Jednokierunkowe sieci wielowarstwowe

Rodzaje funkcji aktywacji

purelin – funkcja liniowa
logsig – funkcja sigmoidalna
tansig – tangens hiperboliczny

Polecenie tworzące sieć bez podziału danych na część uczącą i testującą:

```
net = newff(PR,[S1 ... Sn],{F1 ... Fn})
```

PR – macierz określająca minimum i maksimum dla wejść sieci
S1 ... Sn – ilość neuronów na kolejnych warstwach sieci (ostatnia wartość musi być taka jak ilość wyjść)
TF1 ... TFn – funkcje aktywacji na warstwach sieci (możliwe to: `purelin`, `logsig`, `tansig`)

Przykładowo:

```
net = newff(minmax(we), [10 6], {'logsig', 'purelin'});
```

utworzy sieć dwuwarstwową, która ma 10 neuronów na warstwie ukrytej i 6 na wyjściowej. Na warstwie ukrytej neurony będą miały sigmoidalną funkcję aktywacji, a na wyjściowej – funkcję liniową.

```
net = newff(minmax(we), [8 5 1], {'tansig', 'tansig', 'logsig'});
```

utworzy sieć trójwarstwową, która ma 8 neuronów na I warstwie ukrytej, 5 na II warstwie ukrytej i 1 na wyjściowej. Na warstwach ukrytych neurony będą miały funkcję aktywacji typu tangens hiperboliczny, a na wyjściowej – funkcję sigmoidalną.

Polecenie tworzące sieć z podziałem danych na część uczącą i testującą:

```
net = newff(we,wy,[h1 h2],{F1 ... Fn});
```

we – macierz danych wejściowych
wy – wektor (lub macierz) zadanych danych wyjściowych
[h1 h2] – wektor określający ilość neuronów na kolejnych warstwach ukrytych
`{F1 ... Fn}`– wektor określający typy funkcji aktywacji na warstwach ukrytych i wyjściowej

Przykłady:

Sieć z 1 warstwą ukrytą na której jest 5 neuronów z funkcją aktywacji typu tangens hiperboliczny i z neuronami liniowymi na warstwie wyjściowej:

```
net = newff(we,wy,[5],{'tansig', 'purelin'});
```

Sieć z dwoma warstwami ukrytymi. Na pierwszej mamy 10 neuronów z funkcją aktywacji typu tangens hiperboliczny, na drugiej 4 neurony z funkcją sigmoidalną i na warstwie wyjściowej mamy neurony liniowe:

```
net = newff(we,wy,[10 4],{'tansig', 'logsig','purelin'});
```

Przykład dla danych 1-wejściowych

```
we = load('dane_1D_sin1_i.txt'); we = we';
wy = load('dane_1D_sin1_o.txt'); wy = wy';

net = newff(minmax(we),[4 1],{'logsig', 'purelin'}); % bez podzialu danych
%net = newff(we,wy,[4],{'logsig', 'purelin'});       % z podzialem danych
net.inputweights{1,1}.initFcn = 'rands';
net.biases{1}.initFcn = 'rands';
net = init(net); % inicjujemy wagi losowo
net.trainParam.epochs = 100;

figure(1)
we1 = linspace(min(we),max(we),1000);
wy1 = sim(net, we1);
plot(we1,wy1,'r'); hold on;
plot(we,wy,'.b')
title('Charakterystyka modelu przed uczeniem')

net = train(net, we, wy);

figure(2)
wy2 = sim(net, we1);
plot(we1,wy2,'g'); hold on;
plot(we,wy,'.b')
title('Charakterystyka modelu po uczeniu')
```

Przykład dla danych 2-wejściowych:

```
we = load('dane_2D_5_i.txt'); we = we';
wy = load('dane_2D_5_o.txt'); wy = wy';

net = newff(minmax(we),[5 1],{'tansig', 'logsig'}); % bez podzialu danych
%net = newff(we,wy,[5],{'tansig', 'logsig'});       % z podzialem danych
%net.outputs{2}.processFcns = {};
net.inputweights{1,1}.initFcn = 'rands';
net.biases{1}.initFcn = 'rands';
net = init(net); % inicjujemy wagi losowo
net.trainParam.epochs = 100;

figure(1)
wy1 = sim(net, we);
x_lin = linspace(min(we(1,:)),max(we(1,:)),50);
y_lin = linspace(min(we(2,:)),max(we(2,:)),50);
[X,Y] = meshgrid(x_lin,y_lin);
Z = griddata(we(1,:), we(2,:), wy1, X, Y, 'cubic');
mesh(X,Y,Z)
hold on
plot3(we(1,:), we(2,:), wy,'.');
title('Powierzchnia modelu przed uczeniem')

net = train(net, we, wy);

figure(2)
wy2 = sim(net, we);
x_lin = linspace(min(we(1,:)),max(we(1,:)),50);
y_lin = linspace(min(we(2,:)),max(we(2,:)),50);
[X,Y] = meshgrid(x_lin,y_lin);
Z = griddata(we(1,:), we(2,:), wy2, X, Y, 'cubic');
mesh(X,Y,Z)
hold on
plot3(we(1,:), we(2,:), wy,'.');
title('Powierzchnia modelu po uczeniu')
```

```
figure(3) % wykres tylko dla zadania klasyfikacji binarnej
%plot(we(1,:), we(2,:),'.'); hold on
plotpv(we,wy); hold on
contour(X,Y,Z,[0.5])
title('Linia separujaca klasy')

figure(4)
x_lin = linspace(min(we(1,:)),max(we(1,:)),50);
y_lin = linspace(min(we(2,:)),max(we(2,:)),50);
[X,Y] = meshgrid(x_lin,y_lin);
Z = griddata(we(1,:), we(2,:), wy, X, Y, 'cubic');
mesh(X,Y,Z)
hold on
plot3(we(1,:), we(2,:), wy,'.');
title('Interpolowana powierzchnia danych')
```

Przykład dla danych z większą ilością wejść niż 2:

```
we = load('dane_8D_diabet_i.txt'); we = we';
wy = load('dane_8D_diabet_o.txt'); wy = wy';

net = newff(minmax(we),[10 1],{'tansig', 'logsig'}); % bez podzialu danych
%net = newff(we,wy,[10],{'tansig', 'logsig'});       % z podzialem danych
%net.outputs{2}.processFcns = {};
net.inputweights{1,1}.initFcn = 'rands';
net.biases{1}.initFcn = 'rands';
net = init(net); % inicjujemy wagi losowo
net.trainParam.epochs = 100;

% wykresy dla wybranego wyjscia:
nr_wyjscia = 1;
figure(1)
y = sim(net, we);
plot(abs(y(nr_wyjscia,:)-wy(nr_wyjscia,:)),'r'); hold on;

net = train(net, we, wy);

y1 = sim(net, we);
plot(abs(y1(nr_wyjscia,:)-wy(nr_wyjscia,:)),'g'); hold off;
title('Blad dla kolejnych probek przed (czerwony) i po (zielony) uczeniu');

figure(3)
plot(1:length(we),wy(nr_wyjscia,:),'.b', 1:length(we),y(nr_wyjscia,:), ...
'.r', 1:length(we),y1(nr_wyjscia,:),'.g');
title('Wyjscie dla kolejnych probek: zadane (niebieski), przed uczeniem
(czerwony) , po uczeniu (zielony)')
```

Zadania do wykonania:

1. Przygotować dane i dobrać najprostszą sieć realizującą działanie bramki logicznej XOR.
2. Dla wybranych plików z danymi:
   a. zapoznać się z danymi: (ile jest wejść i wyjść? jakiego typu są wejścia i wyjścia? jakie są ich zakresy? czy mamy do czynienia z zadaniem klasyfikacji czy aproksymacji?),
   b. dobrać strukturę sieci neuronowej (ilość wejść i wyjść zależy od danych, dobrać ilość warstw ukrytych – 1 lub 2, dobrać ilość neuronów w warstwach ukrytych – np. metodą addytywną, dobrać typy funkcji aktywacji w warstwach ukrytych i w warstwie wyjściowej – zależnie od problemu),
   c. dla badanych struktur sieci przeprowadzić uczenie, wyznaczyć błąd dla danych uczących i obserwować kształt charakterystyki modelu.
3. Opracować samodzielnie skrypt, który:

a. podzieli dane na część uczącą i testującą w zadanej proporcji,
b. dobierze optymalną ilość neuronów na warstwie ukrytej. W trakcie uczenia dane uczące powinny być wykorzystane do modyfikacji wag, a dane testujące do określenia jakości działania sieci.

W sprawozdaniu zamieszczamy raport z przeprowadzonych eksperymentów dla każdych badanych danych, najlepiej w formie tabeli (należy zaznaczyć, która struktura sieci okazała się najlepsza):

Struktura sieci	Rodzaje funkcji aktywacji	Błąd danych uczących

Zamieścić należy także skrypty opracowane w czasie eksperymentów.

Opis plików z danymi:

Plik(i)	Ilość próbek	Ilość wejść	Ilość wyjść	Opis
dane_1D_4 dane_1D_6 dane_1D_7 dane_1D_9		1	1	
dane_1D_sin1 dane_1D_sin2 dane_1D_sin3		1	1	Sinusoida próbkowana z różną gęstością
dane_1D_sin1a dane_1D_sin2a dane_1D_sin3a		1	1	Zaszumiona sinusoida próbkowana z różną gęstością
dane_2D_1 dane_2D_2 dane_2D_3 dane_2D_4 dane_2D_5 dane_2D_a		2	1	
dane_2D_percep		2	1	
dane_2D_elusage		2	1	Średnie miesięczne zużycie prądu w zależności od miesiąca i średniej temperatury miesięcznej
dane_3D_kapitan	269	3	1	Wyjście to ocena stopnia niebezpieczeństwa dla statku dokonana przez eksperta-kapitana (0, 0.5, 1) Wejścia to prędkości statku
dane_5D_parity	32	5	1	Wyjście = 1 gdy suma wejść jest parzysta, 0 gdy nie
dane_5D_build	4208	5	3	Wejścia to: godzina, temperatura zewnętrzna, wilgotność, nasłonecznienie, siła wiatru Wyjścia to: zużycie zimnej wody, ciepłej wody i prądu w budynku
dane_6D_transak	128	6	1	Wejścia to: kwota, indeks firmy, godzina, typ osoby autoryzującej transakcję, dzień tygodnia, typ dnia (czy wolny czy nie) Wyjście to: wiarygodność transakcji w %
dane_8D_diabet	768	8	1	Wejścia: dane o pacjentach Wyjście: ryzyko zachorowania na cukrzycę
dane_9D_glass	214	9	6	Wejścia to parametry fizyko-chemiczne szkła Wyjścia (0,1) określają klasę do której zaliczamy szkło
dane_35D_heart	920	35	1	Wejścia: znormalizowane dane o pacjentach Wyjście: ryzyko zachorowania na serce