1 Przeznaczenie i instalacja biblioteki

Biblioteka „Sieci jednokierunkowe” została stworzona jako rozszerzenie standardowych możliwości programu Matlab. Jej podstawowym zadaniem jest symulacja działania perceptronu jedno i dwuwarstwowego.

1.1 Instalacja biblioteki w programie Matlab

W celu udostępnienia biblioteki w programie Matlab należy skopiować ją do wybranego folderu (standardowe biblioteki zainstalowane są w *Matlab\toolbox*). Po uruchomieniu programu należy ustawić ścieżkę dostępu do biblioteki wybierając *File|SetPath*.

1.2 Instalacja biblioteki w programie Octave

W celu udostępnienia biblioteki w programie Octave jako kolejnej standardowo dostępnej biblioteki funkcji należy skopiować ją do folderu, w którym zainstalowany jest program (domyślnie *C:\Program Files\GNU Octave*), a następnie dodać ścieżkę dostępu do pliku konfiguracyjnego *octaverc* (plik nie ma rozszerzenia). Lokalizacja bibliotek i pliku konfiguracyjnego zależy od wersji programu. Poniżej został opisany proces instalacji dla programu Octave w wersji 2.1.50 i 2.1.42.

Octave 2.1.50

1. Należy przejść do folderu, w którym zainstalowany jest program (w przypadku domyślnej instalacji *C:\Program Files\GNU Octave 2.1.50*)
2. Folder zawierający funkcje biblioteki przekopiować do folderu *opt\octave\share\octave\2.1.50\m*
3. W tym samym folderze odszukać *startup* i znajdujący się wewnątrz plik *octaverc*
4. Otworzyć plik *octaverc* w edytorze tekstów (ze względu na sposób zapisu znaków nowego wiersza systemowy Notatnik może być niewygodny w takim przypadku można skorzystać np. z programu WordPad)
5. W ostatnim wierszu pliku *octaverc* dopisać polecenie
   ```
   LOADPATH = ['/opt/octave/share/octave/2.1.50/m/slp//', LOADPATH];
   ```
6. Po uruchomieniu Octava biblioteka będzie dostępna

Octave 2.1.42

1. Należy przejść do folderu, w którym zainstalowany jest program (w przypadku domyślnej instalacji *C:\Program Files\GNU Octave 2.1.42*)
2. Folder zawierający funkcje biblioteki przekopiować do folderu *usr\share\octave\2.1.42\m*
3. W folderze *\usr\share\octave\site\m* odszukać *startup* i znajdujący się wewnątrz plik *octaverc*
4. Otworzyć plik *octaverc* w edytorze tekstów (ze względu na sposób zapisu znaków nowego wiersza systemowy Notatnik może być niewygodny w takim przypadku można skorzystać np. z programu WordPad)

5. W ostatnim wierszu pliku *octaverc* dopisać polecenie
   `LOADPATH = ['/usr/share/octave/2.1.42/m/slp//', LOADPATH]`
6. Po uruchomieniu Octava biblioteka będzie dostępna.

Uwaga: miejsce umieszczenia biblioteki jest dowolne. Zaproponowana lokalizacja zachowuje jednak standardy stosowane w programie Octave. W przypadku umieszczenia biblioteki w innym miejscu należy odpowiednio zmodyfikować ścieżkę dostępu w poleceniu `LOADPATH`.

2 Opis funkcji implementujących działanie perceptronu jednowarstwowego

2.1 Definicja sieci

net = slp(nin, nout, func, bias)

Funkcja tworzy jednowarstwowy perceptron o zadanej liczbie wejść, wyjść, neurony sieci przyjmują określoną funkcję aktywacji.

Argumenty wejściowe:

- nin - ilość wejść sieci
- nout - ilość wyjść sieci (równa ilości neuronów)
- func - funkcja aktywacji, może ona przyjmować wartości:
  - 'linear' - funkcja liniowa
  - 'hardlim' - unipolarna funkcja skoku
  - 'hardlims' - bipolarna funkcja skoku
  - 'logistic' - funkcja $y = 1/(1+e^{-x})$
  - 'tanh' - tangens hiperboliczny
- bias - określa czy neurony sieci posiadają bias
  - 1 - neurony z biasem
  - 0 - neurony bez biasu

  domyślnie bias=1

Argumenty wyjściowe:

- net - zmienna reprezentująca utworzona sieć

  zmienna ta ma ustawione pola:
  - type = 'slp' (typ sieci)
  - nin (ilość wejść sieci)
  - nout (ilość neuronów sieci)
  - actfn (funkcja aktywacji neuronów sieci)
  - w (macierz wag neuronów sieci - inicjowana zerami)
  - b (wektor biasów - inicjowany zerami)
  - wb (macierz zawierająca wagi i biasy sieci)

Przykład 2.1.

Należy utworzyć neuron z biasem o dwóch wejściach z bipolarną funkcja skoku jako funkcją aktywacji.

```
>> n = slp(2, 1, 'hardlims')
```

2.2 Ustawianie wag i biasów

net = slpset(net, w, b)

Ustawia wagi i biasy sieci na podane przy wywołaniu funkcji.

Argumenty wejściowe:

net - zmienna reprezentująca perceptron jednowarstwowy
w - macierz której wiersze zawierają wagi kolejnych neuronów sieci
b - wektor kolumnowy którego elementy zawierają biasy kolejnych neuronów sieci
domyślnie wektor o elementach równych zero

Argumenty wyjściowe:

net - zmienna reprezentująca sieć zawierająca nowe wartości wag i biasów

Przykład 2.2.

Ustawić wagi i bias neuronu z przykładu 2.1. zgodnie z poniższym opisem:

- wagę związaną z pierwszym wejściem ustawić na (-1),
- wagę związaną z drugim wejściem ustawić na (1),
- bias ustawić na (3).

```
>> n = slpset(n, [-1 1], 3)
```

2.3 Symulacja działania sieci

y = slpfwd(net, x)

Funkcja oblicza wartości sygnałów na wyjściu perceptronu jednowarstwowego dla podanego wektora sygnałów wejściowych.

Argumenty wejściowe:

net - zmienna reprezentująca perceptron jednowarstwowy
x - kolumnowy wektor sygnałów wejściowych lub
macierz której kolumny zawierają kolejne wektory sygnałów wejściowych

Argumenty wyjściowe:

y - kolumnowy wektor sygnałów wyjściowych lub
macierz której kolumny zawierają kolejne wektory sygnałów wyjściowych

Przykład 2.3.

Obliczyć wartość sygnału wyjściowego neuronu z przykładu 2.2. po podaniu na pierwsze wejście sygnału 1 a na drugie sygnału 2.

```
>> y = slpfwd(n, [1; 2])
```

2.4 Graficzna prezentacja działania sieci

2.4.1 Prezentacja działania sieci o jednym wejściu

slpplt1(net, x, color)

Funkcja rysuje wykres przedstawiający wartości sygnałów na wyjściach neuronów sieci w zależności od wartości sygnału podawanego na wejście. Sieć musi mieć dokładnie jedno wejście.

Argumenty wejściowe:

net - zmienna reprezentująca perceptron jednowarstwowy o jednym wejściu
x - wektor wierszowy zawierający wartości sygnałów dla których wyznaczana będzie odpowiedz sieci dla potrzeb kreślonego wykresu
color - parametr opcjonalny, zawiera definicje kolorów jakimi rysowane będą wykresy, domyślnie kolejność kolorów odpowiada kolejności wybieranej przez funkcję plot,
jest to litera lub łańcuch znaków (w zależności od ilości wyjść sieci),
każdy ze znaków definiuje kolor, stosowane oznaczenia kolorów odpowiadają oznaczeniom stosowanym przez funkcję plot

Przykład 2.4.

Należy przedstawić sposób przetwarzania sygnałów wejściowych z zakresu od (-5) do (5) przez sieć jednowarstwową o jednym wejściu zbudowaną z dwóch neuronów z biasem z bipolarną funkcja skoku jako funkcją aktywacji. Waga pierwszego neuronu jest ustawiana na (2) a bias na (3), waga drugiego neuronu jest ustawiana na (-1) a bias na (1).

```
>> n = slp(1, 2, 'hardlims')
>> n = slpset(n, [2; -1], [3; 1])
>> slpplt1(n, -5:0.5:5, 'rg')
```

2.4.2 Prezentacja działania neuronu o dwóch wejściach

slpplt2(net, x1, x2, fig)

Funkcja rysuje wykres przedstawiający wartości sygnału na wyjściu neuronu o dwóch wejściach w zależności od wartości sygnałów podawanych na wejście. Neuron musi mieć dokładnie dwa wejścia.

Argumenty wejściowe:

- net - zmienna reprezentująca neuron o dwóch wejściach
- x1 - wektor wierszowy zawierający wartości sygnałów podawanych na pierwsze wejście neuronu
- x2 - wektor wierszowy zawierający wartości sygnałów podawanych na drugie wejście neuronu
  domyślnie x2 = x1
- fig - wartość liczbowa określająca rodzaj wykresu
  - 1 - wykres powierzchniowy
  - 2 - wykres poziomicowy

  domyślnie wybierany jest wykres 1

Przykład 2.5.

Należy narysować wykres powierzchniowy pokazujący sposób przetwarzania sygnałów wejściowych przez neuron z biasem o dwóch wejściach z bipolarną funkcja skoku jako funkcją aktywacji. Sygnały podawane na wejścia neuronu mieszczą się w przedziale [-5, 5]. Waga związana z pierwszym wejściem neuronu powinna wynosić (-1), waga związana z drugim wejściem powinna być równa (2) a bias (3).

```
>> n = slp(2, 1, 'hardlims')
>> n = slpset(n, [-1 2], 3)
>> slpplt2(n, -5:0.5:5, -5:0.5:5, 1)
```

2.4.3 Wykres powierzchni decyzyjnych sieci o dwóch wejściach

slppltdb(net, x1, x2, color)

Rysuje podział przestrzeni wejściowej generowany przez sieć o dwóch wejściach.

Argumenty wejściowe:

- net - zmienna reprezentująca perceptron jednowarstwowy o dwóch wejściach
- x1 - dwuelementowy wektory wierszowy zawierający dolną i górną wartość sygnału podawanego na pierwsze wejście sieci
- x2 - dwuelementowy wektory wierszowy zawierający dolną i górną wartość sygnału podawanego na drugie wejście sieci
  domyślnie x2 = x1,

color - parametr opcjonalny, zawiera definicje kolorów jakimi rysowane będą powierzchnie,
domyślnie kolejność kolorów odpowiada kolejności wybieranej przez funkcję plot,
jest to litera lub łańcuch znaków (w zależności od ilości wyjść sieci),
każdy ze znaków definiuje kolor, stosowane oznaczenia kolorów odpowiadają oznaczeniom stosowanym przez funkcję plot

Przykład 2.6.

Należy narysować podział przestrzeni wejściowej generowany przez sieć jednowarstwową zbudowaną z dwóch neuronów o dwóch wejściach z bipolarną funkcją skoku jako funkcją aktywacji. Zakresy osi przestrzeni wejściowej należy ustalić na [-5, 5]. Wagi i biasy sieci należy ustawić na podstawie poniższego opisu:

- waga związana z pierwszym wejściem pierwszego neuronu: (1),
- waga związana z drugim wejściem pierwszego neuronu: (2),
- bias pierwszego neuronu: (3)
- waga związana z pierwszym wejściem drugiego neuronu: (2)
- waga związana z drugim wejściem drugiego neuronu: (0)
- bias drugiego neuronu: (0)

```
>> n = slp(2, 2, 'hardlims')
>> n = slpset(n, [1 2; 2 0], [3; 0])
>> slppltdb(n, [-5 5])
```

2.5 Uczenie sieci

[net, se] = slptrain(net, xd, yd, opt)

Funkcja uczy sieć odpowiadać sygnałami **yd** na sygnały wejściowe **xd**, parametr **opt** decyduje o parametrach procesu uczenia.

Argumenty wejściowe:

net - zmienna reprezentująca perceptron jednowarstwowy
xd - kolumnowy wektor sygnałów wejściowych lub
macierz której kolumny zawierają kolejne wektory sygnałów wejściowych
yd - kolumnowy wektor sygnałów wyjściowych lub
macierz której kolumny zawierają kolejne wektory sygnałów wyjściowych

opt - tablica komórkowa zawierająca opcje decydujące np. o maksymalnej liczbie prezentacji wzorców, wartości współczynnika korekcji, maksymalnej możliwej wartości błędu uczenia czy pokazywaniu szczegółów przebiegu tego procesu. Parametr *opt* jest parametrem opcjonalnym i może być pominięty – w takim przypadku zostaną przyjęte domyślne wartości opcji charakteryzujących przebieg procesu uczenia. Argument *opt* należy przekazywać jako tablicę komórkową o dwóch kolumnach. Pierwsza kolumna powinna zawierać nazwę opcji w postaci łańcucha znaków, druga kolumna wartość przypisywaną do opcji. Schemat tworzenia tablicy komórkowej jest podobny do zwykłej macierzy: elementy podawane są wierszami, oddzielone od siebie przecinkami lub spacjami, wiersze oddzielane średnikami. Jedyną różnicą jest umieszczanie definicji tablicy w nawiasach klamrowych.

Dostępne opcje:

view - parametr decydujący o pokazywaniu szczegółów przebiegu procesu uczenia, dopuszczalne wartości:
- 0 - szczegóły nie są pokazywane
- 1 - w każdym kroku algorytmu pokazywane są: numer iteracji oraz wartość bezwzględna sumy błędów dla wzorców zawartych w xd (oznaczana jako: se)
- 2 - w każdym kroku algorytmu wywoływana jest funkcja użytkownika, której nazwę należy podać w opcji viewFn

domyślnie view = 1

viewFn - parametr zawierający nazwę funkcji użytkownika która jest odpowiedzialna za pokazywanie szczegółów procesu uczenia, funkcja ta powinna mieć następujący nagłówek:

```
function F(net, it, se)
```

parametry wejściowe funkcji zawierają:
- net - aktualne dane perceptronu
- it - numer aktualnej iteracji
- se - wartość bezwzględną sumy błędów dla wzorców zawartych w xd

maxit - określa maksymalna liczbę prezentacji wzorców
domyślnie maxit = 1000

c - współczynnik wykorzystywany w trakcie uczenia
domyślnie c = 0.1

eps - określa wymaganą maksymalna wartość którą musi osiągnąć suma błędów dla wzorców wykorzystywanych w trakcie uczenia
domyślnie eps = 0.001

Argumenty wyjściowe:

net - zmienna reprezentująca sieć zawierającą nowe wartości wag i biasów

se - wartość bezwzględna z sumy błędów dla prezentowanych wzorców

Przykład 2.7.

Należy przeprowadzić proces uczenia dla neuronu o dwóch wejściach z bipolarną funkcją skoku jako funkcją aktywacji. Neuron powinien punkt (1, 1) przypisywać do klasy (1) a punkt (1, -1) do klasy (-1). Po zakończeniu uczenia należy wyświetlić błąd uczenia, znalezione wagi i bias neuronu, wartości sygnałów na wyjściu neuronu po podaniu na jego wejście punktów: (1, 1) i (1, -1).

```
>> n = slp(2, 1, 'hardlims')
>> x1 = [1;  1]; y1 = -1;          % definicja punktów
>> x2 = [1; -1]; y2 =  1;
>> xd = [x1 x2]                    % definicja wzorców uczących zgodnie z wymogami funkcji slptrain
>> yd = [y1 y2]
>> [n, e] = slptrain(n, xd, yd);   % zwraca błąd uczenia i sieć ze zmodyfikowanymi
                                   % w trakcie uczenia wagami
>> e                               % wyświetla błąd uczenia
>> w = n.w                         % wyświetla wagi neuronu
>> b = n.b                         % wyświetla wartość biasu
>> y = slpfwd(n, xd)               % zwraca wartości na wyjściu neuronu dla uczonych wzorców
```

Przykład 2.8.

Należy rozwiązać problem sformułowany w przykładzie poprzednim wyświetlając w każdym kroku procesu uczenia oprócz numeru iteracji i aktualnej wartości błędu również aktualne wartości wag i biasów neuronu. Dodatkowo należy przyjąć wartość współczynnika korekcji jako c = 0.5 a maksymalną wartość sumy błędów eps = 0.01.

Do uwzględnienia powyższych parametrów uczenia niezbędne jest w tym przypadku zdefiniowanie opcji. Modyfikacja domyślnego sposobu monitorowania przebiegu procesu wymaga zdefiniowania funkcji o nagłówku: `function F(net, it, se)`. Poniżej przedstawiona została przykładowa funkcja wyświetlająca wszystkie wymagane informacje:

```
function Pokaz(net, it, se)
disp('Iteracja:')
disp(it)
disp('Blad:')
disp(se)
disp('Wagi')
disp(net.w)
disp('Bias')
disp(net.b)
disp('')
```

Po zdefiniowaniu powyższej funkcji można poprawić wywołanie funkcji uczącej **slptrain**. W tym celu można wcześniej utworzyć zmienną zawierającą wartości wszystkich ustawianych opcji, np. jako:

```
>> opt = {'view', 2;  'viewFn', 'Pokaz';  'c', 0.5;  'eps', 0.01};
```

a następnie uruchomić proces uczenia:

```
>> [n, e] = slptrain(n, xd, yd, opt);
```

3 Opis funkcji implementujących działanie perceptronu dwuwarstwowego

3.1 Definicja sieci

net = dlp(nin, nhidden, nout, fhidden, fout, bias)

Funkcja tworzy dwuwarstwowy perceptron o zadanej liczbie wejść, wyjść i określonej ilości neuronów na warstwie ukrytej. Neurony poszczególnych warstw sieci przyjmują określone funkcje aktywacji.

Argumenty wejściowe:

- nin - ilość wejść sieci
- nhidden - ilość neuronów na warstwie ukrytej
- nout - ilość wyjść sieci (równa ilości neuronów warstwy wyjściowej)
- fhidden, fout - nazwa funkcji aktywacji dla neuronów warstwy ukrytej oraz warstwy wyjściowej sieci, dozwolone są następujące nazwy dla funkcji aktywacji:
  - 'linear' - funkcja liniowa
  - 'hardlim' - unipolarna funkcja skoku
  - 'hardlims' - bipolarna funkcja skoku
  - 'logistic' - funkcja $y = 1/(1+e^{-x})$
  - 'tanh' - tangens hiperboliczny
- bias - dwuelementowy wektor zawierający flagi 0 i 1 dla kolejnych warstw sieci
  - 1 neurony z biasem
  - 0 neurony bez biasu

Argumenty wyjściowe:

- net - zmienna reprezentująca utworzona sieć
  zmienna ta ma ustawione pola:
  - type = 'dlp' (typ sieci)
  - nin (ilość wejść sieci)
  - nhidden (ilość neuronów warstwy ukrytej sieci)
  - nout (ilość neuronów warstwy wyjściowej sieci)

outfn	(funkcja aktywacji neuronów warstwy wyjściowej sieci)
hiddenfn	(funkcja aktywacji neuronów warstwy ukrytej sieci)
outfn	(funkcja aktywacji neuronów warstwy wyjściowej sieci)
w1	(macierz wag neuronów warstwy ukrytej sieci - inicjowana zerami)
b1	(wektor biasów warstwy ukrytej sieci - inicjowany zerami)
wb1	(macierz zawierająca wagi i biasy warstwy ukrytej sieci)
w2	(macierz wag neuronów warstwy wyjściowej sieci - inicjowana zerami)
b2	(wektor biasów warstwy wyjściowej sieci - inicjowany zerami)
wb2	(macierz zawierająca wagi i biasy warstwy wyjściowej sieci)

Przykład 3.1.

Należy utworzyć sieć o dwóch wejściach i jednym wyjściu z trzema neuronami na warstwie ukrytej. Wszystkie neurony sieci mają bipolarną funkcję skoku jako funkcję aktywacji.

```
>> n = dlp(2, 3, 1, 'hardlims', 'hardlims', [1, 1])
```

3.2 Ustawianie wag i biasów

net = dlpset(net, w1, b1, w2, b2)

Ustawia wagi i biasy sieci na podane przy wywołaniu funkcji.

Argumenty wejściowe:

- net - zmienna reprezentująca perceptron dwuwarstwowy
- w1 - macierz której wiersze zawierają wagi kolejnych neuronów pierwszej warstwy sieci
- b1 - wektor kolumnowy którego elementy zawierają biasy kolejnych neuronów pierwszej warstwy sieci (jeżeli neurony nie mają biasów należy przesłać wektor pusty)
- w2 - macierz której wiersze zawierają wagi kolejnych neuronów drugiej warstwy sieci
- b2 - wektor kolumnowy którego elementy zawierają biasy kolejnych neuronów drugiej warstwy sieci (jeżeli neurony nie mają biasów należy przesłać wektor pusty)

Argumenty wyjściowe:

- net - zmienna reprezentująca sieć zawierająca nowe wartości wag i biasów

Przykład 3.2.

Ustawić wagi i bias neuronu z przykładu 3.1. zgodnie z poniższym opisem:

- wszystkie wagi i bias pierwszego neuronu warstwy ukrytej ustawić na (1),
- wszystkie wagi i bias drugiego neuronu warstwy ukrytej ustawić na (2),
- wszystkie wagi i bias trzeciego neuronu warstwy ukrytej ustawić na (3),
- wszystkie wagi i bias neuronu warstwy wyjściowej ustawić na (4),

```
>> n = dlp(2, 3, 1, 'hardlims', 'hardlims', [1, 1])
>> n = dlpset(n, [1 1; 2 2; 3 3], [1; 2; 3], [4 4 4], 4)
```

3.3 Symulacja działania sieci

y = dlpfwd(net, x)

Funkcja oblicza wartości sygnałów na wyjściu perceptronu dwuwarstwowego dla podanego wektora sygnałów wejściowych.

Argumenty wejściowe:

net - zmienna reprezentująca perceptron jednowarstwowy
x - kolumnowy wektor sygnałów wejściowych lub
macierz której kolumny zawierają kolejne wektory sygnałów wejściowych

Argumenty wyjściowe:

y - kolumnowy wektor sygnałów wyjściowych lub
macierz której kolumny zawierają kolejne wektory sygnałów wyjściowych

Przykład 3.3.

Obliczyć wartość sygnału wyjściowego neuronu z przykładu 3.2. po podaniu na pierwsze wejście sygnału 1 a na drugie sygnału 2.

```
>> y = dlpfwd(n, [1; 2])
```

3.4 Graficzna prezentacja działania sieci

3.4.1 Prezentacja działania sieci o jednym wejściu

dlpplt1(net, x, color)

Funkcja rysuje wykres przedstawiający wartości sygnałów na wyjściach neuronów sieci w zależności od wartości sygnału podawanego na wejście. Sieć musi mieć dokładnie jedno wejście.

Argumenty wejściowe:

net - zmienna reprezentująca perceptron dwuwarstwowy o jednym wejściu
x - wektor wierszowy zawierający wartości sygnałów dla których wyznaczana będzie odpowiedz sieci dla potrzeb kreślonego wykresu
color - parametr opcjonalny, zawiera definicje kolorów jakimi rysowane będą wykresy, domyślnie kolejność kolorów odpowiada kolejności wybieranej przez funkcję plot,
jest to litera lub łańcuch znaków (w zależności od ilości wyjść sieci),
każdy ze znaków definiuje kolor, stosowane oznaczenia kolorów odpowiadają oznaczeniom stosowanym przez funkcję plot

Przykład 3.4.

Należy przedstawić sposób przetwarzania sygnałów wejściowych z zakresu od (-5) do (5) przez sieć dwuwarstwową zbudowaną z neuronów z biasem z bipolarną funkcja skoku jako funkcją aktywacji. Sieć ma jedno wejście, dwa neurony na warstwie ukrytej i jedno wyjście. Waga i biasy neuronów są równe:

- pierwszy neuron warstwy ukrytej: waga (1), bias (-2)
- drugi neuron warstwy ukrytej: waga (-1), bias (0)
- neuron warstwy wyjściowej: waga związana z pierwszym wejściem (1), waga związana z drugim wejściem (2), bias (2)

```
>> n = dlp(1, 2, 1, 'hardlims', 'hardlims', [1, 1])
>> n = dlpset(n, [1; -1], [-2; 0], [1 2], 2)
>> dlpplt1(n, -5:0.5:5, 'r')
```

3.4.2 Prezentacja działania sieci o dwóch wejściach i jednym wyjściu

dlpplt2(net, x1, x2, fig)

Funkcja rysuje wykres przedstawiający wartości sygnału na wyjściu sieci dwuwarstwowej o dwóch wejściach w zależności od wartości sygnałów podawanych na wejście. Sieć musi mieć dokładnie dwa wejścia i jedno wyjście.

Argumenty wejściowe:

net - zmienna reprezentująca perceptron dwuwarstwowy o dwóch wejściach i jednym wyjściu
x1 - wektor wierszowy zawierający wartości sygnałów podawanych na pierwsze wejście sieci
x2 - wektor wierszowy zawierający wartości sygnałów podawanych na drugie wejście sieci
domyślnie x2 = x1
fig - wartość liczbowa określająca rodzaj wykresu
1 - wykres powierzchniowy
2 - wykres poziomicowy
domyślnie wybierany jest wykres 1

Przykład 3.5.

Należy narysować wykres powierzchniowy pokazujący sposób przetwarzania sygnałów wejściowych przez sieć dwuwarstwową zbudowaną z neuronów z biasem z bipolarną funkcja skoku jako funkcją aktywacji. Sieć ma dwa wejścia, dwa neurony na warstwie ukrytej i jedno wyjście. Waga i biasy neuronów są równe:

- pierwszy neuron warstwy ukrytej: wagi równe (1), bias (-2)
- drugi neuron warstwy ukrytej: wagi równe (-1), bias (0)
- neuron warstwy wyjściowej: waga związana z pierwszym wejściem (1), waga związana z drugim wejściem (2), bias (2)

```
>> n = dlp(2, 2, 1, 'hardlims', 'hardlims', [1, 1])
>> n = dlpset(n, [1 1; -1 -1], [-2; 0], [1 2], 2)
>> dlpplt2(n, -5:0.5:5)
```

3.5 Uczenie sieci

[net, se] = dlptrain(net, xd, yd, opt)

Funkcja uczy sieć odpowiadać sygnałami **yd** na sygnały wejściowe **xd**, parametr **opt** decyduje o parametrach procesu uczenia. Składnia funkcji i znaczenie poszczególnych argumentów jest analogiczne do funkcji **slptrain** (patrz punkt 2.5). Jedyna różnica w jej wykorzystaniu polega na ustalaniu wartości wag początkowych – jeżeli nie zostaną one przypisane, funkcja generuje wagi o wartościach losowych.

Uwaga 1. Funkcja działa tylko dla sieci z neuronami o ciągłych funkcjach aktywacji, funkcje ‘hardlims’ i ‘hardlim’ są więc niedozwolone.

Uwaga 2. Wynik uczenia jest mocno zależny od wartości wag początkowych. Często uczenie kończy się niepowodzeniem również dla zadań, które mają rozwiązania. W takim przypadku należy kilkakrotnie uruchamiać proces uczenia z różnymi wagami początkowymi.