2. DEFINIOWANIE ARCHITEKTURY SIECI – SZYBKI START

Biblioteka *Neural Network Toolbox* udostępnia wiele różnych metod tworzenia sieci neuronowych. Można na przykład korzystać z polecenia **network** umożliwiającego wygenerowanie sieci o dowolnej strukturze. Polecenie to tworzy zmienną, która reprezentuje sieć o niezdefiniowanej jeszcze architekturze. Architekturę tą można zdefiniować odwołując się do odpowiednich elementów składowych zmiennej. Metoda ta jest dosyć pracochłonna – wymaga ustalania wielu parametrów zmiennej sieciowej. Należy na przykład określić liczbę warstw sieci, sposób ich połączenia, ilości neuronów, funkcje aktywacji itp.

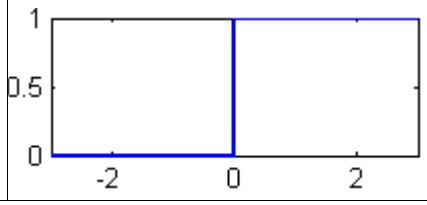
Dostępne są również specjalizowane funkcje do tworzenia typowych sieci:

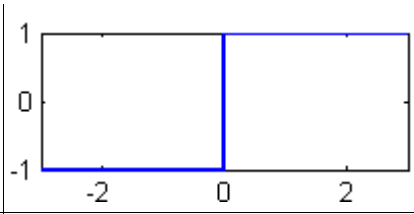
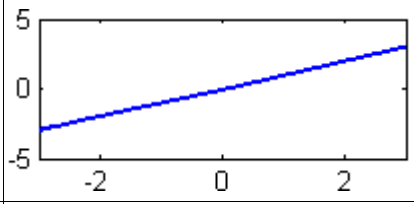
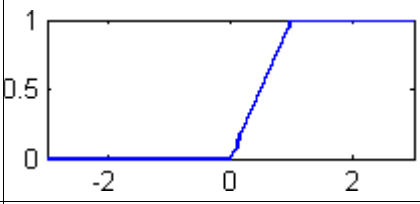
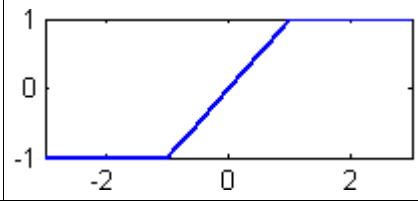
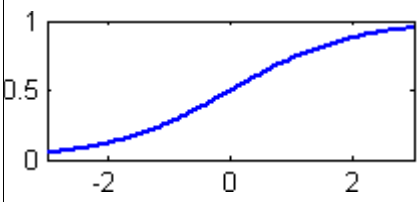
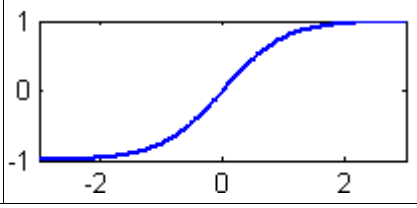
Funkcja	***Definiowana sieć***
newp	perceptron jednowarstwowy
newff	jednokierunkowa sieć wielowarstwowa
newlin	jednowarstwowa sieć zbudowana z neuronów liniowych
newrb	Sieć radialna RBF ang. Radial Basis Functions
newc	Sieć samoorganizująca się na zasadzie współzawodnictwa
newelm	Sieć rekurencyjną Elmana
newhop	Sieć rekurencyjna Hopfielda

Każda z wymienionych w powyższej tabeli funkcji wymaga określenia niewielkiej ilości niezbędnych parametrów. Wynikiem działania tych funkcji jest zmienna sieciowa o dokładnie zdefiniowanej architekturze. W każdym przypadku użytkownik może jednak zmieniać parametry tak utworzonej zmiennej.

Poniżej krótko omówiony zostanie sposób definiowania zmiennej reprezentującej sieć – przy pomocy poleceń: **newp**, **newlin** i **newff**. Wcześniej jednak przedstawione zostaną funkcje wykorzystywane jako funkcje aktywacji neuronu.

2.1. Funkcje aktywacji neuronów

Funkcja	**Wykres**	**Funkcja Matlab-a**
$\mathbf{a} = \begin{cases} 1, & \varphi \geq 0 \\ 0, & \varphi < 0 \end{cases}$		**hardlim**

$\mathbf{a} = \begin{cases} 1, & \varphi \geq 0 \\ -1, & \varphi < 0 \end{cases}$		**hardlims**
$\mathbf{a} = \varphi$		**purelin**
$\mathbf{a} = \begin{cases} 1, & \varphi > 1 \\ \varphi, & 0 \leq \varphi \leq 1 \\ 0, & \varphi < 0 \end{cases}$		**satlin**
$\mathbf{a} = \begin{cases} 1, & \varphi > 1 \\ \varphi, & -1 \leq \varphi \leq 1 \\ -1, & \varphi < -1 \end{cases}$		**satlins**
$\mathbf{a} = \frac{1}{1+e^{-\varphi}}$		**logsig**
$\mathbf{a} = \tanh(\varphi)$		**tansig**

2.2. Perceptron jednowarstwowy – newp

Perceptron jednowarstwowy jest siecią złożoną z jednej warstwy neuronów z bipolarną lub unipolarną funkcją skoku jako funkcją aktywacji. Pełna definicja architektury sieci tego typu musi zawierać:

- ilość wejść sieci,
- ilość neuronów,
- typ funkcji aktywacji.

Funkcja **newp** wykorzystywana do definiowania sieci tego typu wymaga określania ilości wejść sieci nie w postaci liczby lecz w postaci macierzy. Macierz ta powinna mieć dwie kolumny i tyle wierszy ile sieć ma wejść. Każdy wiersz macierzy definiuje właściwości kolejnych wejść sieci. Na definicję wejścia sieci składa się informacja o minimalnej i maksymalnej wartości sygnału, który może być podany na to wejście – informacje te wykorzystywane są podczas uczenia sieci. W pierwszej kolumnie macierzy

należy umieścić wartość odpowiadającą minimalnej wartości sygnału wejściowego a w drugiej kolumnie wartość odpowiadającą maksymalnej wartości tego sygnału. Ilość wejść sieci musi być zdefiniowana w pierwszym parametrze podczas wywołania funkcji **newp**.

Ilość neuronów sieci jest drugim parametrem polecenia **newp**. Ilość tą definiuje się podając liczbę całkowitą odpowiadającą liczbie neuronów sieci.

Trzeci parametr funkcji **newp** definiuje funkcję aktywacji neuronów sieci. W przypadku neuronów z bipolarną funkcją skoku jako funkcją aktywacji parametr ten powinien mieć wartość '*hardlims'*, a dla neuronów z unipolarną funkcją skoku: '*hardlim'*.

Parametry drugi i trzeci mogą być pomijane podczas wywoływania funkcji **newp** – powoduje to przyjęcie odpowiednich wartości domyślnych. Domyślną liczbą neuronów sieci jest jeden, a domyślną funkcją aktywacji jest '*hardlim'*.

Przykład 1.

Należy utworzyć zmienną o nazwie **net** reprezentującą perceptron o trzech wejściach z dwoma neuronami na warstwie. Sygnały podawane na wejścia sieci osiągają wartości z przedziału [-1 1]. Funkcją aktywacji neuronów sieci jest bipolarna funkcja skoku.

```
>> net = newp([-1 1; -1 1; -1 1], 2, 'hardlims');
```

2.3. Jednowarstwowa sieć zbudowana z neuronów liniowych – newlin

Sieć tego typu złożona jest z jednej warstwy neuronów z liniową funkcją aktywacji. Pełna definicja architektury sieci tego typu musi zawierać:

- ilość wejść sieci,
- ilość neuronów.

Parametry te definiuje się w taki sam sposób jak w przypadku funkcji **newp**.

Przykład 2.

Należy utworzyć zmienną o nazwie **net** reprezentującą sieć o dwóch wejściach z dwoma liniowymi neuronami na warstwie. Sygnały podawane na pierwsze wejście sieci osiągają wartości z przedziału [0 10] a na drugie wejście sieci z przedziału [-5 5].

```
>> net = newlin([0 10; -5 5], 2);
```

2.4. Wielowarstwowa sieć jednokierunkowa – newff

Funkcja **newff** tworzy sieć zbudowaną z wielu warstw, w której sygnały przesyłane są w jednym kierunku od wejścia do wyjścia. Definicja architektury sieci tego typu musi zawierać:

- ilość wejść sieci,
- ilość neuronów na wszystkich warstwach sieci,
- typy funkcji aktywacji neuronów.

Pierwszy parametr funkcji **newff** określa ilość wejść sieci i musi być zdefiniowany w taki sam sposób jak w przypadku poleceń **newp** i **newlin**.

W drugim parametrze funkcji należy określić ilości neuronów na wszystkich warstwach sieci. Parametr ten jest wektorem wierszowym, którego kolejne elementy definiują ilości neuronów na kolejnych warstwach sieci.

Neurony znajdujące się na tej samej warstwie sieci wielowarstwowej muszą mieć takie same funkcje aktywacji. Trzeci parametr polecenia **newff** definiuje funkcje aktywacji neuronów kolejnych warstw sieci. Parametr ten musi być *tablicą blokową* o jednym wierszu i tylu kolumnach ile sieć ma warstw. W kolejnych elementach tablicy należy umieścić teksty odpowiadające nazwom funkcji aktywacji neuronów leżących na kolejnych warstwach sieci. Parametr ten może zostać pominięty podczas wywołania – funkcją aktywacji neuronów wszystkich warstw sieci będzie w takim przypadku funkcja *'tansig'*.

Tablica blokowa – (ang. cell array), elementami tablicy blokowej są bloki, w których mogą być przechowywane wartości różnych typów: teksty, macierze. Sposób tworzenia zmiennych tego typu jest podobny do tworzenia zwykłych macierzy, jedyną różnicą jest konieczność użycia nawiasów klamrowych zamiast nawiasów kwadratowych.

Przykład:
Należy zdefiniować tablicę blokową **X** o wymiarach 2x2. Elementy tablicy powinny otrzymać wartość zgodnie z poniższym opisem:

element o indeksie (1,1): liczba 5
element o indeksie (1,2): wektor wierszowy zawierający kolejne liczby całkowite od 1 do 10
element o indeksie (2,1): macierz o wymiarach 2x2 zawierająca zera,
element o indeksie (2,2): macierz jednostkowa o wymiarach 3x3.

Zmienną można utworzyć wykonując polecenie:
>> **X = {5 [1:10] ; zeros(2) eye(3)}**

Odwołanie do określonego elementu tablicy blokowej wygląda podobnie jak do elementu macierzy, jedyną różnicą jest konieczność użycia nawiasów klamrowych zamiast nawiasów okrągłych. Aby odwołać się do elementu w pierwszym wierszu i drugiej kolumnie należy wykonać polecenie:
>> **X{1,2} % wyświetlony zostanie wektor: 1 2 .. 10**

Przykład 3.

Należy utworzyć zmienną o nazwie **net** reprezentującą dwuwarstwową sieć o dwóch wejściach. Na pierwszej warstwie sieci powinny znajdować się trzy neurony, na drugiej warstwie jeden neuron. Funkcją aktywacji neuronów sieci powinna być bipolarna funkcja sigmoidalna. Sygnały podawane na wejścia sieci osiągają wartości z przedziału [-1 1].

```
>> net = newff([-1 1; -1 1], [3 1], {'tansig', 'tansig'});
```

2.5. Składowe zmiennej reprezentującej sieć neuronową

W poprzednich punktach omówione zostały metody definiowania zmiennej reprezentującej sieć neuronową. Biblioteka *Neural Network Toolbox* zawiera funkcje, które umożliwiają przeprowadzenie procesu uczenia oraz symulację działania sieci reprezentowanej przez taką zmienną. Funkcje te zostaną omówione w dalszej części materiału, w punkcie bieżącym przedstawione zostaną natomiast sposoby modyfikacji pewnych parametrów zmiennej sieciowej (założono, że zmienną tą jest zmienna o nazwie **net**).

2.5.1. Funkcje aktywacji neuronów sieci

Dostęp do funkcji aktywacji neuronów *i*-tej warstwy jest możliwy poprzez odwołanie **net.layers{i}.transferFcn**.

Przykład 4.

Należy wyświetlić nazwy funkcji aktywacji neuronów sieci z przykładu 3.

```
>> net.layers{1}.transferFcn
ans =
    'tansig'
>> net.layers{2}.transferFcn
ans =
    'tansig'
```

nazwa funkcji aktywacji neuronów pierwszej warstwy

nazwa funkcji aktywacji neuronów drugiej warstwy

Przykład 5.

Należy zmienić funkcje aktywacji neuronów sieci z przykładu 1. Funkcją aktywacji neuronów sieci powinna być unipolarna funkcja skoku.

```
>> net.layers{1}.transferFcn = 'hardlim';
```

Uwaga! W przypadku wykorzystywania zmiennej sieciowej wyłącznie do symulacji działania sieci neuronowej, neurony sieci mogą mieć dowolne funkcje aktywacji. Uczenie sieci wprowadza jednak pewne ograniczenia, na przykład:

- wykorzystywana do uczenia jednowarstwowego perceptronu *'reguła perceptronu'* wymaga aby funkcje aktywacji neuronów sieci były funkcjami skokowymi,
- *'algorytm propagacji wstecznej'* wykorzystywany do uczenia wielowarstwowych sieci jednokierunkowych wymaga natomiast aby funkcje aktywacji neuronów sieci były funkcjami ciągłymi – funkcje skokowe są więc niedozwolone.

2.5.2. Wagi i biasy

Zmienna reprezentująca sieć neuronową nie zawiera wyłącznie informacji dotyczących architektury sieci, przechowuje również na przykład aktualne wartości wag i bias-ów.

Wagi każdej warstwy neuronów są przechowywane w postaci macierzy, której liczba wierszy odpowiada liczbie neuronów a liczba kolumn liczbie wejść danej warstwy sieci (wagi związane z *i*-tym neuronem stanowią *i*-ty wiersz tej macierzy). Dostęp do wag neuronów sieci zależy od warstwy na której leżą neurony. W poniższej tabelce zestawione zostały odwołania do macierzy wag kolejnych czterech warstw sieci jednokierunkowej:

Dostęp do macierzy wag sieci jednokierunkowej			
Warstwa 1	**Warstwa 2**	**Warstwa 3**	**Warstwa 4**
>> **net.IW{1,1}**	>> **net.LW{2,1}**	>> **net.LW{3,2}**	>> **net.LW{4,3}**

Dostęp do bias-ów neuronów *i*-tej warstwy jest możliwy poprzez odwołanie **net.b{i}**. Bias-y neuronów każdej warstwy sieci są przechowywane w postaci wektorów kolumnowych, których liczba elementów odpowiada ilości neuronów na warstwie.

Przykład 6.

Należy wyświetlić aktualne wartości wag i bias-ów sieci reprezentowanej przez zmienną z przykładu 3.

>> **net.IW{1,1}**

macierz wag neuronów pierwszej warstwy, dla przykładowej zmiennej sieciowej macierz ta ma wymiar 3x2 (3 wiersze ponieważ na pierwszej warstwie są 3 neurony, 2 kolumny ponieważ sieć ma 2 wejścia)

>> **net.b{1}**

bias-y neuronów pierwszej warstwy, dla przykładowej zmiennej sieciowej jest to wektor kolumnowy o 3 elementach (ponieważ na pierwszej warstwie są 3 neurony)

>> **net.LW{2,1}**

macierz wag neuronów drugiej warstwy, dla przykładowej zmiennej sieciowej macierz ta ma wymiar 1x3 (1 wiersz ponieważ na drugiej warstwie jest 1 neuron, 3 kolumny ponieważ wejściami tej warstwy są 3 wyjścia warstwy poprzedniej)

>> **net.b{2}** — *bias-y neuronów drugiej warstwy, dla przykładowej zmiennej sieciowej jest to liczba (ponieważ na drugiej warstwie jest 1 neuron)*

Przykład 7.

Wagi i bias-y neuronów sieci z przykładu 3 powinny wynosić 1.

```
>> net.IW{1,1} = ones(3,2)
>> net.b{1} = ones(3,1)
>> net.LW{2,1} = ones(1,3)
>> net.b{2} = 1
```

3. SYMULACJA DZIAŁANIA SIECI

W każdej chwili można sprawdzić w jaki sposób sieć przetwarza sygnały wejściowe w wyjściowe. Do symulacji działania sieci należy wykorzystać funkcję **sim**.

Pierwszym parametrem funkcji **sim** musi być zmienna reprezentująca sieć neuronową – sposób definiowania takiej zmiennej został omówiony w punkcie pierwszym.

W drugim parametrze funkcji należy podać wartości sygnałów podawanych na wejścia sieci dla których wyznaczana będzie następnie wartość odpowiedzi na wyjściach. Sygnały wejściowe muszą być podawane w postaci wektorów kolumnowych o rozmiarze równym ilości wejść sieci.

Funkcja **sim** wyznacza wartości sygnałów wyjściowych i zwraca odpowiedź w postaci wektora kolumnowego, którego kolejne elementy odpowiadają kolejnym wyjściom sieci.

Uwaga! Istnieje możliwość wykonania symulacji dla wielu zbiorów sygnałów wejściowych jednocześnie. W takim przypadku drugi parametr funkcji **sim** musi być macierzą, której kolejne kolumny muszą odpowiadać kolejnym zbiorom sygnałów wejściowych. Odpowiedź zwracana przez funkcję będzie również macierzą – *i*-ta kolumna tej macierzy zawierać będzie wartości na wyjściach sieci dla *i*-tego zbioru sygnałów wejściowych.

Przykład 8.

Należy utworzyć zmienną o nazwie **net** reprezentującą perceptron o trzech wejściach z dwoma neuronami na warstwie. Sygnały podawane na wejścia sieci osiągają wartości z przedziału [1 10]. Funkcją aktywacji neuronów sieci jest bipolarna funkcja skoku. Wagi pierwszego neuronu powinny wynosić: w_1=1, w_2=2, w_3=1, bias b=0, a wagi drugiego neuronu: w_1=-2, w_2=0, w_3=1, bias b=2. Należy wyznaczyć wartości na wyjściach sieci:

- dla sygnałów wejściowych równych 1,
- jednocześnie dla sygnałów równych 1, sygnałów równych 2 i sygnałów równych 3.

```
>> net = newp([1 10; 1 10; 1 10], 2, 'hardlims');
>> net.IW{1,1} = [1 2 1; -2 0 1]
>> net.b{1} = [0; 2]
```

definicja zmiennej

ustalenie wartości wag i bias-ów

Aby wyznaczyć odpowiedź sieci dla sygnałów wejściowych równych 1 należy wykonać polecenie:

```
>> sim(net, [1;1;1])
ans =
     1
     1
```

Odpowiedź sieci dla trzech zbiorów sygnałów wejściowych zostanie wyznaczona w wyniku wykonania polecenia:

```
>> sim(net, [1 2 3;1 2 3;1 2 3])
ans =
     1     1     1
     1     1     1
```

sygnały wejściowe równe 3

sygnały wejściowe równe 2

sygnały wejściowe równe 1