\theta = \theta_0 + 0.6$

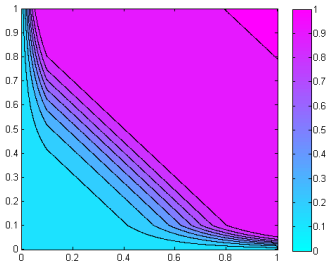


(b) $\gamma = 0.5$, $w_1 = w_2 = 0.3$

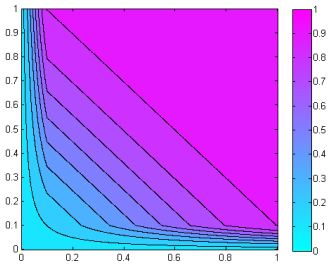


(c) $\gamma = 0.5$, $w_1 = 0.5$, $w_2 = 0.1$

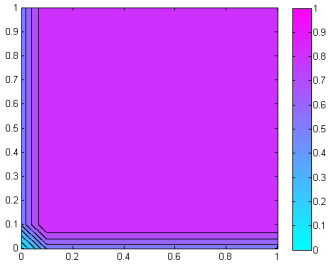
Rysunek 4.7: Kształty obszaru decyzyjnego realizowanego przez neuron z funkcją aktywacji (4.7), dla $\alpha = 0.5$, dla różnych wag i γ.



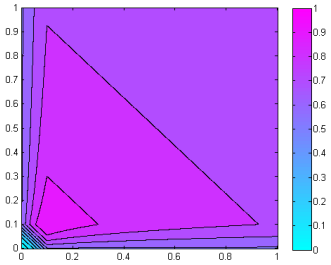
(a) $\gamma = 0.25$



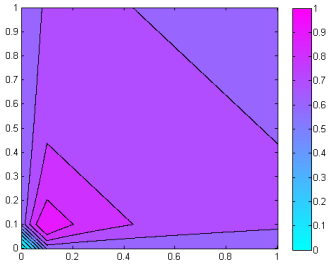
(b) $\gamma = 0.5$



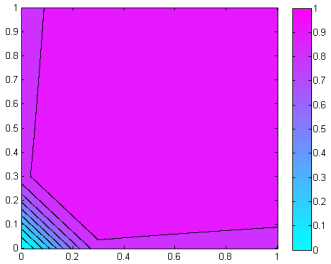
(c) $\gamma = 1$



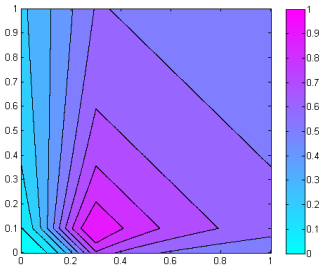
(d) $\gamma = 2$



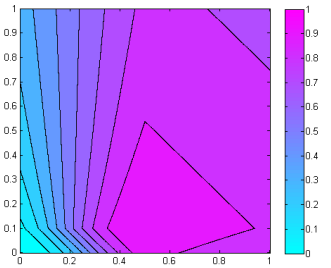
(e) $\gamma = 4$



(f) $\gamma = 4, w_1 = w_2 = 0.3$

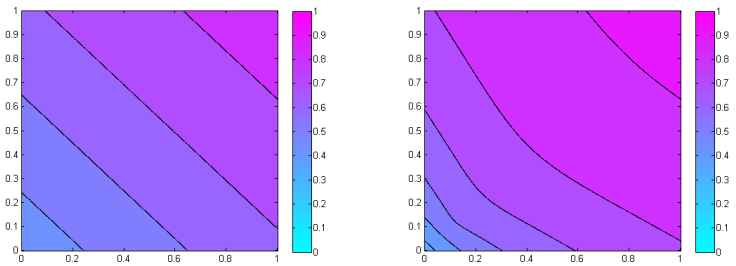


(g) $\gamma = 4, w_1 = 0.3, w_2 = 0.1$

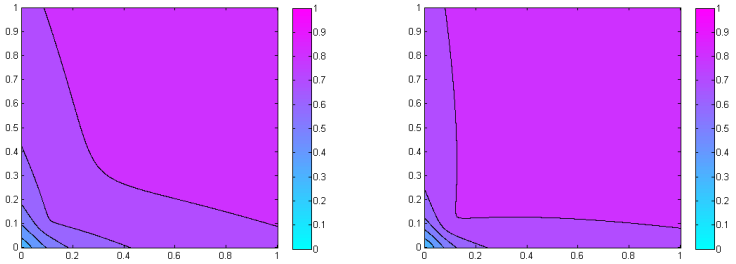


(h) $\gamma = 3, w_1 = 0.5, w_2 = 0.1$

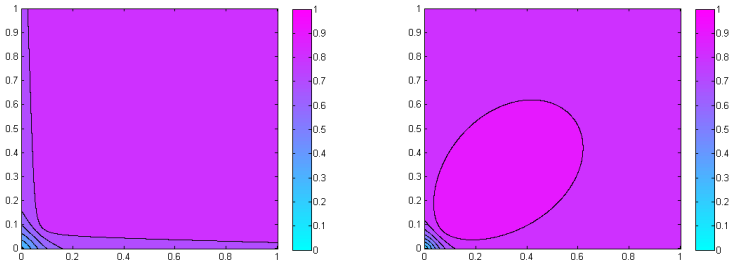
Rysunek 4.8: Kształty obszaru decyzyjnego realizowanego przez neuron z funkcją aktywacji (4.7), dla $\alpha = 1$. Jeśli nie podano inaczej to $w_1 = w_2 = 0.1$.



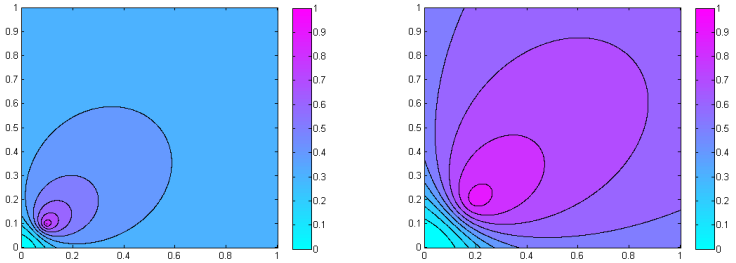
(a) $\gamma = 1$ (b) $\gamma = 1.25$



(c) $\gamma = 1.5$ (d) $\gamma = 1.75$

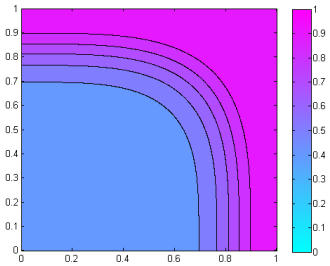


(e) $\gamma = 2$ (f) $\gamma = 3$

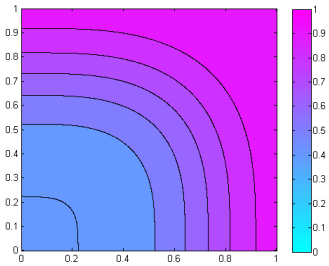


(g) $\gamma = 3$, $\theta = \theta_0 + 0.3$ (h) $\gamma = 3$, $\theta = \theta_0 + 0.3$, $w_1 = w_2 = 0.2$

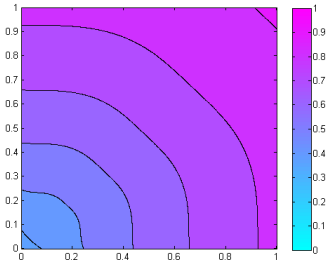
Rysunek 4.9: Ewolucja kształtu obszaru decyzyjnego realizowanego przez neuron z funkcją aktywacji (4.7), wraz ze zmianą parametru γ, dla wag: $w_1 = w_2 = 0.1$ i parametru $\alpha = 2$ (rys a-f)



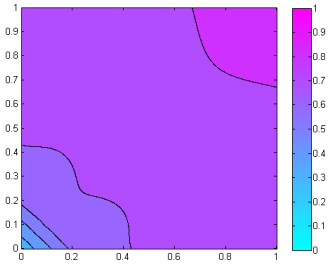
(a) $\gamma = 0.5$



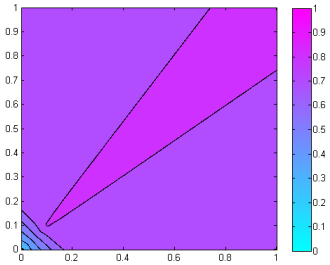
(b) $\gamma = 1$



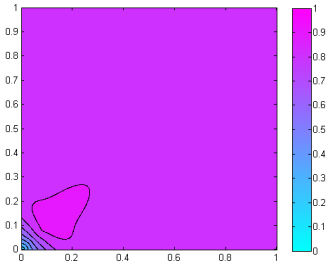
(c) $\gamma = 2$



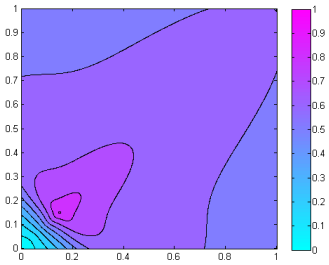
(d) $\gamma = 3$



(e) $\gamma = 4$



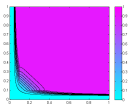
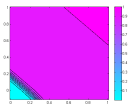
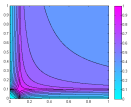
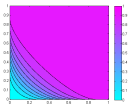
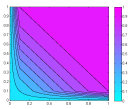
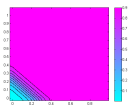
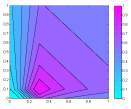
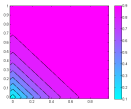
(f) $\gamma = 6$

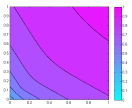
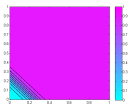
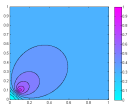
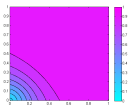
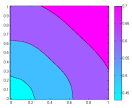
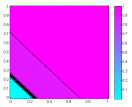
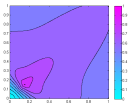
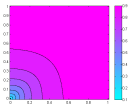


(g) $\gamma = 6$,$\theta = \theta_0 + 0.3$,

Rysunek 4.10: Kształty obszaru decyzyjnego realizowanego przez neuron z funkcją aktywacji (4.7), dla $\alpha = 4$ i $w_1 = w_2 = 0.1$.

Tabela 4.1: Porównanie typowych kształtów generowanych przez równanie (4.7)

α	γ	$[w_1, w_2]$	Typowy kształt
$[0, 1)$	$[0, \alpha)$	$[0-1, 0-1]$	
		$[1-\infty, 1-\infty]$	
	$[\alpha, \infty)$	$[0-1, 0-1]$	
		$[1-\infty, 1-\infty]$	
1	$[0, 1)$	$[0-1, 0-1]$	
		$[1-\infty, 1-\infty]$	
	$[1, \infty)$	$[0-1, 0-1]$	
		$[1-\infty, 1-\infty]$	

α	γ	$[w_1, w_2]$	Typowy kształt
2	$[0, 2)$	$[0-1, 0-1]$	
		$[1-\infty, 1-\infty]$	
	$[2, \infty)$	$[0-1, 0-1]$	
		$[1-\infty, 1-\infty]$	
$(2, \infty)$	$[0, \alpha)$	$[0-1, 0-1]$	
		$[1-\infty, 1-\infty]$	
	$[\alpha, \infty)$	$[0-1, 0-1]$	
		$[1-\infty, 1-\infty]$	

Rozdział 5

Implementacja sieci

W celu sprawdzenia działania i porównania skuteczności zaimplementowane zostały dwa rodzaje sieci: MLP i D-MLP. Do nauki wykorzystano algorytm propagacji wstecznej błędu. Sieci zostały przetestowane na sztucznych rozkładach danych. Program został napisany w języku $C\sharp$. Poniżej omówiono niektóre kwestie dotyczące programu.

5.1 Wstępne przetwarzanie danych

Sieci neuronowe działają zazwyczaj na wartościach numerycznych. Jeśli atrybuty danych wejściowych nie są wartościami numerycznymi (np. kolor, płeć), powinny zostać zastąpione przez wektory binarne bądź liczby. Na przykład cecha przyjmująca wartości: dziecko, kobieta, mężczyzna, może zostać zrealizowana w następujący sposób: dziecko (1 0 0), kobieta (0 1 0), mężczyzna (0 0 1). Także klasy na jakie podzielone są dane oznacza się za pomocą numerycznych etykiet. Taką etykietę realizuje się zazwyczaj przez wektor binarny o wymiarze równym ilości neuronów wyjściowych. Ewentualnie w przypadku 2 klas można zastosować jeden neuron wyjściowy i oznaczenie dla klas 0 i 1.

Często spotyka się dane, których cechy mają wartości mieszczące się w różnych, nierzadko odległych przedziałach. Zbyt duże dysproporcje pomiędzy wartościami cech w poszczególnych wymiarach są niebezpieczne, zwłaszcza dla sieci określających prawdopodobieństwo przynależności do danej klasy na podstawie odległości. Biorąc pod uwagę powyższe, przed "podaniem" danych wejściowych do sieci neuronowej zazwyczaj poddaje się je pewnym transformacjom. Poniżej zostaną omówione dwie z takich transformacji: normalizacja i standaryzacja[4].

Normalizacja Normalizacja to prosta transformacja sprawiająca, że wartości cech mieszczą się w przedziale $[0, 1]$. W celu dokonania normalizacji stosuje się przekształcenie:

$$\acute{x}_i^s = \frac{x_i^s - x_{min}^s}{x_{max}^s - x_{min}^s} \tag{5.1}$$

Gdzie x_i^s to s-ta cecha i-tego wektora danych, $x_{min/max}^s$ to najmniejsza/największa wartość s-tej cechy spośród wszystkich wektorów danych.

Czasami stosuje się normalizację z usunięciem elementów odstających (tzw. normalizację z obcięciem), polegającą na odrzuceniu pewnej części skrajnych wartości atrybutów przy liczeniu maksimum i minimum danej cechy. Ma to duże znaczenie, gdy zbiór danych zawiera obiekty, których atrybuty mają wartości znacznie wykraczające poza normalny zakres danej cechy.

Standaryzacja Standaryzacja w odróżnieniu od normalizacji zachowuje oryginalny rozkład wartości cech. Jest bezpieczniejsza w stosowaniu od zwykłej normalizacji. Po dokonaniu standaryzacji średnia wartość atrybutu wynosi 0, z odchyleniem standardowym równym 1. Standaryzacji dokonuje się używając przekształcenia:

$$\acute{x}_i^s = \frac{x_i^s - \overline{x^s}}{\sigma_x^s} \tag{5.2}$$

Gdzie $\overline{x}$ to wartść średnia:

$$\overline{x^s} = \frac{1}{N}\sum_{i=1}^{N} x_i^s \tag{5.3}$$

σ_x^2 jest wariancją :

$$\sigma_x^{s\,2} = \frac{1}{N-1}\sum_{i=1}^{N} (x_i^s - \overline{x^s})^2 \tag{5.4}$$

5.2 Zastosowane funkcje wyjścia

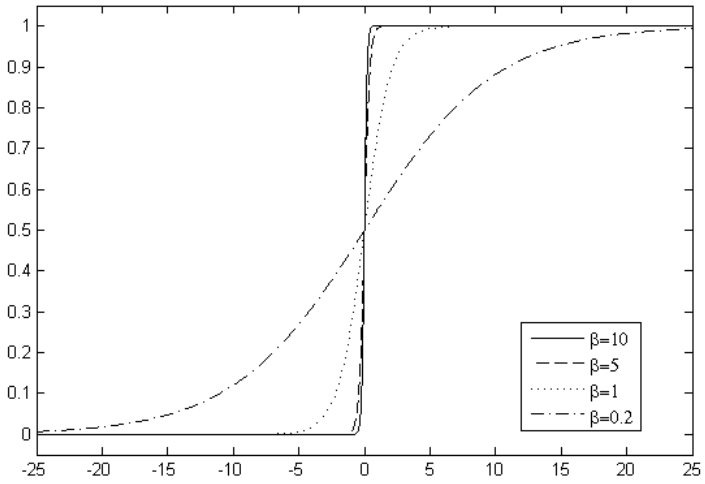
W pracy wykorzystano trzy funkcje wyjścia.

- Sigmoidalna funkcja unipolarna: Szeroko stosowana i podawana w wielu opracowaniach jako typowa funkcja transferu dla wielowarstwowych sieci neuronowych. Posiada prostą pochodną w postaci : $f_1^{'}(net) = \beta f_1(1 - f_1)$. Parametr β określa nachylenie funkcji.

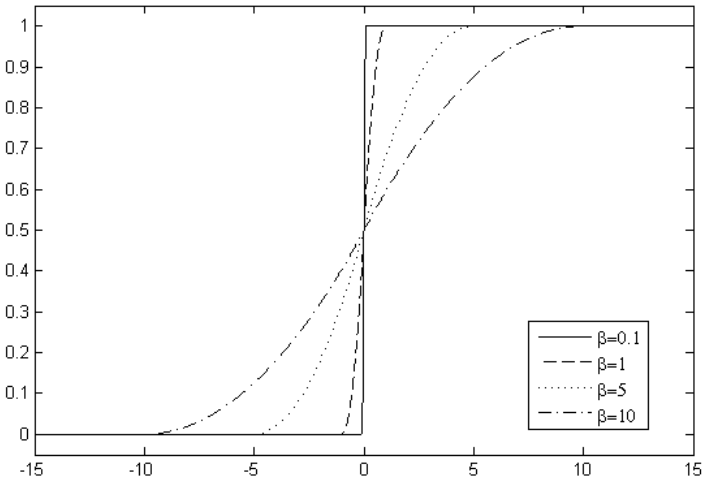
$$f_1(net) = \frac{1}{1 + e^{-\beta \cdot net}} \tag{5.5}$$

- Funkcja f_2 w postaci:

$$f_2(x - a, \Delta x) = \begin{cases} 0, & x < a - \Delta x \\ \frac{1}{2} + \frac{(x-a)(2\Delta x + x - a)}{2(\Delta x)^2}, & x \in [a - \Delta x, a) \\ \frac{1}{2} + \frac{(x-a)(2\Delta x - x + a)}{2(\Delta x)^2}, & x \in [a, a + \Delta x] \\ 1, & x > a + \Delta x \end{cases} \tag{5.6}$$



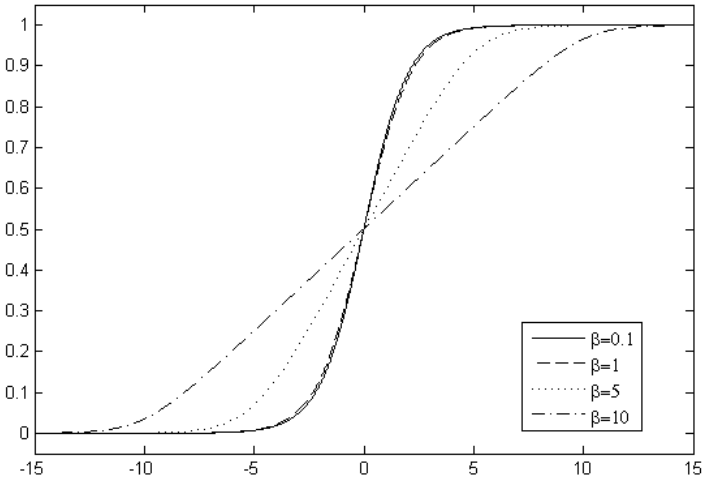
Rysunek 5.1: Wykres funkcji logistycznej f_1.



Rysunek 5.2: Wykres funkcji f_2.

- Funkcja f_3 o wzorze:

$$f_3(x-a,b) = \frac{1}{2\beta} ln \left[\frac{1+e^{x-a+\beta}}{1+e^{x-a-\beta}} \right] \tag{5.7}$$



Rysunek 5.3: Wykres funkcji f_3.

Funkcje f_2 i f_3 pochodzą z [6].

Skos w wykorzystywanych w pracy funkcjach transferu, jest regulowany za pomocą odpowiednich parametrów (β,Δx), parametry te nie są adaptacyjne. Zbyt mocny skos utrudnia naukę, z kolei funkcja o małej stromiźnie często nie jest w stanie prawidłowo rozdzielić obszarów decyzyjnych. W zwykłych sieciach MLP rzeczywisty skos funkcji zależny jest dodatkowo od wartości wag, zmienia się więc podczas nauki. W sieciach D-MLP ze względu na specyficzną formę aktywacji skos nie zależy już w sposób tak oczywisty od wag. Rodzi to pewien problem w przypadku funkcji f_3. Nachylenie wywołane przez przypisanie β wartości 0.1 w wielu przypadkach jest niewystarczające, a ustawienie β poniżej 0.1 nie wpływa już na zwiększenie nachylenia funkcji. O ile w MLP korygowało się to podczas nauki poprzez zmianę wag, to sieci D-MLP miały problem z nauką i określeniem odpowiednich obszarów decyzyjnych, ze względu na zbyt mały skos funkcji. W związku z tym dla f_3 w D-MLP wprowadzono jeszcze jeden parametr nieadaptacyjny : d, pozwalający odpowiednio regulować nachylenie. Po tej zmianie funkcja f_3 używana w D-MLP ma postać:

$$f_3(x-a,b) = \frac{1}{2\beta} ln \left[\frac{1+e^{d(x-a)+\beta}}{1+e^{d(x-a)-\beta}} \right] \tag{5.8}$$

5.3 Inicjalizacja

5.3.1 Inicjalizacja wag

Istnieją różne techniki optymalizacji inicjalizacji parametrów adaptacyjnych sieci. Dobrze zainicjalizowana sieć uczy się krócej i ma mniejsze szanse utknąć w minimum lokalnym funkcji kosztu. W pracy przyjęto najprostszy sposób inicjalizacji wag: sposób losowy. Inicjalizacja była powtarzana kilkunastokrotnie. Wylosowane wagi mieszczą się w przedziale $(0, 1)$. Dla danych znormalizowanych mieszczących się w pewnym zakresie wartości, najlepiej przyjąć wagi startowe w podobnym przedziale. Dodatkowo zbyt wysoka wartość wag na początku nauki w przypadku MLP powodowałaby za duże nachylenie funkcji transferu, utrudniając naukę. Dla sieci DMLP przedział możliwych wartości wag został ograniczony do $(0, 0.5)$.

5.4 Parametry α i γ

Parametry α i γ określają rodzaj metryki Minkowskiego, wpływając na kształty obszarów decyzyjnych tworzonych przez sieć D-MLP. Zostały one potraktowane jako parametr adaptacyjny. Podczas nauki zmieniają się w każdym neuronie, na zasadzie analogicznej do korekcji wag.

Rozdział 6

Wyniki

W rozdziale tym przedstawiono wyniki zastosowania sieci D-MLP do klasyfikacji danych. Testy wykonano na przykładowych rozkładach, efekty porównano z rezultatami dla zwykłych sieci MLP.

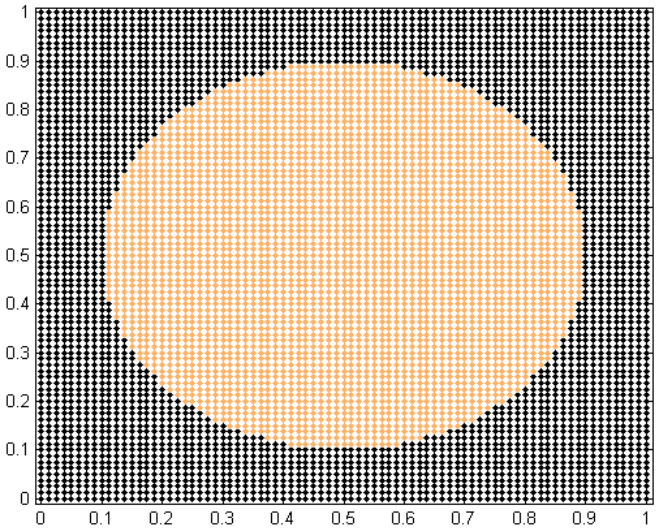
6.1 Dane sztuczne 2D

Wykorzystane w tym podrozdziale rozkłady pochodzą ze strony Tom'a Fawcett'a[13], gdzie zamieszczone zostały wraz z wynikami dla różnych algorytmów klasyfikacji. W pracy zachowano oryginalne nazwy rozkładów, takie same jak na stronie.

Każdy plik pierwotnie składał się z 6561 punktów należących do dwóch klas. Punkty były rozmieszczone na dwuwymiarowym obszarze [0,4]x[0,4] z rozdzielczością 0.05. Dane zostały znormalizowane, aby mieściły się w obszarze [0,1]x[0,1]. Testowanie sieci odbyło się za pomocą metody dziesięciokrotnej kroswalidacji. Dla każdego zbioru danych zamieszczone zostały: rycina przedstawiająca dany rozkład, tabela porównawcza dokładności klasyfikacji osiągniętej przez sieci MLP i D-MLP dla trzech funkcji f_1, f_2, f_3, przykładowe obszary decyzyjne wygenerowane przez sieci dla próbki 1000 losowo wybranych punktów i wykresy porównujące zbieżność sieci MLP i D-MLP. Liczba punktów na rysunkach przedstawiających obszary decyzyjne została ograniczona do 1000, ponieważ większa ich ilość zakłócałaby czytelność, a próbka 1000 obiektów wydaje się być wystarczająco reprezentatywna dla 6561 punktowych zbiorów danych. Żółtą barwą zaznaczono dokładny przebieg granicy decyzyjnej.

6.1.1 Circle

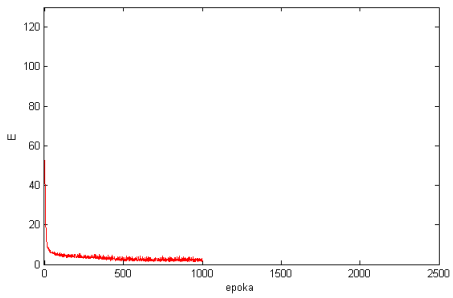
Dane Circle składają się z 5651 punktów podzielonych na dwie klasy: po 3364 i 3197 obiektów. Circle nie jest trudnym rozkładem. Dla obu rodzajów sieci, dla wszystkich funkcji wyjścia zbieżność funkcji błędu jest szybka. Sieć D-MLP daje bardzo dobre wyniki i wykorzystuje jeden neuron, MLP radzi sobie dość dobrze, przybliżając kształt koła za pomocą 6 neuronów.



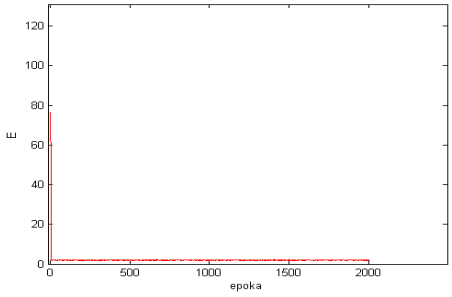
Rysunek 6.1: Rozkład Circle

Funkcja transferu	Rodzaj sieci	Liczba neuronów	Poprawność klasyfikacji
f_1	DMLP	1	99.94 ± 0.14
	MLP	6	98.43 ± 0.64
f_2	DMLP	1	99.83 ± 0.11
	MLP	6	97.35 ± 1.43
f_2	DMLP	1	99.76 ± 0.26
	MLP	6	98.96 ± 0.39

Tabela 6.1: Wyniki otrzymane dla rozkładu Circle dla sieci MLP i D-MLP, metodą dziesięciokrotnej kroswalidacji.

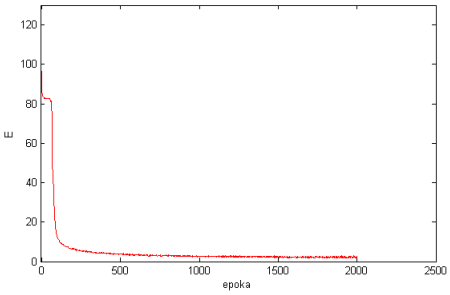


(a) MLP 6 neuronów

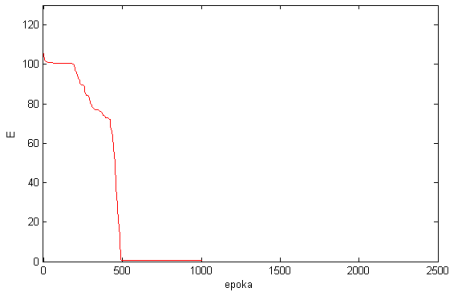


(b) D-MLP 1 neuron

Rysunek 6.2: Zmiana błędu średniokwadratowego w zależności od epoki uczenia dla danych Circle dla funkji: f_1

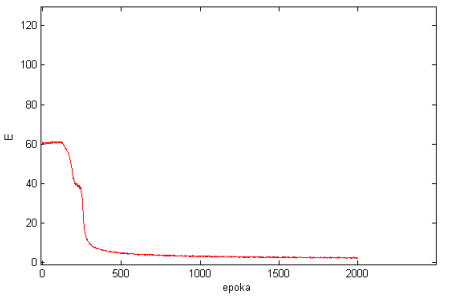


(a) MLP 6 neuronów

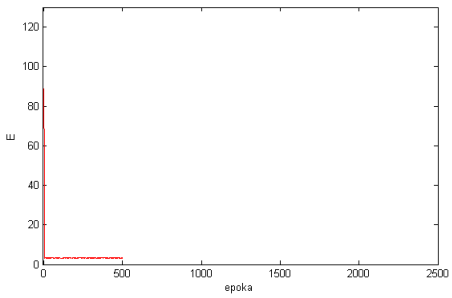


(b) D-MLP 1 neuron

Rysunek 6.3: Zmiana błędu średniokwadratowego w zależności od epoki uczenia dla danych Circle dla funkji: f_2

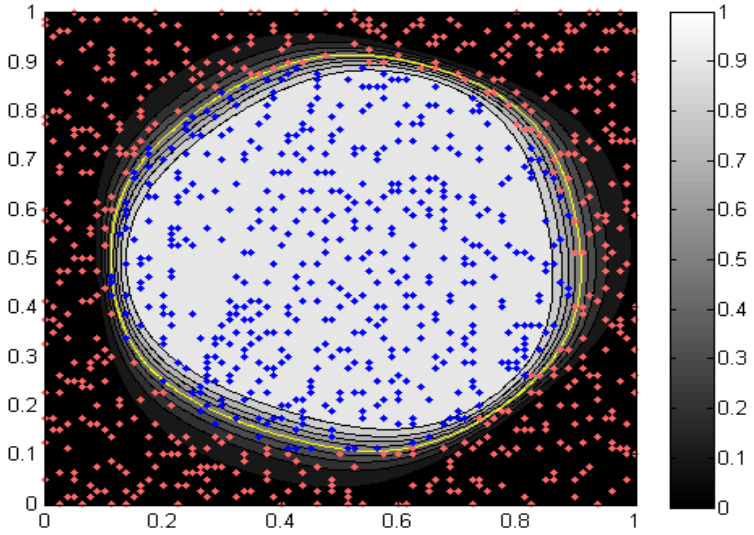


(a) MLP 6 neuronów

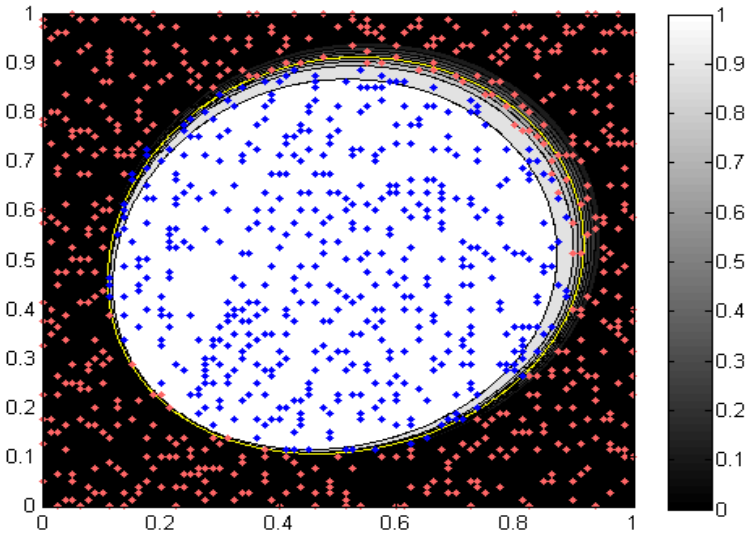


(b) D-MLP 1 neuron

Rysunek 6.4: Zmiana błędu średniokwadratowego w zależności od epoki uczenia dla danych Circle dla funkji:f_3

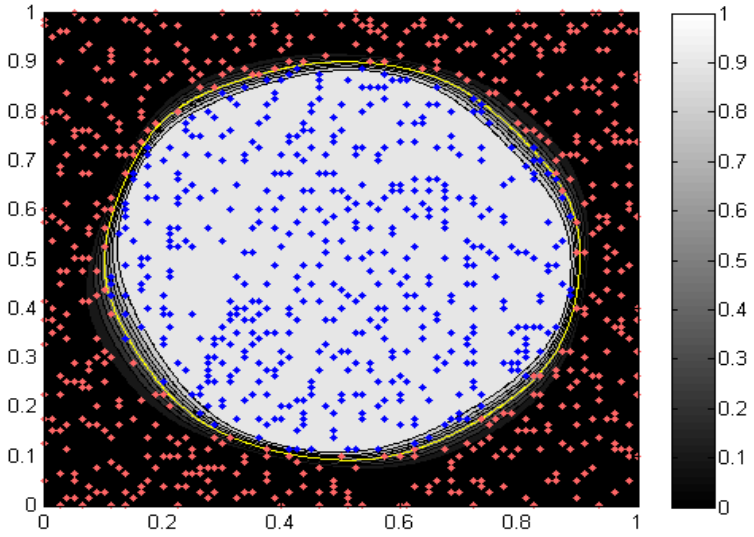


(a) MLP -6 neuronów.

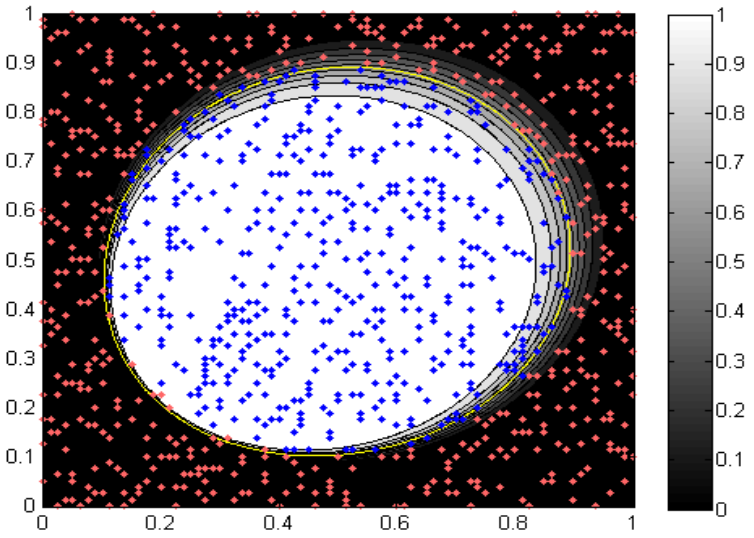


(b) D-MLP -1 neuron.

Rysunek 6.5: Obszary decyzyjne utworzone dla danych Circle, przez sieci MLP i D-MLP używające funkcji: f_1.

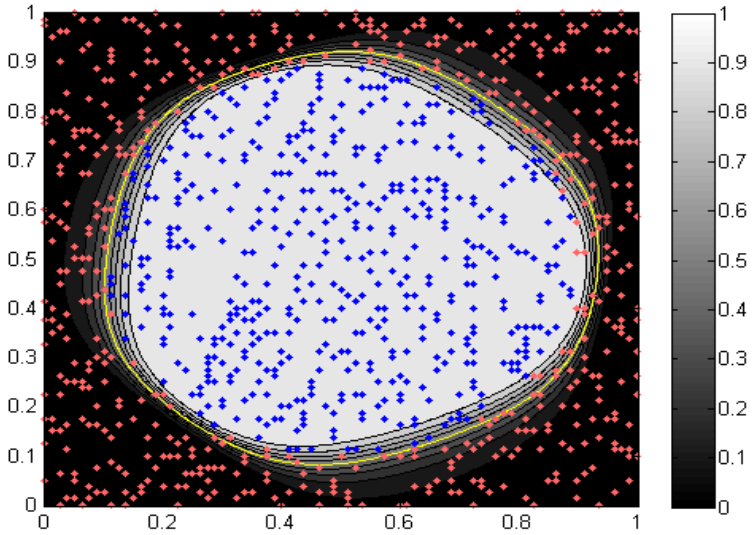


(a) MLP -6 neuronów.

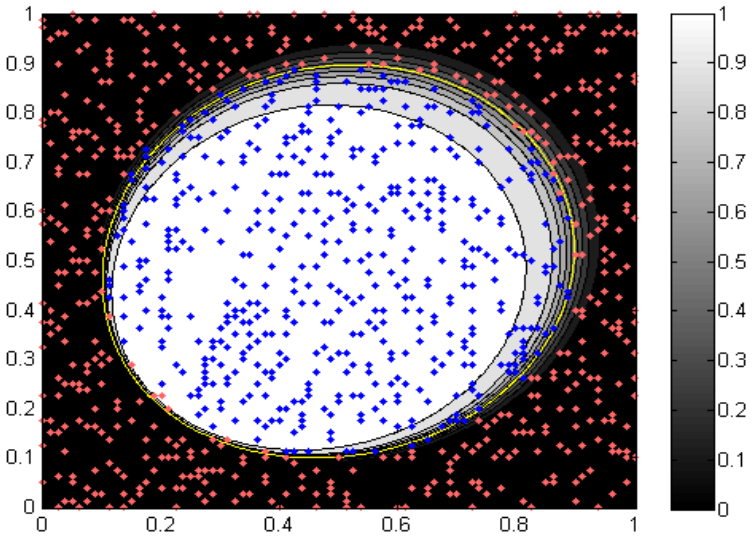


(b) D-MLP -1 neuron.

Rysunek 6.6: Obszary decyzyjne utworzone dla danych Circle, przez sieci MLP i D-MLP używające funkcji: f_2.



(a) MLP -6 neuronów.

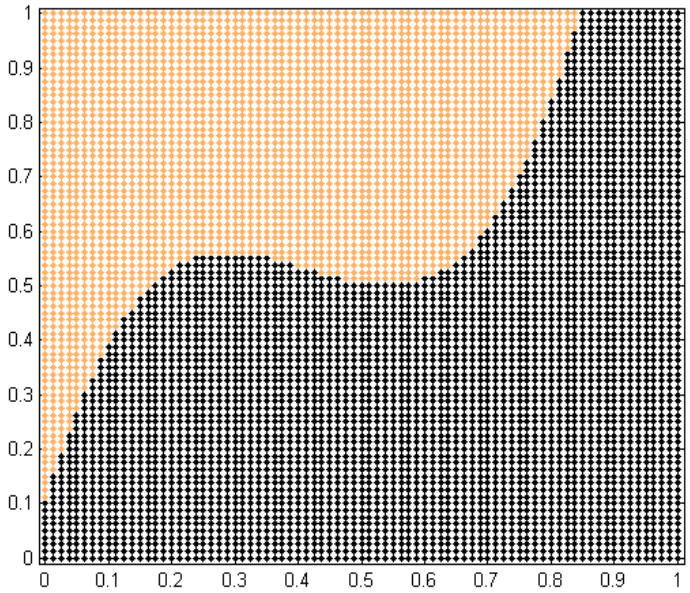


(b) D-MLP -1 neuron.

Rysunek 6.7: Obszary decyzyjne utworzone dla danych Circle, przez sieci MLP i D-MLP używające funkcji: f_3.

6.1.2 Polynomial

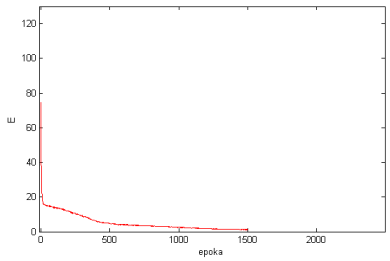
Dane Polynomial składają się z 6561 punktów podzielonych na dwie klasy: po 3989 i 2572 obiektów. Przetestowano na nich sieć MLP składającą się z trzech neuronów oraz dwie sieci D-MLP, jedną złożoną z dwóch, drugą z trzech neuronów. Dodanie trzeciego neuronu nieznacznie poprawia wyniki dla D-MLP. Efekty działania obu rodzajów sieci są porównywalne, zaznaczyć należy jedynie, że D-MLP potrzebuje większej liczby inicjalizacji do osiągnięcia prawidłowego rozwiązania, niż sieć MLP.



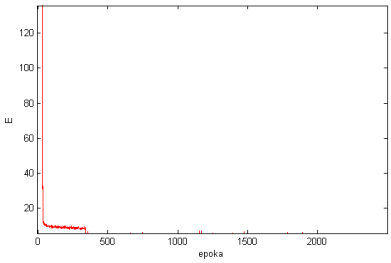
Rysunek 6.8: Rozkład Polynomial

Funkcja transferu	Rodzaj sieci	Liczba neuronów	Poprawność klasyfikacji
f_1	DMLP	2	97.10 ± 0.97
	DMLP	3	98.55 ± 0.89
	MLP	3	96.28 ± 3.24
f_2	DMLP	2	96.45 ± 1.01
	DMLP	3	97.87 ± 1.20
	MLP	3	98.92 ± 0.72
f_3	DMLP	2	98.25 ± 0.40
	DMLP	3	98.58 ± 0.64
	MLP	3	99.42 ± 0.27

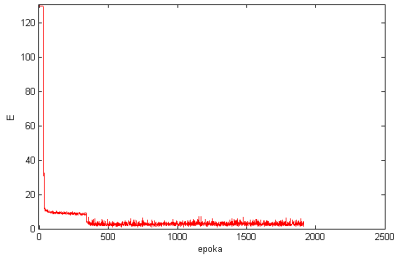
Tabela 6.2: Wyniki otrzymane dla rozkładu Polynomial dla sieci MLP i D-MLP metodą dziesięciokrotnej kroswalidacji.



(a) MLP 3 neurony

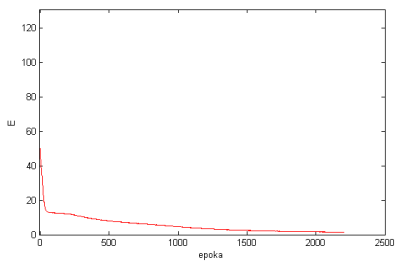


(b) D-MLP 2 neurony

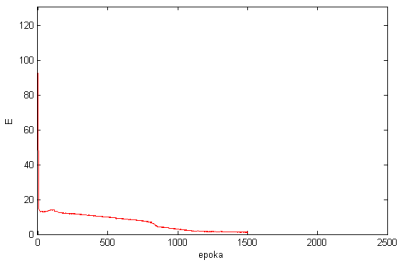


(c) D-MLP 3 neurony

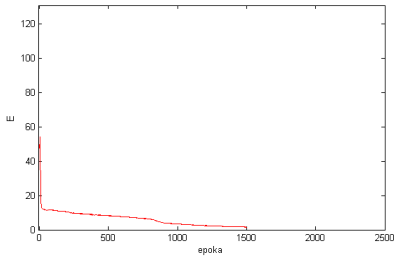
Rysunek 6.9: Zmiana błędu średniokwadratowego w zależności od iteracji uczenia dla danych Polynomial dla funkcji f_1.



(a) MLP 3 neurony

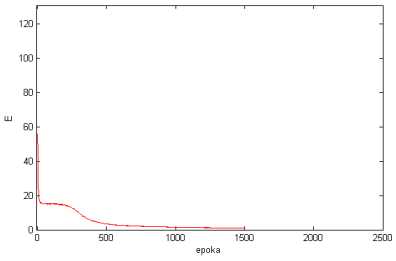


(b) D-MLP 2 neurony

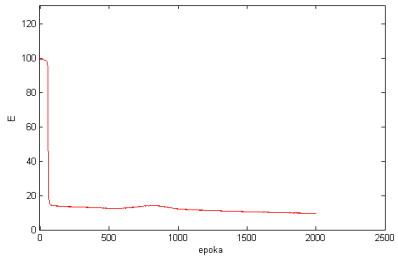


(c) D-MLP 3 neurony

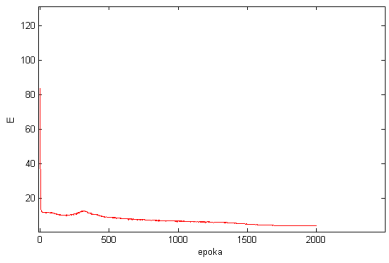
Rysunek 6.10: Zmiana błędu średniokwadratowego w zależności od epoki uczenia dla danych Polynomial dla funkcji f_2.



(a) MLP 3 neurony

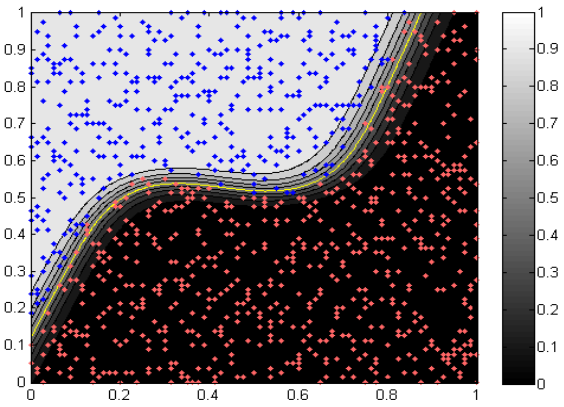


(b) D-MLP 2 neurony

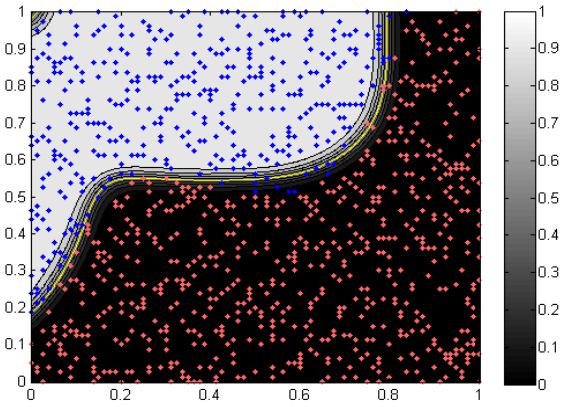


(c) D-MLP 3 neurony

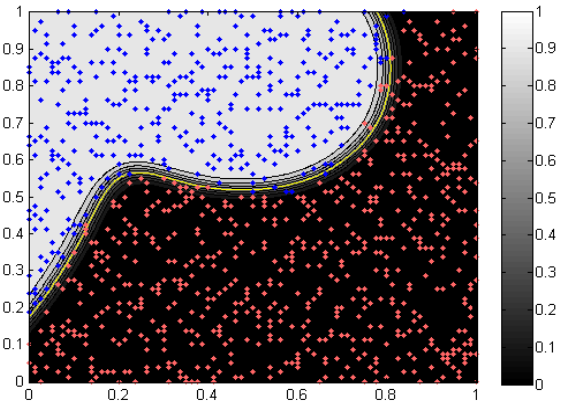
Rysunek 6.11: Zmiana błędu średniokwadratowego w zależności od epoki uczenia dla danych Polynomial dla funkcji f_3.



(a) MLP 3 neurony.

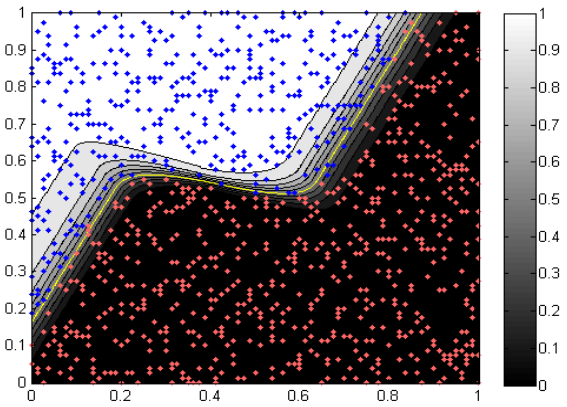


(b) D-MLP 2 neurony.

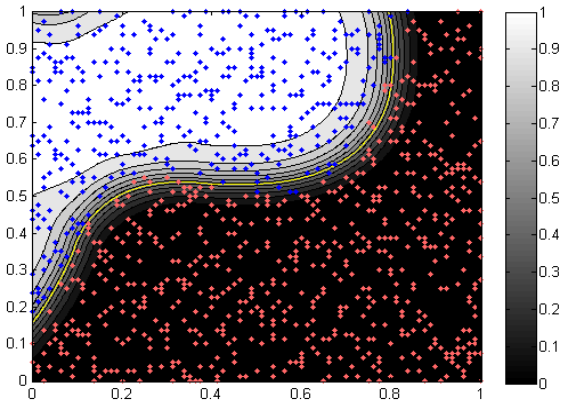


(c) D-MLP 3 neurony.

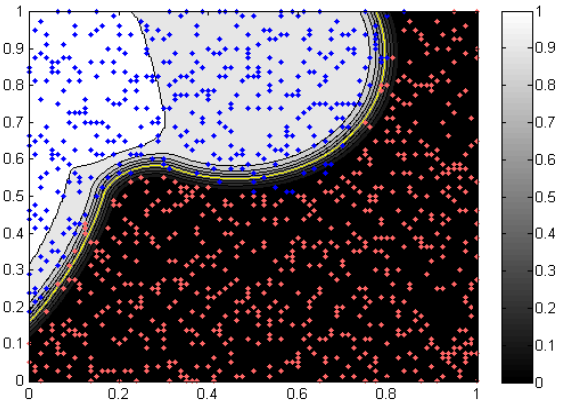
Rysunek 6.12: Obszary decyzyjne utworzone dla danych Polynomial, przez sieci MLP i D-MLP używające funkcji: f_1.



(a) MLP 3 neurony.

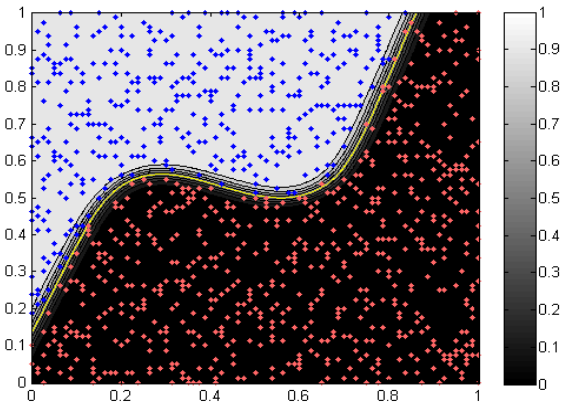


(b) D-MLP 2 neurony.

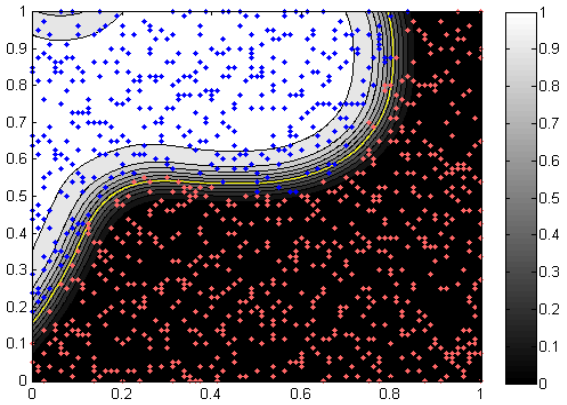


(c) D-MLP 3 neurony.

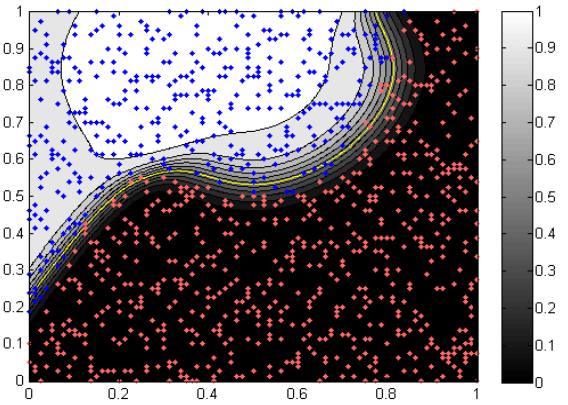
Rysunek 6.13: Obszary decyzyjne utworzone dla danych Polynomial, przez sieci MLP i D-MLP używające funkcji: f_2.



(a) MLP 3 neurony.



(b) D-MLP 2 neurony.

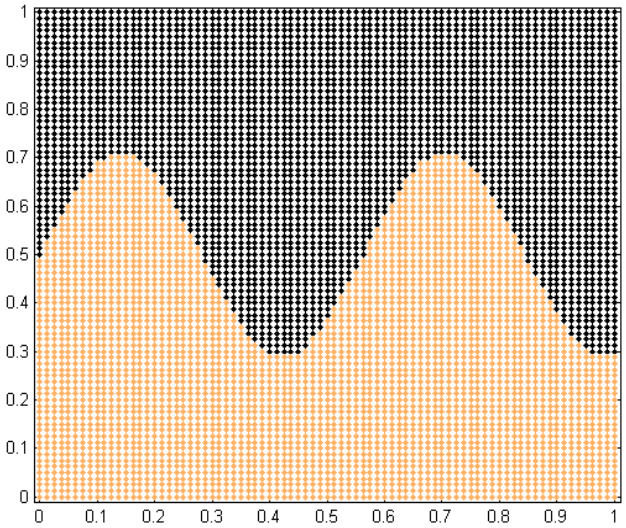


(c) D-MLP 3 neurony.

Rysunek 6.14: Obszary decyzyjne utworzone dla danych Polynomial, przez sieci MLP i D-MLP używające funkcji: f_3.

6.1.3 Sinus

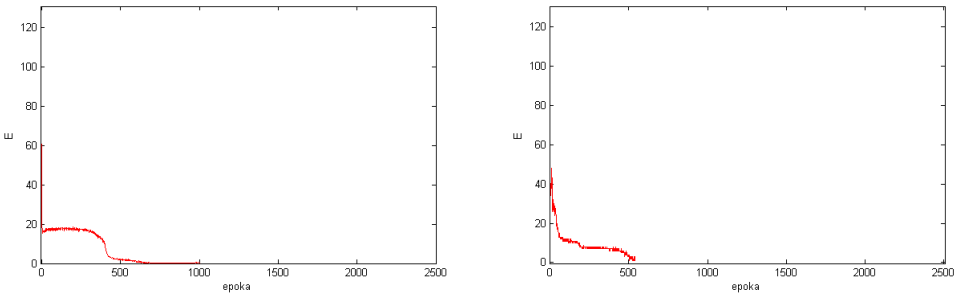
Dane Sinus składają się z 5651 punktów podzielonych na dwie klasy: po 3368 i 3193 obiektów. W sieci MLP wykorzystano 5 neuronów, D-MLP składa się z 3 neuronów. Zbieżność błędu jest w miarę szybka i porównywalna dla obu sieci, z tym że dla D-MLP do uzyskania poprawnego wyniku potrzebna jest większa liczba inicjalizacji niż dla MLP. Rezultaty klasyfikacji są zbliżone, choć MLP daje nieznacznie lepsze rezultaty.



Rysunek 6.15: Rozkład Sinus

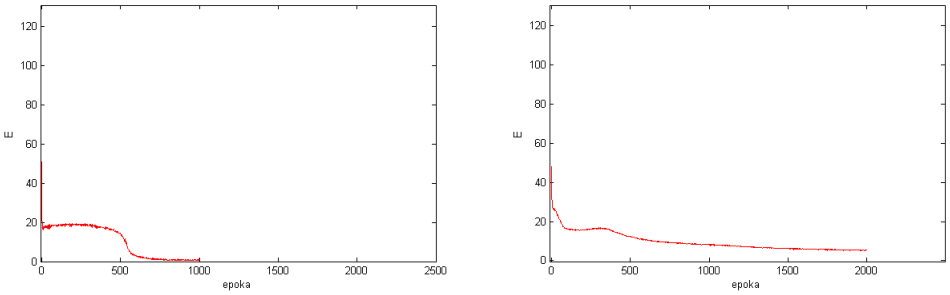
Funkcja transferu	Rodzaj sieci	Liczba neuronów	Poprawność klasyfikacji
f_1	DMLP	3	97.70 ± 0.66
	MLP	5	99.47 ± 0.37
f_2	DMLP	3	97.58 ± 0.61
	MLP	5	98.48 ± 0.61
f_3	DMLP	3	96.74 ± 0.97
	MLP	5	98.45 ± 0.44

Tabela 6.3: Wyniki otrzymane dla rozkładu Sinus dla sieci MLP i D-MLP metodą dziesięciokrotnej kroswalidacji.



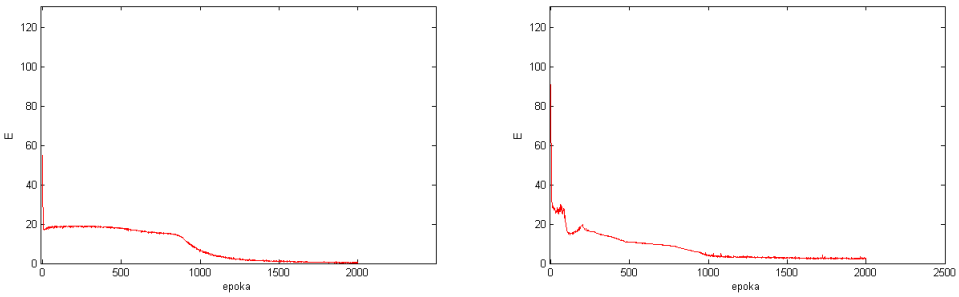
(a) MLP 5 neuronów (b) D-MLP 3 neurony

Rysunek 6.16: Zmiana błędu średniokwadratowego w zależności od epoki uczenia dla danych Sinus dla funkcji f_1.



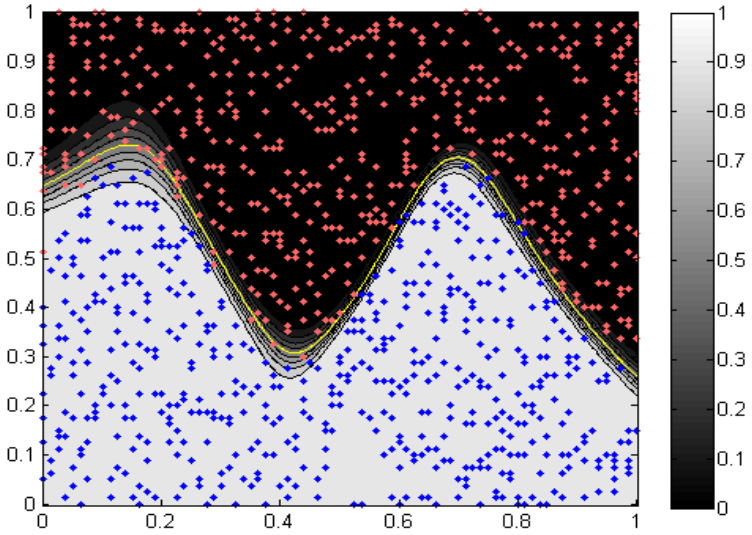
(a) MLP 5 neuronów (b) D-MLP 3 neurony

Rysunek 6.17: Zmiana błędu średniokwadratowego w zależności od epoki uczenia dla danych Sinus dla funkcji f_2.

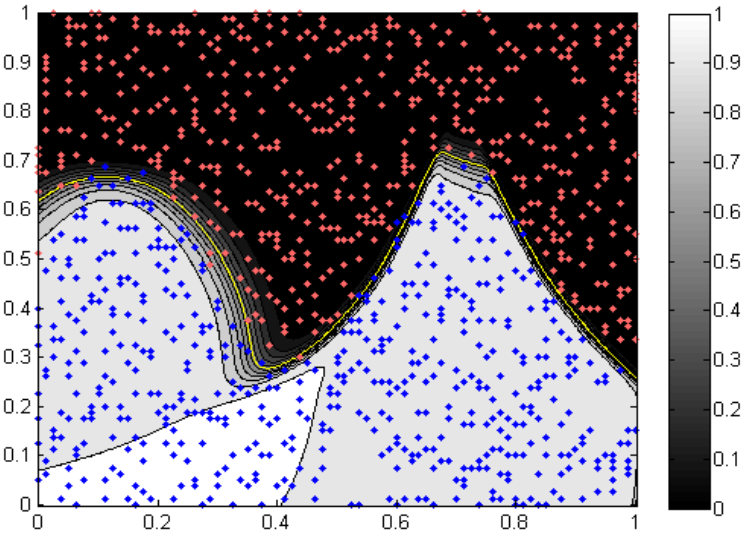


(a) MLP 5 neuronów (b) D-MLP 3 neurony

Rysunek 6.18: Zmiana błędu średniokwadratowego w zależności od epoki uczenia dla danych Sinus dla funkcji f_3.

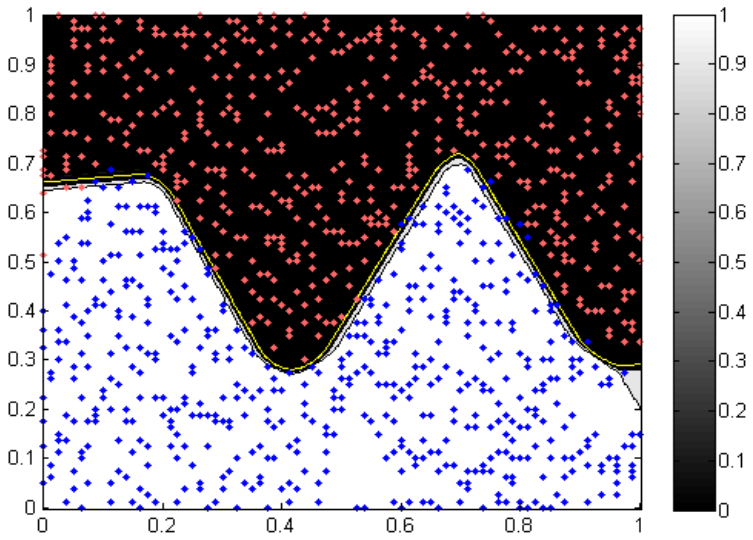


(a) MLP - 5 neuronów.

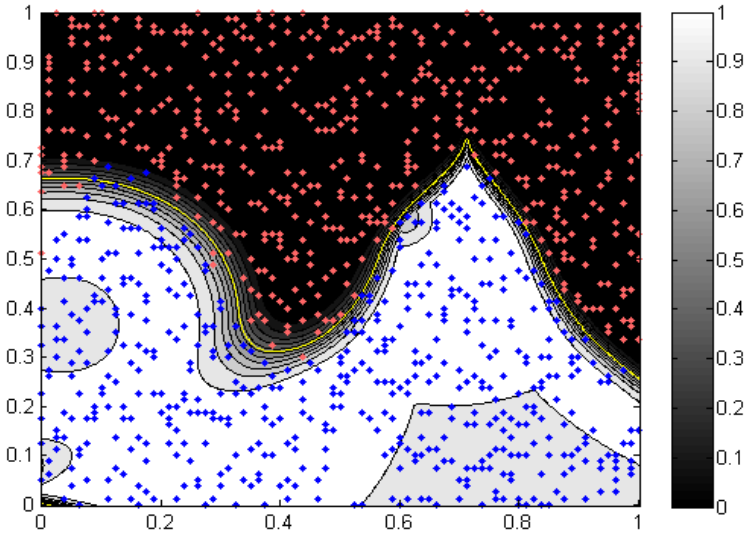


(b) D-MLP - 3 neurony.

Rysunek 6.19: Obszary decyzyjne utworzone dla danych Sinus przez sieci MLP i D-MLP używające funkcji f_1.

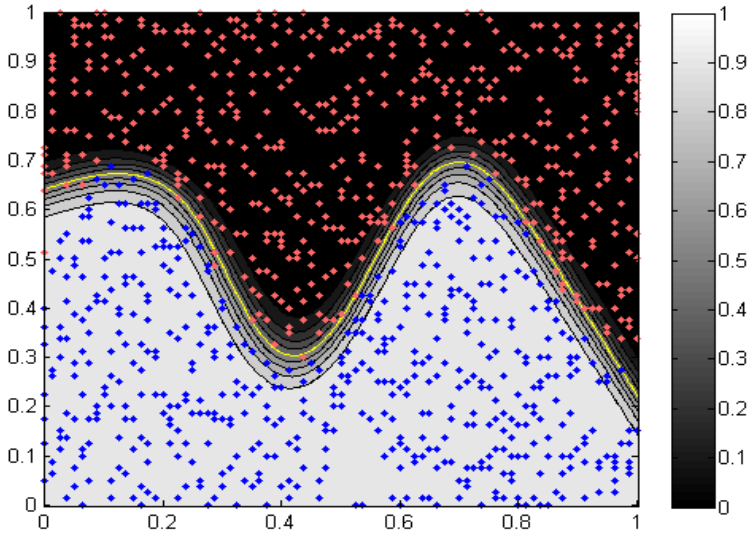


(a) MLP - 5 neuronów.

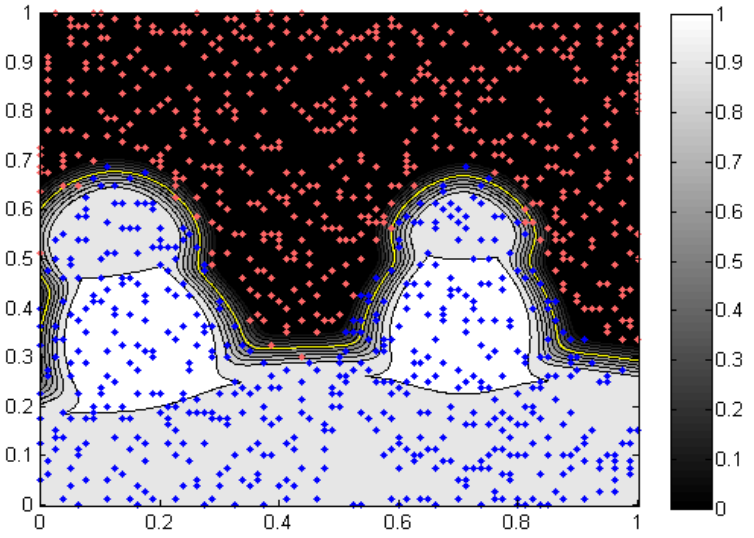


(b) D-MLP - 3 neurony.

Rysunek 6.20: Obszary decyzyjne utworzone dla danych Sinus przez sieci MLP i D-MLP używające funkcji f_2.



(a) MLP - 5 neuronów.

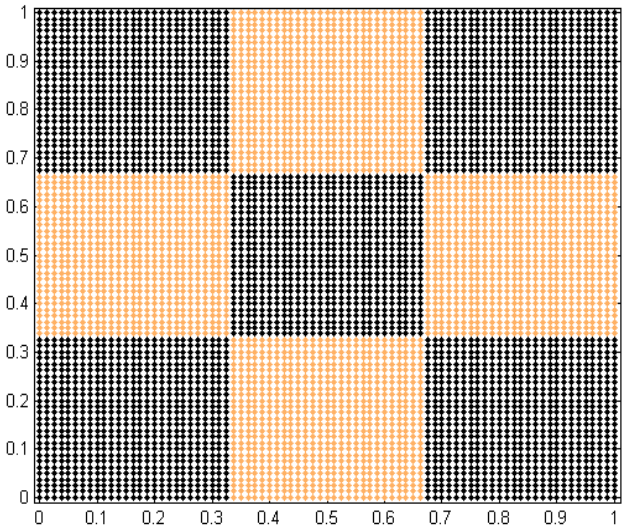


(b) D-MLP - 3 neurony.

Rysunek 6.21: Obszary decyzyjne utworzone dla danych Sinus przez sieci MLP i D-MLP używające funkcji f_3.

6.1.4 Checkerboard

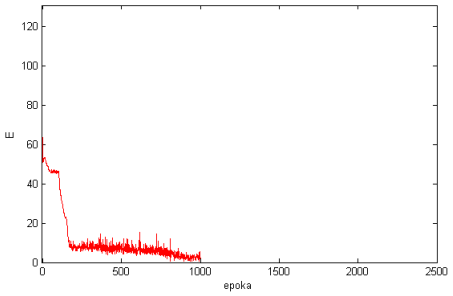
Dane Checkerboard składają się z 5651 punktów podzielonych na dwie klasy: po 3645 i 2916 obiektów. Checkerboard jest trudnym rozkładem. Sieć MLP daje w miarę dobre wyniki, choć potrzebuje do tego aż dwóch warstw, każda po 4 neurony. Charakterystyczne dla tego problemu jest to, że sieć MLP z funkcją f_3 dość wolno zbiega do minimum (Rys. 6.25a). Sieć D-MLP złożona z 4 neuronów jest w stanie osiągać poprawne rozwiązania (Rys. 6.26b), zdarza się to jednak rzadko i typowe granice decyzyjne mają raczej jakość porównywalną z Rys. 6.27b i 6.28b. Proces nauki w przypadku D-MLP jest bardzo trudny, sieć trzeba inicjalizować dużą ilość razy aby uzyskać zadowalający efekt. Zwraca uwagę duża wariancja.



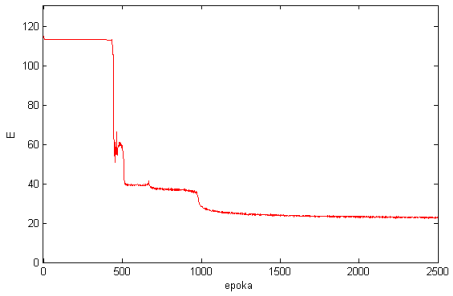
Rysunek 6.22: Rozkład Checkerboard

Funkcja transferu	Rodzaj sieci	Liczba neuronów	Poprawność klasyfikacji
f_1	DMLP	4	90.42 ± 3.67
	MLP	4+4	94.97 ± 1.97
f_2	DMLP	4	86.69 ± 5.05
	MLP	4+4	94.62 ± 1.63
f_3	DMLP	4	84.94 ± 6.11
	MLP	4+4	96.40 ± 0.93

Tabela 6.4: Wyniki otrzymane dla rozkładu Checkerboard dla sieci MLP i D-MLP metodą dziesięciokrotnej kroswalidacji.

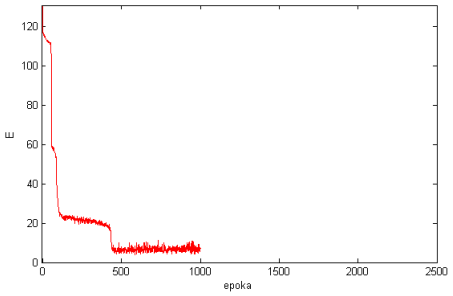


(a) MLP 4+4 neurony

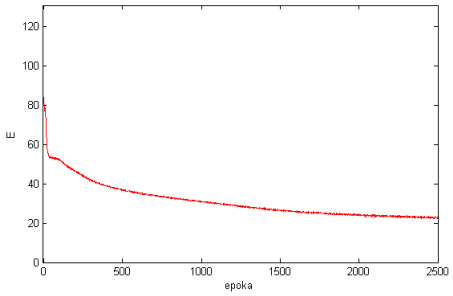


(b) D-MLP 4 neurony

Rysunek 6.23: Zmiana błędu średniokwadratowego w zależności od epoki uczenia dla danych Checkerboard dla funkcji f_1.

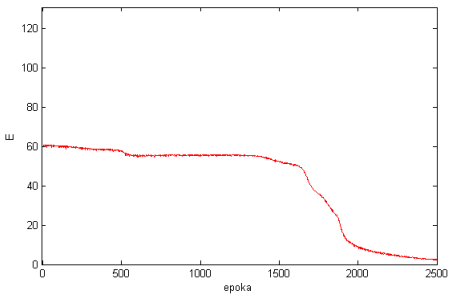


(a) MLP 4+4 neurony

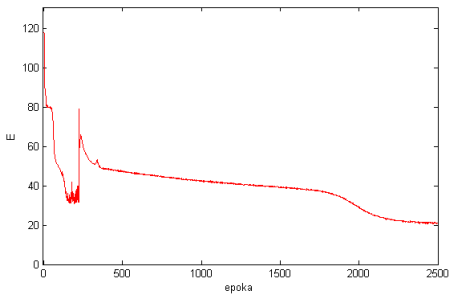


(b) D-MLP 4 neurony

Rysunek 6.24: Zmiana błędu średniokwadratowego w zależności od epoki uczenia dla danych Checkerboard dla funkcji f_2.

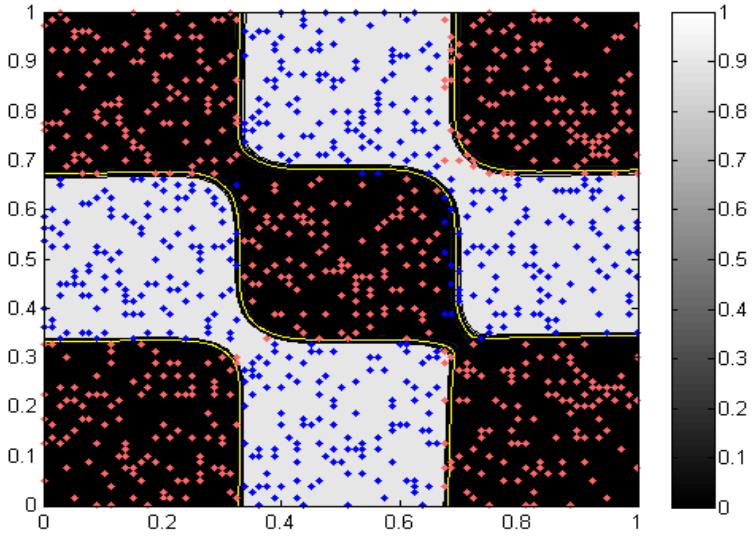


(a) MLP 4+4 neurony

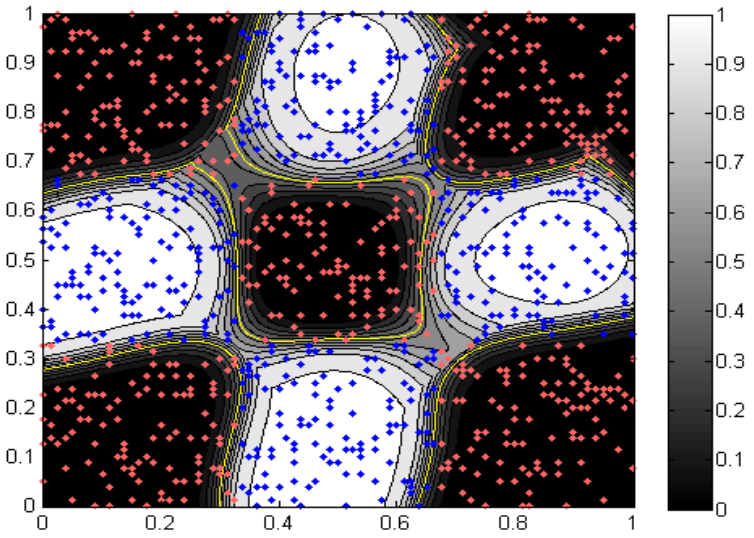


(b) D-MLP 4 neurony

Rysunek 6.25: Zmiana błędu średniokwadratowego w zależności od epoki uczenia dla danych Checkerboard dla funkcji f_3.

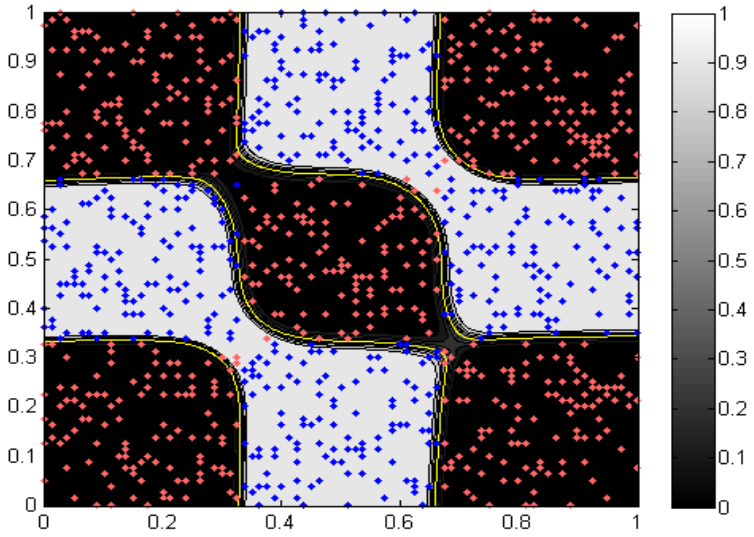


(a) MLP - 4+4 neurony neuronów.

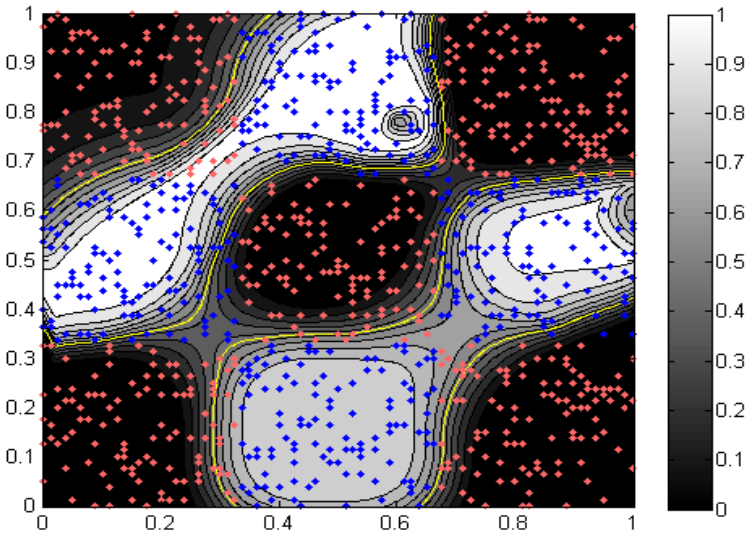


(b) D-MLP - 4 neurony.

Rysunek 6.26: Obszary decyzyjne utworzone dla danych Checkerboard przez sieci MLP i D-MLP używające funkcji f_1.

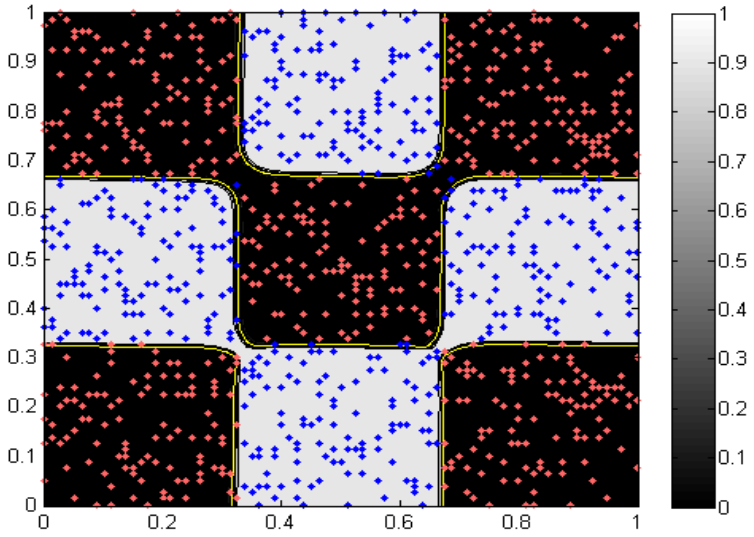


(a) MLP - 4+4 neurony neuronów.

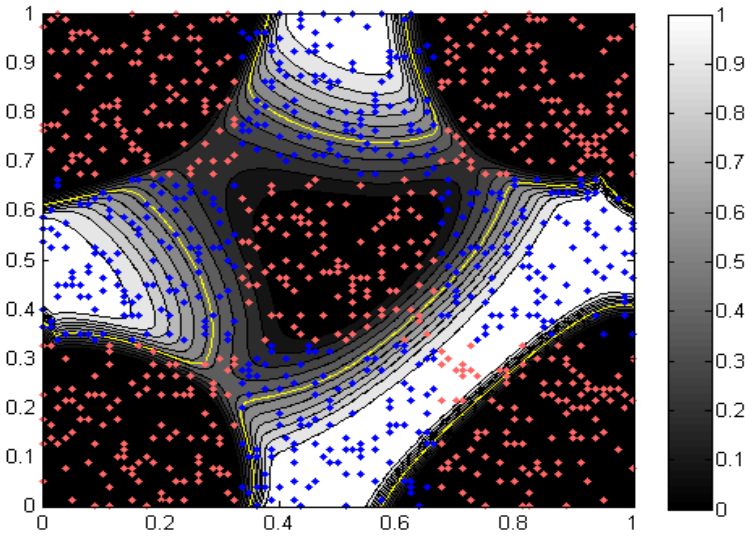


(b) D-MLP - 4 neurony.

Rysunek 6.27: Obszary decyzyjne utworzone dla danych Checkerboard przez sieci MLP i D-MLP używające funkcji f_2.



(a) MLP - 4+4 neurony neuronów.



(b) D-MLP - 4 neurony.

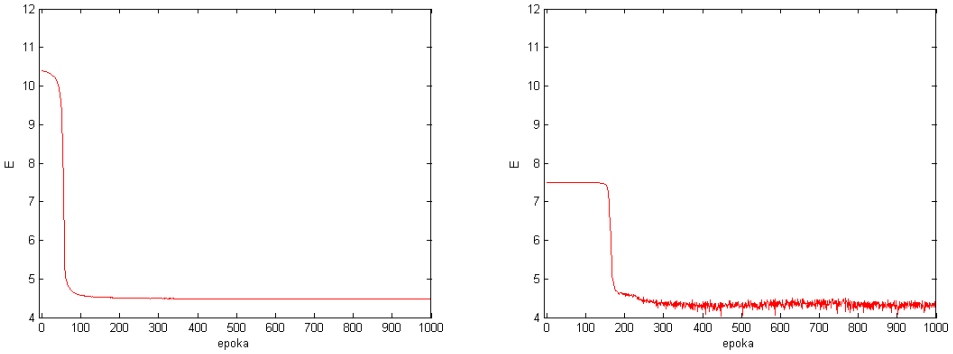
Rysunek 6.28: Obszary decyzyjne utworzone dla danych Checkerboard przez sieci MLP i D-MLP używające funkcji f_3.

6.2 Dane Appendicitis

Dane Appendicitis dotyczą zapalenia wyrostka robaczkowego, pochodzą od Shalom Weiss [15]. Cały zbiór składa się ze 106 wektorów, każdy po 8 cech. Dane podzielone są na dwie klasy: 85 przypadków ostrego zapalenia i 21 obiektów należących do drugiej klasy opisującej inne problemy. Dane zostały znormalizowane z 5% obcięciem i podanne testowi dziesięciokrotnej kroswalidacji dla sieci MLP i D-MLP. Dla obu rodzajów sieci błąd szybko zbiega do wartości minimalnej, sieć D-MLP dość łatwo uczy się dla tego rozkładu. Widać, że zastosowanie sieci D-MLP daje nieco lepsze wyniki niż MLP przy tej samej liczbie wykorzystanych neuronów. Skuteczność innych metod klasyfikacji dla tego zbioru znaleźć można na stronie [15].

Funkcja transferu	Rodzaj sieci	Liczba neuronów	Poprawność klasyfikacji
f_1	DMLP	1	86.33 ± 6.9
	MLP	1	86.00 ± 8.00
f_2	DMLP	1	88.00 ± 9.80
	MLP	1	85.33 ± 6.00
f_3	DMLP	1	88.67 ± 10.12
	MLP	1	86.67 ± 7.46

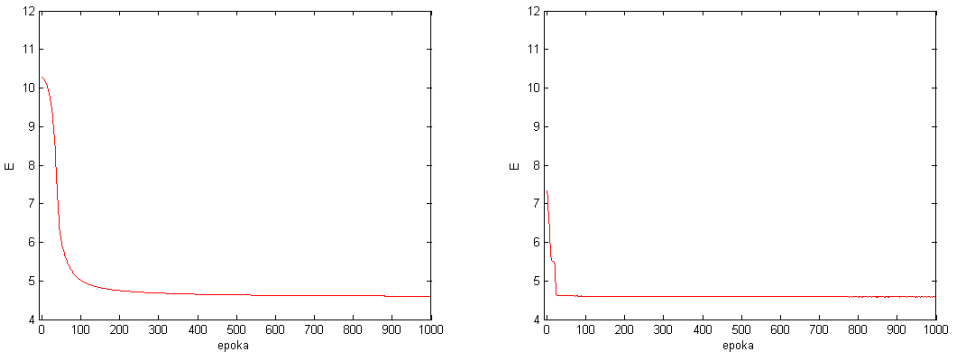
Tabela 6.5: Wyniki otrzymane dla danych Appendicitis dla sieci MLP i D-MLP metodą dziesięciokrotnej kroswalidacji.



(a) MLP 1 neuron

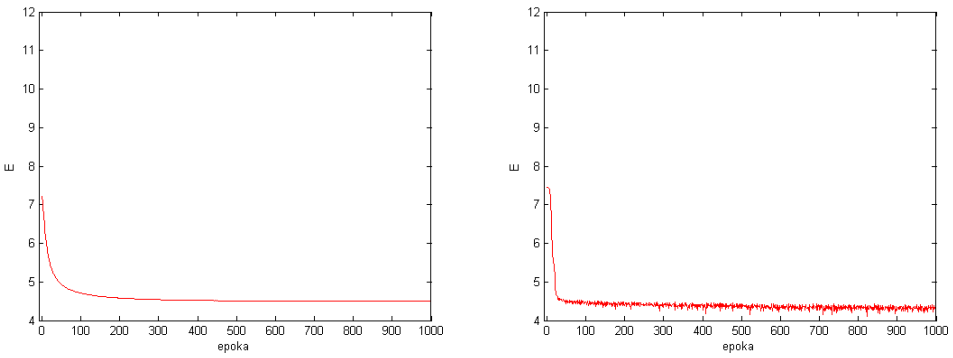
(b) D-MLP 1 neuron

Rysunek 6.29: Zmiana błędu średniokwadratowego w zależności od epoki uczenia dla danych Appendicitis dla funkcji f_1.



(a) MLP 1 neuron (b) D-MLP 1 neuron

Rysunek 6.30: Zmiana błędu średniokwadratowego w zależności od epoki uczenia dla danych Appendicitis dla funkcji f_2.



(a) MLP 1 neuron (b) D-MLP 1 neuron

Rysunek 6.31: Zmiana błędu średniokwadratowego w zależności od epoki uczenia dla danych Appendicitis dla funkcji f_3.

Rozdział 7

Zakończenie

W niniejszej pracy dokonano przeglądu ewolucji kształtów możliwych do wygenerowania przez sieć D-MLP. W tym celu wykorzystano funkcję aktywacji w postaci(4.7). Podjęto też próbę uporządkowania tych kształtów w zależności od odpowiednich parametrów. Integralną częścią pracy była również implementacja zwykłej sieci MLP i sieci D-MLP wykorzystującej normę Minkowskiego oraz porównanie działania tych sieci na kilku zbiorach danych. Wykazano, że sieci D-MLP są niewątpliwie dość ciekawym rozwiązaniem, pozwalającym na uzyskanie elastycznych obszarów decyzyjnych przy znacznie zredukowanej złożoności sieci. Dla niektórych rozkładów danych, (na przykład Circle) sprawdzają się świetnie. Niestety sieci te w porównaniu do MLP bardzo trudno się uczą, znaczenie może tu mieć nieliniowa zależność aktywacji od wartości parametrów. Odpowiednia inicjalizacja powinna w pewnym stopniu zredukować ten problem, dlatego z pewnością dopracowanie metody inicjalizacji sieci powinno być wzięte pod uwagę.

Praca nie wyczerpuje problemu D-MLP. Testy zostały przeprowadzone jedynie dla paru sztucznych dwuwymiarowych rozkładów danych, oraz na jednym zbiorze danych medycznych. Interesującym mogłoby być sprawdzenie jak D-MLP radzi sobie w innych problemach o większej liczbie wymiarów i rzeczywistych rozkładach. Otwarte pozostaje pytanie na ile różne formy aktywacji sieci D-MLP są sobie równoważne. W pracy szerzej zajęto się transformacją (4.1) i jej uogólnieniem, pomijając zupełnie przekształcenie (4.2). Ograniczono się też do normy Minkowskiego jako funkcji odległości, podczas gdy zastosowanie innych miar odległości także mogłoby dać ciekawe rezultaty.

Na końcu trzeba jeszcze wspomnieć o wykorzystanych w pracy funkcjach wyjścia. Zastosowano trzy funkcje, oznaczane jako: f_1, f_2, f_3, wszystkie sprawdziły się dobrze, efekty ich działania są porównywalne. Jedynie funkcja f_3 w D-MLP sprawiała pewne kłopoty. Podczas testowania sieci okazało się, że parametr regulujący nachylenie funkcji f_3 ma ograniczone działanie. Dlatego w sieciach D-MLP, gdzie zależność nachylenia od wielkości wag jest zazwyczaj różna niż w MLP, aby sieć była w stanie dobrze się uczyć konieczne stało się dodanie do funkcji nowego parametru.

Podsumowując, sieci D-MLP są na pewno interesującą alternatywą dla zwykłych sieci wielowarstwowych. Dla niektórych rozkładów ich efektywność jest porównywalna, a czasami nawet lepsza niż sieci MLP, a ilość neuronów wykorzystanych do rozwiązania zagadnienia mniejsza. Tym niemniej przydatność sieci D-MLP wiąże się z otwartą jak na razie kwestią poprawy ich zdolności do nauki.

Bibliografia

[1] R. Adamczak, *Zastosowanie sieci neuronowych do klasyfikacji danych doświadczalnych.* Praca doktorska, UMK, Toruń, 2001.

[2] E. Alpaydm, *Introduction to Machine Learning.* The MIT Press, Cambridge, 2004.

[3] C.A.L. Bailer-Jones, M. Irwin,G. Gilmore T. von Hippel, Physical parametrization of stellar spectra - The neural network approach. *MNRAS* vol. 292, p. 157.

[4] Christopher M. Bishop, *Neural Networks for Pattern Recognition.* Oxford University Press, 2002.

[5] Sing-Tze Bow, *Pattern Recognition and Image Preprocessing.* Marcel Dekker Inc., 2002.

[6] W. Duch, Uncertainty of data, fuzzy membership functions, and multi-laye perceptrons.*IEEE Transactions on Neural Networks* 16(1): 10-23, 2005.

[7] W. Duch, R. Adamczak, G.H.F Diercksen, Classification, Association and Pattern Completion using Neural Similarity Based Methods. *Applied Mathematics and Computer Science.* 10:4 (2000) 101-120.

[8] W. Duch, R. Adamczak, G. H. F. Diercksen. Distance-based multilayer perceptrons. *International Conference on Computational Intelligence for Modelling Control and Automation.* s. 75 80, Vienna, Austria, 1999.

[9] W. Duch, R. Adamczak, G. H. F. Diercksen. Neural networks in non-euclidean spaces. *Neural Processing Letters.* 10:110, 1999.

[10] W. Duch, N. Jankowski. Survey of Neural Transfer Functions. *Neural Computing Surveys 2.* (1999) p. 163-213.

[11] W. Duch, N. Jankowski. Taxonomy of neural transfer functions. *IEEE, International Joint Conference on Neural Networks 2000 (IJCNN).* Vol. III, pp. 477-484.

[12] L. Fausett, *Fundamentals of neural networks: architectures, algorithms and applications.* Prentice-Hall, Inc. , 1994.

[13] T. Fawcett, Machine learning classifier gallery.
http://home.comcast.net/~tom.fawcett/public_html/ML-gallery/pages/index.html

[14] N. Jankowski, *Ontogeniczne sieci neuronowe w zastosowaniu do klasyfikacji danych medycznych.* Praca doktorska, UMK, Toruń, 1999.

[15] Podstrona Katedry Informatyki Stosowanej UMK.
http://www.is.umk.pl/projects/datasets.html

[16] T. Korol, B. Prusak, *Upadłość przedsiębiorstw a wykorzystanie sztucznej inteligencji.* CeDeWu, Warszawa, 2009.

[17] O. Lahav, A. Naim , L. Jr. Sodre,M.C. Storne-Lombardi, Neural Comptutation as a tool for galaxy classification: methods and examples. Institute od Astronomy, Cambridge, Techni- cal report CB3 OHA, 1995.

[18] D. Michie, D.J. Spiegelhalter, C.C. Taylor, *Machine learning, neural and statistical classification.* Elis Horwood, London 1994.

[19] A. Naim, O. Lahav, L. Sodre Jr, M. C. Storrie-Lombardi, Automated morphological classification of apm galaxies by supervised artificial neural networks. *Astro-ph/9503001.*

[20] S. Osowski, *Sieci neuronowe w ujęciu algorytmicznym.* Wydawnictwa Naukowo-Techniczne, Warszawa, 1996.

[21] R. Tadeusiewicz, *Sieci neuronowe.* Akademicka Oficyna Wydawnicza, Warszawa, 1993.

[22] J. Żurada, M. Barski, W. Jędruch, *Sztuczne sieci neuronowe. Podstawy teori i zastosowania.* Wydawnictwo Naukowe PWN, Warszawa, 1996.

Spis rysunków

Spis tabel