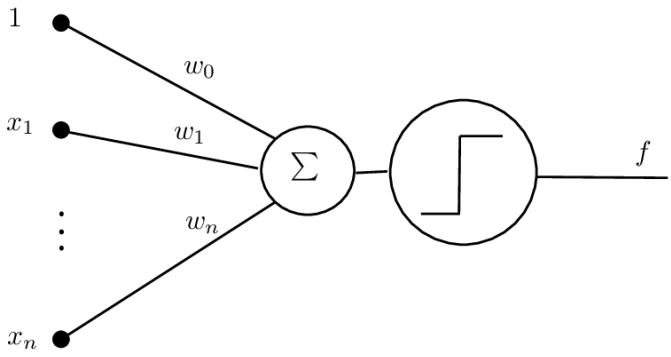


Rys. 5.1: Schemat graficzny perceptronu prostego. (źródło: *opracowanie własne*)

czona symbolicznie na wykresie). Myśląc ponownie o biologicznych neuronach, można powiedzieć, że działanie funkcji aktywacji odzwierciedla sposób wzbudzania odpowiedzi neuronu.

Przyjmując jako próg decyzyjny wartość 0, omówiony schemat obliczeń można zapisać następująco:

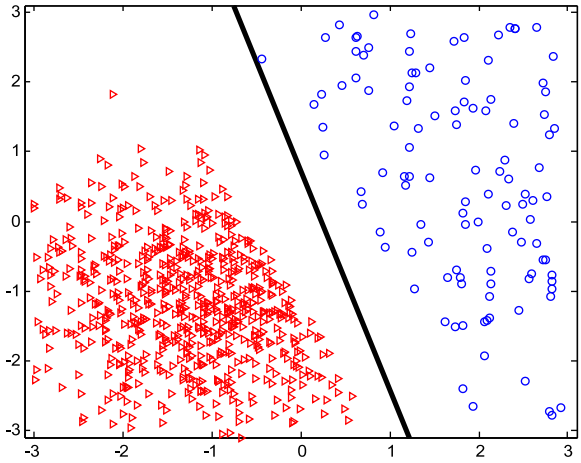
$$s = w_0 + w_1 x_1 + \cdots + w_n x_n. \tag{5.1}$$

$$f(s) = \begin{cases} 1, & \text{dla } s > 0; \\ -1, & \text{dla } s \leqslant 0. \end{cases} \tag{5.2}$$

5.1.2 Notacja, dane, sens geometryczny

Niech $D = \{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,m}$ oznacza zbiór danych uczących (przykładów) zawierający pary, w których $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \in \mathbb{R}^n$ są wektorami cech rzeczywistoliczbowych, a $y_i \in \{-1, 1\}$ skojarzonymi z nimi etykietami klas. A zatem w przypadku perceptronu prostego o przykładach uczących można myśleć jako o punktach w przestrzeni $\mathbb{R}^n$ pokolorowanych za pomocą dwóch kolorów[2]. Ilustrację dla takiej interpretacji stanowi rys. 5.2, gdzie pokazano przykładową klasyfikację binarną na płaszczyźnie ($n = 2$). W tym przypadku każdy przykład (lub inaczej — punkt danych) posiada dwie cechy rzeczywiste, które możemy traktować jak współrzędne kartezjańskie. Wskazana na rysunku czarna prosta stanowi liniową granicę decyzyjną pomiędzy dwiema klasami punktów. Oczywiście jest to prosta przykładowa, a w pokazanej sytuacji można wskazać wiele innych prostych (nieskończenie wiele) prawidłowo separujących dane.

[2]Perceptron prosty wymaga cech rzeczywistoliczbowych. Jest to ograniczenie oznaczające niemożność pracy na zmiennych (cechach) wyliczeniowych, takich jak np. kolor oczu.



Rys. 5.2: Przykład klasyfikacji binarnej na płaszczyźnie. (źródło: *opracowanie własne*)

Równanie prostej można zapisać jako $w_0 + w_1x_1 + w_2x_2 = 0$. W przypadku $n = 3$, tzn. gdy dane przebywają w trójwymiarowej przestrzeni cech, liniową granicą decyzyjną byłaby pewna płaszczyzna o równaniu: $w_0 + w_1x_1 + w_2x_2 + w_3x_3 = 0$. A zatem w ogólności można powiedzieć, że zadaniem perceptronu prostego jest znalezienie odpowiednich współczynników wielowymiarowej płaszczyzny w przestrzeni $\mathbb{R}^n$ (nazywanej także *hiperpłaszczyzną*) o równaniu:

$$w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = 0. \tag{5.3}$$

Uściślając, chcemy znaleźć pewien wektor współczynników $\mathbf{w} = (w_0, w_1, \ldots, w_n)$, definiujący konkretne równanie płaszczyzny, która właściwie separuje dane uczące wedle ich klasy, tj. taki wektor, że:

$$\forall i, y_i = 1 \quad w_0 + w_1x_{i1} + w_2x_{i2} + \cdots + w_nx_{in} > 0. \tag{5.4}$$

i

$$\forall i, y_i = -1 \quad w_0 + w_1x_{i1} + w_2x_{i2} + \cdots + w_nx_{in} \leqslant 0. \tag{5.5}$$

W nomenklaturze sztucznych sieci neuronowych poszukiwane współczynniki nazywane są często współczynnikami wagowymi lub krótko *wagami*.

Patrząc ponownie na schemat graficzny (rys. 5.1) pewnego komentarza wymaga obecność sygnału wejściowego ustalonego na 1 powiązanego ze współczynnikiem

w_0. Oczywiście stała wartość 1 nie stanowi żadnej istotnej informacji wejściowej, która w jakikolwiek sposób różnicowałaby obiekty. Należy jednak zrozumieć, że tym sposobem wprowadzamy współczynnik w_0, który stanowi wyraz wolny w równaniu płaszczyzny. Bez obecności tego wyrazu do konkurencji wchodziłyby tylko te płaszczyzny, które przebiegają przez środek przyjętego układu współrzędnych. Tym samym nie wszystkie zbiory danych, dla których istnieje granica liniowa pomiędzy klasami, mogłyby zostać skutecznie odseparowane. A zatem obecność wyrazu wolnego w_0 jest podyktowana geometrycznie.

Wprowadzimy teraz notację wektorów $\mathbf{x}_i$ rozszerzoną o dodatkową cechę $x_0 = 1$ (dla wygody dalszych zapisów matematycznych):

$$\mathbf{x}_i = (1, x_{i1}, x_{i2}, \ldots, x_{in}).$$

Dzięki temu będziemy mogli iloczyn skalarny wektora wag $\mathbf{w}$ i wektora cech $\mathbf{x}_i$ zapisać krótko jako parę: $\langle \mathbf{w}, \mathbf{x}_i \rangle$:

$$\langle \mathbf{w}, \mathbf{x}_i \rangle = \sum_{j=0}^{n} w_j x_{ij}. \tag{5.6}$$

5.1.3 Algorytm uczenia on-line dla perceptronu prostego

Podany poniżej algorytm 12, nazywany zwyczajowo „regułą perceptronu”[3], przedstawia sposób uczenia on-line dla perceptronu prostego. Uczenie on-line oznacza

Algorytm 12 „Reguła perceptronu”

1: **procedura** PERCEPTRONLEARNINGRULE(D, η)
2: $\mathbf{w}(0) := (0, 0, \ldots, 0)$ ▷ początkowy wektor wag
3: $k := 0$ ▷ licznik kroków
4: **dopóki** zbiór błędnie sklasyfikowanych punktów $E = \{(\mathbf{x}_i, y_i) \colon y_i \neq f(\langle \mathbf{w}(k), \mathbf{x}_i \rangle)\}$ jest niepusty **wykonaj**
5: wylosuj ze zbioru E dowolną parę $(\mathbf{x}_i, y_i)$
6: popraw wektor wag wg wzoru: $\mathbf{w}(k+1) := \mathbf{w}(k) + \eta y_i \mathbf{x}_i$
7: $k := k + 1$
8: **zwróć** $\mathbf{w}(k)$

w ogólności, że poprawki współczynników wagowych odbywają się na podstawie obejrzanego pojedynczego przykładu[4]. Oprócz zbioru danych uczących D dodatkowym argumentem podawanym przez użytkownika na wejście algorytmu jest liczba $\eta \in (0, 1]$ nazywana *współczynnikiem uczenia* (ang. *learning rate* lub *learning*

[3] Lub alternatywnie „regułą delty” w przypadku nieco innej formy zapisu algorytmu.

[4] W odróżnieniu, uczenie off-line (stosowane czasami w bardziej zaawansowanych sieciach neuronowych) dokonuje pojedynczej poprawki po obejrzeniu całego zbioru uczącego.

coefficient). Współczynnik ten decyduje o wielkości dokonywanych poprawek i stanowi on analogię do długości kroku w metodach optymalizacji. Oznaczenie $\mathbf{w}(k)$, używane w zapisie algorytmu, oznacza wektor wag w k-tym kroku (lub inaczej po wykonaniu k poprawek). Licznik k pełni rolę czysto informacyjną i będzie rozważany podczas analizy zbieżności algorytmu. W implementacji może zostać pominięty.

Najważniejszą rolę w algorytmie odgrywa krok poprawiania (aktualizacji) wektora wag wg wzoru:

$$\mathbf{w}(k+1) := \mathbf{w}(k) + \eta y_i \mathbf{x}_i. \tag{5.7}$$

Postać tego wzoru może być dla czytelnika na ten moment niejasna. Dlaczego składnik poprawki wynosi akurat $\eta y_i \mathbf{x}_i$? Poniżej podamy pewną uproszczoną motywację stojącą za tym wzorem, natomiast jego poprawność stanie się w pełni jasna po analizie zbieżności algorytmu.

Przypatrzmy się przez chwilę wyrażeniu $y_i \langle \mathbf{w}(k), \mathbf{x}_i \rangle$. Jeżeli punkt danych $(\mathbf{x}_i, y_i)$ jest aktualnie błędnie klasyfikowany, to na pewno $y_i \langle \mathbf{w}(k), \mathbf{x}_i \rangle \leqslant 0$. Mówiąc dokładniej, jeżeli y_i różni się od wyjścia perceptronu f, to rozważane wyrażenie jest iloczynem dwóch czynników przeciwnych znaków (lub wynosi zero tylko w przypadku, gdy $y_i = 1$ i $\langle \mathbf{w}(k), \mathbf{x}_i \rangle = 0$, co powoduje $f(0) = -1$). Można zatem skonstruować następującą wielkość reprezentującą błąd dla i-tego przykładu:

$$e_i = -y_i \langle \mathbf{w}(k), \mathbf{x}_i \rangle. \tag{5.8}$$

Gdy wyjście perceptronu jest niezgodne z oczekiwaną etykietą klasy, to $e_i > 0$ — błąd jest obecny. Gdy zaś wyjście perceptronu jest zgodne z oczekiwaną etykietą klasy, to $e_i \leqslant 0$, co można interpretować jako brak błędu (lub umownie jako błąd ujemny). Podstawowa technika znana z metod optymalizacji, nakazuje wyznaczyć gradient (czyli wektor pochodnych cząstkowych funkcji błędu ze względu na poszczególne parametry) i poprawiać optymalizowany wektor parametrów w kierunku przeciwnym do gradientu, tzn.:

$$\mathbf{w}(k+1) := \mathbf{w}(k) - \frac{\partial e_i}{\partial \mathbf{w}(k)}. \tag{5.9}$$

Łatwo sprawdzić, że

$$\frac{\partial e_i}{\partial \mathbf{w}(k)} = -y_i \mathbf{x}_i, \tag{5.10}$$

co uzasadnia postać wzoru (5.7) na rzecz i-tego punktu danych. Jest to jednak uzasadnienie tylko częściowe. Oznacza tylko bowiem to, że konsekwentne stosowanie

W procesie uczenia duże znaczenie ma również współczynnik uczenia η. Niestety nie istnieje ogólna metoda doboru jego wartości. Duża wartość współczynnika powoduje duże zmiany wag i gdy funkcja celu jest stroma, może to skutkować oscylacją wokół rozwiązania. Niska wartość współczynnika uczenia, sprawdzająca się przy stromych funkcjach celu, wydłuża proces uczenia na powierzchniach, gdzie wartości gradientu są niewielkie. Współczynnik ten można w sposób równoważny interpretować jako wielkość kroku poprawki w metodach gradientowych. Wartość współczynnika zwykle dobiera się eksperymentalnie i zależy ona od problemu, który należy rozwiązać [39].

Istnieje wiele modyfikacji metody wstecznej propagacji błędu, mających na celu lepsze radzenie sobie z problemem minimów lokalnych, a także przyspieszenie procesu uczenia. Niektóre z tych modyfikacji to: uczenie z rozpędem, algorytm zmiennej metryki, RPROP (ang. *resilient backpropagation*) [41], metoda gradientów sprzężonych, algorytm Levenberga-Marquardta [17] czy uczenie oparte na rekurencyjnej metodzie najmniejszych kwadratów [3].

5.3.1 Przykład prostej sieci neuronowej

Sieci neuronowe mogą tworzyć różne skomplikowane struktury. W tym miejscu postaramy się przybliżyć jedną z podstawowych i najprostszych architektur sieci neuronowej: sieci z jedną warstwą ukrytą i jednym wyjściem, która może zostać użyta do zadania aproksymacji danych. Schemat sieci przedstawia rysunek 5.6. Sieć ta skład się z N neuronów. W każdym z neuronów obliczana jest odpowiedź bloku sumowania s_i:

$$s_i = v_{i0} + \sum_{j=1}^{n} v_{ij} \cdot x_j. \tag{5.30}$$

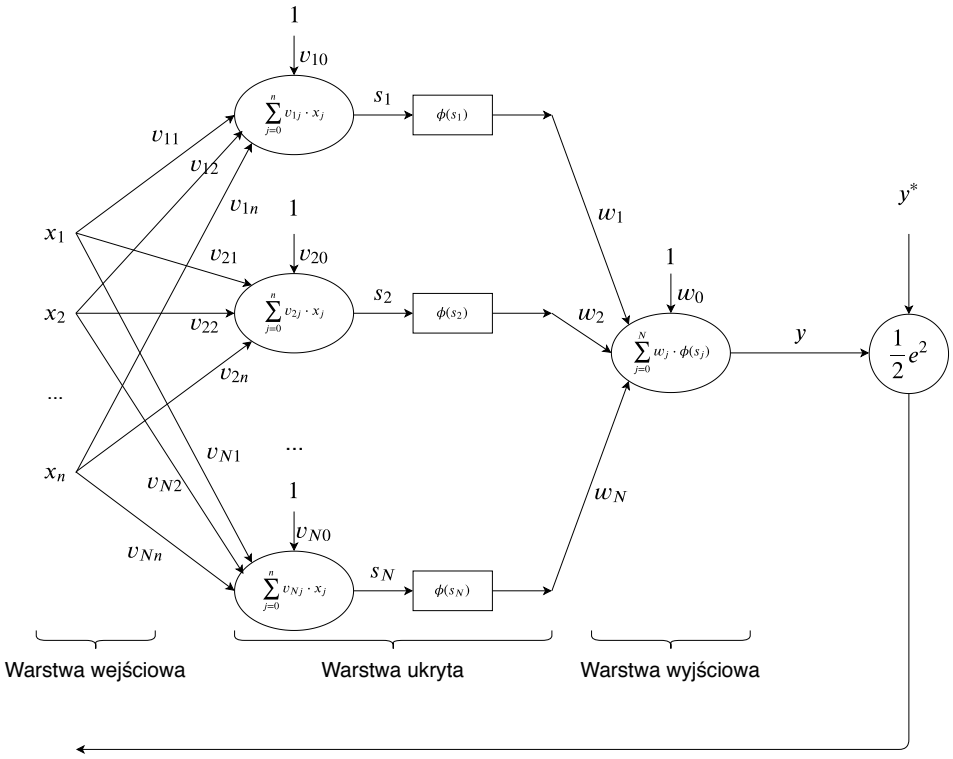
Następnie obliczana jest odpowiedź bloku funkcji aktywacji:

$$\phi(s_i) = \frac{1}{1+e^{-s_i}}. \tag{5.31}$$

W powyższym przykładzie została użyta sigomoidalna unipolarna[6] funkcja aktywacji. Na wyjściu sieci znajduje się pojedynczy neuron z liniową funkcją aktywacji. Liniowa funkcja aktywacji może zostać pominięta, a działanie warstwy wyjściowej posiada następującą postać:

$$y = w_0 + \sum_{j=1}^{N} w_j \, \phi(s_j). \tag{5.32}$$

[6]Przymiotnik unipolarna oznacza, że funkcja przyjmuje wartości z przedziału $[0,\ 1]$, funkcja bipolarna przyjmowałaby wartości z przedziału $[-1,\ 1]$.



Rys. 5.6: Szczegółowy schemat działania sieci neuronowej z jedną warstwą ukrytą. (źródło: *opracowanie własne*)

Wzór na uczenie sieci neuronowej bazuje na wzorze (5.26). Błąd sieci może zostać uproszczony do postaci:

$$\frac{1}{2}e^2 = \frac{1}{2}(y - y^*)^2 \tag{5.33}$$

gdzie y to odpowiedź sieci neuronowej, a y^* to wartość oczekiwana dla danych wejściowych. Znając ogólny wzór na korekcję wag (5.26), można wyprowadzić z niego wzory na korekcję wag w oraz v. W przypadku wag v wyprowadzenie ma

następującą postać:

$$\begin{aligned}\frac{\partial \frac{1}{2}e^2}{\partial v_{ij}} &= \frac{\partial \frac{1}{2}e^2}{\partial y}\cdot\frac{\partial y}{\partial \phi(s_i)}\cdot\frac{\partial \phi(s_i)}{\partial s_i}\cdot\frac{\partial s_i}{\partial v_{ij}} \\ &= \frac{\partial \frac{1}{2}(y-y^*)^2}{\partial y}\cdot\frac{\partial (w_0+w_1\phi(s_1)+\cdots+w_i\phi(s_i)+\cdots+w_N\phi(s_N))}{\partial \phi(s_i)}\cdot \\ &\quad\cdot\frac{\partial \phi(s_i)}{\partial s_i}\cdot\frac{\partial (v_{i0}+v_{i1}x_1+\cdots+v_{ij}x_j+\cdots+v_{in}x_n)}{\partial v_{ij}} \\ &= (y-y^*)\cdot w_i\cdot\phi(s_i)\,(1-\phi(s_i))\cdot x_j.\end{aligned} \tag{5.34}$$

Z kolei wyprowadzenie dla wag w ma postać:

$$\begin{aligned}\frac{\partial \frac{1}{2}e^2}{\partial w_j} &= \frac{\partial \frac{1}{2}e^2}{\partial y}\cdot\frac{\partial y}{\partial w_j} \\ &= \frac{\partial \frac{1}{2}(y-y^*)^2}{\partial y}\cdot\frac{\partial (w_0+w_1\phi(s_1)+\cdots+w_j\phi(s_j)+\cdots+w_N\phi(s_N))}{\partial w_j} \\ &= (y-y^*)\cdot\phi(s_j).\end{aligned} \tag{5.35}$$

W efekcie otrzymujemy dwa wzory na korekcje wag sieci:

$$v_{ij} = v_{ij} - \eta\cdot(y-y^*)\cdot w_i\cdot\phi(s_i)(1-\phi(s_i))\cdot x_j, \tag{5.36}$$

$$w_j = w_j - \eta\cdot(y-y^*)\cdot\phi(s_j). \tag{5.37}$$

5.3.2 Uczenie z rozpędem — Momentum Backpropagation

Metoda ta polega na dodaniu do wzoru na korektę dowolnej wagi (parametru) sieci tzw. składnika momentum (lub inaczej rozpędu), który jest ułamkniem korekty z wcześniejszego kroku [25]. Dla czytelności zapisów zaniedbajmy chwilowo indeksy wag, a także oznaczmy różnicę $w(t+1)-w(t)$ jako $\Delta w(t)$. Opisany powyżej zmodyfikowany wzór na korektę dowolnej wagi ma zatem postać:

$$\Delta w(t) = -\eta\frac{\partial \frac{1}{2}e^2(t)}{\partial w(t)} + \nu\Delta w(t-1), \tag{5.38}$$

gdzie ν jest tzw. współczynnikiem momentum (ang. *momentum rate*) wybieranym z przedziału $[0,1)$. Zwykle przyjmuje się stosunkowo wysokie wartości ν, np. $\nu = 0.9$ [4, 36, 40].

Dzięki składnikowi rozpędu zmiana wagi zależy nie tylko od samego gradientu w aktualnym punkcie, ale i od zmian tej wagi z wcześniejszych kroków, jako że zgodnie z regułą (5.38) wyraz $\Delta w(t-1)$ zagnieżdża w sobie rekurencyjnie wyrazy $\Delta w(t-2)$, $\Delta w(t-3)$, itd. Nadaje to procesowi uczenia pewnej bezwładności, która

5.4 Ćwiczenia laboratoryjne (MATLAB)

E **Ćwiczenie 5.1** **Napisz skrypt realizujący algorytm uczenia perceptronu prostego dla liniowo separowalnego zbioru danych na płaszczyźnie** Wygeneruj zbiór danych w sposób sztuczny, kontrolując jego rozmiar m, a także margines odstępu pomiędzy klasami. Dane przechowaj w formie macierzy o wymiarach $m \times 4$, gdzie kolejne kolumny przechowują wartości: 1, x_{i1}, x_{i2}, y_i. Przedstaw dane w formie wykresu (polecenie `scatter`). Zaimplementuj algorytm uczący jako funkcję przyjmująca jako argumenty zbiór danych i współczynnik uczenia η. Uwaga: funkcja powinna być ogólna, tzn. pracować dla danych dowolnej wymiarowości. Jako rezultaty zwróć otrzymany wektor wag oraz liczbę wykonanych kroków aktualizacyjnych (licznik k). Sprawdź wpływ następujących zmian na licznik k: zmiana liczby przykładów (parametr m), zmiana współczynnika uczenia (parametr η), zmiana marginesu odstępu pomiędzy klasami.

E **Ćwiczenie 5.2** **Napisz skrypt realizujący algorytm uczenia perceptronu prostego z wykorzystaniem „podnoszenia wymiarowości”** Wygeneruj na płaszyźnie nieseparowalny liniwo zbiór danych określony nad prostokątem: $[0, 2\pi] \times [-1, 1]$. Punkty danych przebywające wewnątrz pętli $x_2 = \pm \sin x_1$ zalicz do klasy pozytywnej, a na zewnątrz do klasy negatywnej. Przedstaw dane w formie wykresu (polecenie `scatter`). Wprowadź parametr decydujący o żądanej wymiarowości docelowej przestrzeni cech. Podnieś wymiarowość danych zgodnie z przekształceniem Gaussa. Wykonaj uczenie perceptronem prostym, wprowadzając rozszerzony warunek stopu (maksymalna liczba kroków w sytuacji, gdy nie następuje tradycyjne zatrzymanie). Wykreśl na płaszczyźnie otrzymaną nieliniową granicę decyzyjną, korzystając z funkcji `contour` lub `contourf`. Sprawdź wpływ zadanej wymiarowości oraz parametru rządzącego szerokością dzwonów Gaussa na kształt granicy decyzyjnej.

E **Ćwiczenie 5.3** **Napisz skrypt realizujący działanie i uczenie sieci neuronowej typu perceptron wielowarstwowy (Multi-Layer Perceptron).** Polecenia do wykonania:

- Napisz skrypt realizujący działanie i uczenie sieci neuronowej. Skrypt powinien przyjmować na wejście następujące argumenty: zbiór danych, zadaną liczbę neuronów, zadaną liczbę kroków uczenia, współczynnik uczenia. Skrypt powinien zwracać na wyjściu macierz z nauczonymi wartościami wag V i wektor nauczonych wag W.
- Napisz skrypt rysujący (`surf`) wykres powierzchni sieci neuronowej reprezentowanej przez wagi V, W jako funkcji x_1, x_2. Ustal zakres osi odpowiadający zakresom funkcji aproksymowanej.
- Napisz skrypt generujący zbiór danych (zbiór próbek) pochodzących z funkcji dwóch zmiennych $y(x_1, x_2) = \cos(x_1 \cdot x_2) \cdot \cos(2 \cdot x_1)$ zdefiniowanej na dziedzinie: x_1, x_2 należącej do przedziału $[0, \pi]$. Przyjmij rozmiar zbioru danych $m = 1000$. Zbiór danych przechowuj w macierzy o wymiarach $mx3$, gdzie kolejne kolumny będą odpowiadały zmiennym x_1, x_2, y.

- Za pomocą funkcji MATLABa `scatter3` i `surf` sporządź wykresy odpowiednio zbioru próbek i funkcji aproksymowanej.
- Przeprowadź uczenie i zbierz wyniki. Sugerowane ustawienia (rzędy wielkości): liczba kroków uczenia $T \sim \{10^5, ..., 10^6\}$, liczba neuronów $N \sim \{10, 20, ..., 100\}$, współczynnik uczenia $\eta \sim \{10^{-3}, ..., 10^{-1}\}$. Początkowe wartości wylosowanych macierzy V, W powinny być bardzo małe $\sim [-10^{-3}, 10^{-3}]$ (lub jeszcze mniejszy rząd wielkości).
- Wyświetl oba wykresy powierzchni: funkcji aproksymowanej i sieci neuronowej (funkcji aproksymującej) oraz porównaj podobieństwo wizualnie np. nakładając oba wykresy na siebie.

Ćwiczenie 5.4 Przebadaj działanie sieci dla różnej liczby neuronów w warstwie ukrytej (wykorzystaj program z Ćwiczenia 5.3). Polecenia do wykonania:

- Zaczerpnij z aproksymowanej funkcji nowy zbiór uczący o rozmiarze $m = 200$, przy czym wartości y należy obciążyć pewnym losowym błędem, tj. $y = y(x_1, x_2) + \varepsilon$, gdzie $\varepsilon \sim N(0, 0.2)$ - błąd losowy o rozkładzie normalnym, średniej zero i odchyleniu standardowym 0.2. W MATLABie jest funkcja `randn()` losująca liczbę (lub macierz liczb) z rozkładu normalnego.
- Podziel losowo powyższy zbiór na część uczącą i część testową w proporcji 70:30.
- W pętli (wielokrotnie) przeprowadź proces uczenia sieci zadając coraz większą liczbę neuronów: $N = 10, 20, ..., 100$ (10 iteracji). Sieć ma być uczona tylko na zbiorze uczącym. Za każdym razem początkowe wartości wag V i W mają być wylosowane na nowo. W każdej z 10 iteracji po nauczeniu sieci należy obliczyć błąd popełniany przez nią na zbiorze uczącym i na zbiorze testowym (niewidzianym podczas uczenia) jako średnią różnicę bezwzględną pomiędzy oczekiwanymi wartościami y, a odpowiedziami sieci neuronowej. Obie wartości zapamiętaj dla każdej iteracji pętli.
- Po zakończeniu pętli narysuj wykres obu wielkości - błędów uczących i testowych dla kolejnych wartości N. Wskaż, jaka liczba neuronów jest optymalna dla danego zbioru danych, tj. przy jakiej liczbie neuronów błąd na części testowej jest najmniejszy.
- Naucz ostatecznie sieć neuronową już na całym zbiorze danych (a nie tylko na części uczącej), podając optymalną liczbę neuronów N.

Ćwiczenie 5.5 Oszacuj błąd popełniany przez sieć neuronową. Aby wyznaczyć w sposób dokładny błąd popełniany przez finalną nauczoną sieć względem aproksymowanej funkcji na całej dziedzinie, należałoby np. wyliczyć całkę z bezwzględnej różnicy obu funkcji (lub kwadratu różnicy) i podzielić wynik przez miarę dziedziny (tu: π^2). Nie będzie trzeba tego robić, natomiast trzeba oszacować ten błąd poprzez przybliżenie ww. całki sumą o odpowiednio dużej liczbie składników. Korzystając z programu napisanego w ćwiczeniu 5.3, należy pobrać z funkcji dodatkowy duży zbiór próbek (np. o rozmiarze rzędu 10^4) i na jego podstawie należy policzyć średni błąd bezwzględny pomiędzy oczekiwanymi wartościami y, a odpowiedziami dostarczanymi przez sieć neuronową dla testowych punktów (x_1, x_2).