Patryk Kurzeja

Inżynieria Obliczeniowa

Nr albumu : 286112

Podstawy sztucznej inteligencji

Sprawozdanie : Budowa i działanie sieci wielowarstwowej [Scenariusz 3] :

1. Cel ćwiczenia :

Celem ćwiczenia jest poznanie budowy i działania wielowarstwowych sieci neuronowych poprzez uczenie z użyciem algorytmu wstecznej propagacji błędu rozpoznawania konkretnych liter alfabetu.

2. Pojęcia i algorytmy :

- **Sieci neuronowe** - są sztucznymi strukturami, których budowa i działanie zostały zaprojektowane w sposób modelujący działanie naturalnego układu nerwowego, w szczególności mózgu. Cieszą się bardzo dużym zainteresowaniem. Jednym z głównych czynników mających na to wpływ jest możliwość stosowania ich w bardzo wielu dziedzinach życia do rozwiązywania problemów, gdzie użycie innych metod jest trudne lub niemożliwe. Znakomicie sprawdzają się w problemach klasyfikacji, predykcji, związanych ze sterowaniem, analizą danych.

- **Funkcja newff -** Funkcja tworzy wielowarstwową sieć neuronową; każda warstwa składa się z zadanej liczby neuronów o nieliniowych funkcjach aktywacji (jakkolwiek funkcje aktywacji w poszczególnych warstwach mogą mieć również postać liniową).

Wywołanie funkcji: **NET = NEWFF(PR, [S1 S2...SNL], ...**

{TF1 TF2...TFNL}, BTF, BLF, PF)

WEJŚCIE:

PR - macierz o wymiarach Rx2, gdzie R jest liczbą wejść sieci (współrzędnych wektorów wejściowych); pierwsza kolumna zawiera minimalne wartości kolejnych współrzędnych wektorów wejściowych, druga kolumna - maksymalne wartości tych współrzędnych

Si - liczba neuronów w i-tej warstwie sieci; liczba warstw wynosi N1

TFi - nazwa funkcji aktywacji neuronów w i-tej warstwie sieci (zmienna tekstowa); domyślna = 'tansig' (tangens hiperboliczny); dopuszczalne wartości parametru TF to: ‘tansig’ i ‘logsig’ i ‘purelin’

BTF - nazwa funkcji, wykorzystywanej do treningu sieci (zmienna tekstowa); domyślnie BTF = ‘trainlm’ (metoda Levenberga-Marquardta)

BLF - nazwa funkcji, wykorzyWstywanej do wyznaczania korekcji wag sieci podczas treningu (zmienna tekstowa); domyślnie BLF = 'learngd'; dopuszczalne wartości parametru BLF to: 'learngd' (gradient prosty) i 'learngdm' (gradient prosty z momentum)

PF - funkcja wyznaczająca wartość wskaźnika jakości treningu sieci jednokierunkowej (zmienna tekstowa); domyślnie PF = 'mse' (błąd średniokwadratowy); parametr ten może oznaczać dowolną różniczkowalną funkcję błędu, np. 'msereg' (suma błędu średniokwadratowego i kwadratów wag sieci – metoda regularyzacji wag) lub 'sse' (suma kwadratów błędów)

WYJŚCIE:

NET - struktura (obiekt) zawierająca opis architektury, metod treningu, wartości liczbowe wag oraz inne parametry wielowarstwowej sieci jednokierunkowej.

- **Reguła delta** – reguła uczenia z nauczycielem - Polega na tym, że każdy neuron po otrzymaniu na swoich wejściach określone sygnały (z wejść sieci albo od innych neuronów, stanowiących wcześniejsze piętra przetwarzania informacji) wyznacza swój sygnał wyjściowy wykorzystując posiadaną wiedzę w postaci wcześniej ustalonych wartości współczynników wzmocnienia (wag) wszystkich wejść oraz (ewentualnie) progu. Wartość sygnału wyjściowego, wyznaczonego przez neuron na danym kroku procesu uczenia porównywana jest z odpowiedzią wzorcową podaną przez nauczyciela w ciągu uczącym. Jeśli występuje rozbieżność - neuron wyznacza różnicę pomiędzy swoim sygnałem wyjściowym a tą wartością sygnału, która była by - według nauczyciela prawidłowa. Ta różnica oznaczana jest zwykle symbolem greckiej litery d (delta) i stąd nazwa opisywanej metody.

  Schemat algorytmu :

  Reguła delta obowiązuje dla neuronów z ciągłymi funkcjami aktywacji i nadzorowanego trybu uczenia. Regułę delta wyprowadza się jako wynik minimalizacji kryterium **błędu średniokwadratowego Q.**

$$Q = \frac{1}{2}\sum_{j=1}^{N}\left(z^j - y^j\right)^2 = \sum_{j=1}^{N} Q^j, \qquad Q^j = \frac{1}{2}\left(\delta^j\right)^2$$

Korekta wag:

$$\Delta w_i = \eta \cdot \delta^j \cdot f'(e^j) \cdot x_i^j$$

gdzie $f'()$ oznacza pochodną funkcji aktywacji. W przypadku funkcji sigmoidalnej:

$$\Delta w_i = \eta \cdot \delta^j \cdot (1 - y^j) \cdot y^j \cdot x_i^j$$

- **Algorytm wstecznej propagacji błędu** - Jest to podstawowy algorytm uczenia nadzorowanego wielowarstwowych jednokierunkowych sieci neuronowych. Podaje on przepis na zmianę wag wij dowolnych połączeń elementów przetwarzających rozmieszczonych w sąsiednich warstwach sieci. Oparty jest on na minimalizacji sumy kwadratów błędów uczenia z wykorzystaniem optymalizacyjnej metody największego spadku. Dzięki zastosowaniu specyficznego sposobu propagowania błędów uczenia sieci powstałych na jej wyjściu, tj. przesyłania ich od warstwy wyjściowej do wejściowej, algorytm propagacji wstecznej stał się jednym z najskuteczniejszych algorytmów uczenia sieci.

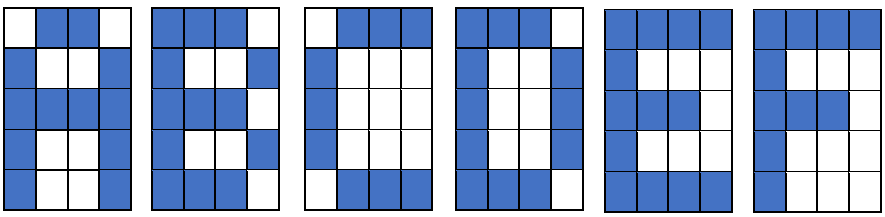
  Kroki algorytmu :

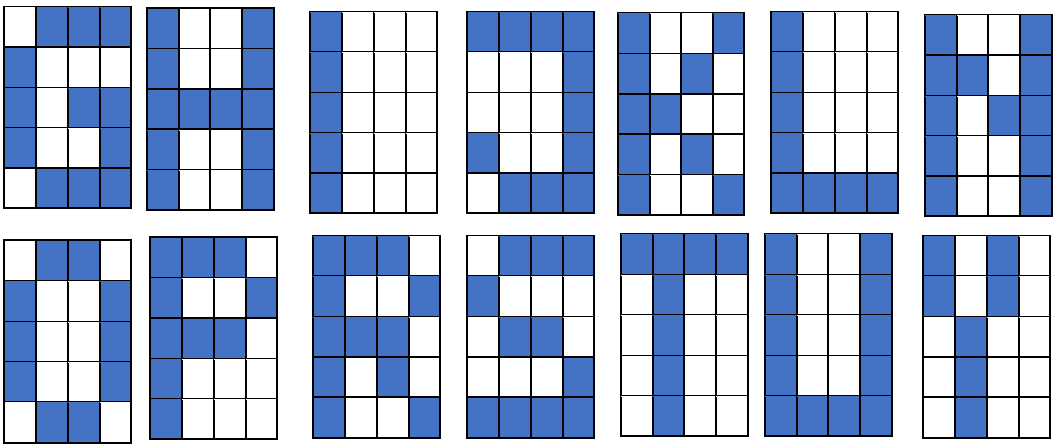
  1. Pobierz ze zbioru uczącego parę wejście-oczekiwane wyjście.
  2. Podaj na wejście neuronu wartości wejściowe.
  3. Przepuść sygnał przez sieć obliczając wyjścia poszczególnych neuronów w kolejnych warstwach.
  4. Oblicz sygnał wyjściowy neuronów z warstwy wyjściowej.
  5. Oblicz błędy w neuronach warstwy wyjściowej - d=(o-y)*f'(z).
  6. Korzystając ze wzoru: $\delta_j = f'(z) * \sum_{k=1}^{K} \delta_{(j+1)k} w_{(j+1)k}$
  przepropaguj (wstecz) błędy przez całą sieć
  7. Uaktualnij (w=w+h*d*x) wszystkie wagi w sieci.
  8. Powyższe kroki powtarzaj dla wszystkich par uczących.
  9. Zakończ naukę gdy całkowity błąd dla wszystkich wzorców uczących będzie mniejszy od pewnej ustalonej na początku nauki wartości.

3. Wykonanie :

Wygenerowałem dane uczące, czyli 20 dużych liter alfabetu
A, B, C, D, E, F, G, H, I, J, K, L, N, O, P, R, S, T, U, Y.

Litery umieściłem w tablicach o rozmiarze 4x5 a następnie tablice przekształciłem w ciąg zer i jedynek, gdzie 1 oznacza pole czarne a 0 – pole białe (puste).





Procedura przekształcania liter do postaci binarnej :

- Wybieramy literę : np. A
- Tabelę 4x5 wypełniam 0 i 1
- A = [0; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1];
- Każdą literę zapisuję w postaci 20 znakowego ciągu
- **Dane wejściowe :**

```
   %A B C D E F G H I J K L N O P R S T U Y
Input=[0 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1;
    1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 1 0 0;
    1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 1 0 1;
    0 0 1 0 1 1 1 1 1 0 1 1 0 1 0 0 0 1 0 1 0;
    1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1;
    0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0;
    0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1;
    1 1 0 1 0 0 0 1 0 1 0 0 1 1 1 1 0 0 1 0;
    1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 0;
    1 1 0 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1;
    1 1 0 0 1 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0;
    1 0 0 1 0 0 1 1 0 1 0 0 1 1 0 0 0 0 1 0;
    1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1;
    0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0;
    1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 0 1 0;
    1 1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 0 0;
    0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 1 1;
    0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 0 1 0;
    1 0 1 0 1 0 1 1 0 1 1 1 1 0 0 1 0 0 0 0;
    ];
```

- **Dane wyjściowe :**

```
   %A B C D E F G H I J K L N O P R S T U Y
Output=[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1;
    ];
```

Dane wejściowe wprowadziłem do sieci i przeprowadziłem jej uczenie funkcją **newff**.

Dane wyjściowe zawierają wartość 1 w kolumnie odpowiadającej danej literze.

Sieć uczyła się rozpoznawać litery z zestawu 20 podanych w danych uczących.

Program określa podobieństwo wprowadzonej litery. Są to liczby z przedziału 0 – 1. Te bliższe 1 oznaczają to że wprowadzona litera zdaje się być prawidłowa.

Listing kodu programu :

```
%% Sprawozdanie 3
close all; clear all; clc;

% Min i max wejsc sieci :
PR=[0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1;
    0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1;];

% Ilosc neuronow w kazdej z warstw :
S=[40 20 20];

% Funkcja NEWFF :
net = newff(PR,S,{'tansig','tansig','tansig'},'traingda');

   %A B C D E F G H I J K L N O P R S T U Y
Input=[0 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1;
    1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 1 0 0;
    1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 1 0 1;
    0 0 1 0 1 1 1 1 0 1 1 0 1 0 0 0 1 0 1 0;
    1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1;
    0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0;
    0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1;
    1 1 0 1 0 0 0 1 0 1 0 0 1 1 1 1 0 0 1 0;
    1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 0;
    1 1 0 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1;
    1 1 0 0 1 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0;
    1 0 0 1 0 0 1 1 0 1 0 0 1 1 0 0 0 0 1 0;
    1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1;
    0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0;
    1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 0 1 0;
    1 1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 0 0;
    0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 1 1;
    0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 0 1 0;
    1 0 1 0 1 0 1 1 0 1 1 1 1 0 0 1 0 0 0 0;
    ];

   %A B C D E F G H I J K L N O P R S T U Y
Output=[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1;
```

```
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1;
    ];

% Parametry uczenia sie :
net.trainParam.epochs = 10000; % Max epochs
net.trainParam.mu = 0.001; % Współczynnik uczenia
net.trainParam.goal = 0.001; % Blad sredniokwadratowy

% Uczenie :
net = train(net, Input, Output);

% Dane testowe :
A = [0; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1];
B = [1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0];
C = [0; 1; 1; 1; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 0; 1; 1; 1];
D = [1; 1; 1; 0; 1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 1; 1; 1; 0];
E = [1; 1; 1; 1; 1; 0; 0; 0; 1; 1; 1; 0; 1; 0; 0; 0; 1; 1; 1; 1];
F = [1; 1; 1; 1; 1; 0; 0; 0; 1; 1; 1; 0; 1; 0; 0; 0; 1; 0; 0; 0];
G = [0; 1; 1; 1; 1; 0; 0; 0; 1; 0; 1; 1; 1; 0; 0; 1; 0; 1; 1; 1];
H = [1; 0; 0; 1; 1; 0; 0; 1; 1; 1; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1];
I = [1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0];
J = [1; 1; 1; 1; 0; 0; 0; 1; 0; 0; 0; 1; 1; 0; 0; 1; 0; 1; 1; 1];
K = [1; 0; 0; 1; 1; 0; 1; 0; 1; 1; 0; 0; 1; 0; 1; 0; 1; 0; 0; 1];
L = [1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 1; 1; 1];
N = [1; 0; 0; 1; 1; 1; 0; 1; 1; 0; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1];
O = [0; 1; 1; 0; 1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 0; 1; 1; 0];
P = [1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0; 1; 0; 0; 0; 1; 0; 0; 0];
R = [1; 1; 1; 0; 1; 0; 0; 1; 1; 1; 1; 0; 1; 0; 1; 0; 1; 0; 0; 1];
S = [0; 1; 1; 1; 1; 0; 0; 0; 0; 1; 1; 0; 0; 0; 0; 1; 1; 1; 1; 0];
T = [1; 1; 1; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0];
U = [1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 1; 0; 0; 1; 0; 1; 1; 0];
Y = [1; 0; 1; 0; 1; 0; 1; 0; 0; 1; 0; 0; 0; 1; 0; 0; 0; 1; 0; 0];

% Symulacje :
efekt=sim(net, A);

% Szukanie najwiekszej wartosci :
max=1;
for i=1:1:20
    if (efekt(max)<efekt(i))
        max=i;
    end;
end

% Wypisywanie jaka to litera :
switch max
    case 1
        disp('Litera : A')
        disp(efekt(1))
    case 2
        disp('Litera : B')
        disp(efekt(2))
    case 3
        disp('Litera : C')
        disp(efekt(3))
    case 4
        disp('Litera : D')
        disp(efekt(4))
    case 5
        disp('Litera : E')
        disp(efekt(5))
    case 6
        disp('Litera : F')
        disp(efekt(6))
    case 7
        disp('Litera : G')
        disp(efekt(7)
```

```
    case 8
        disp('Litera : H')
        disp(efekt(8))
    case 9
        disp('Litera : I')
        disp(efekt(9))
    case 10
        disp('Litera : J')
        disp(efekt(10))
    case 11
        disp('Litera : K')
        disp(efekt(11))
    case 12
        disp('Litera : L')
        disp(efekt(12))
    case 13
        disp('Litera : N')
        disp(efekt(13))
    case 14
        disp('Litera : O')
        disp(efekt(14))
    case 15
        disp('Litera : P')
        disp(efekt(15))
    case 16
        disp('Litera : R')
        disp(efekt(16))
    case 17
        disp('Litera : S')
        disp(efekt(17))
    case 18
        disp('Litera : T')
        disp(efekt(18))
    case 19
        disp('Litera : U')
        disp(efekt(19))
    case 20
        disp('Litera : Y')
        disp(efekt(20))
    otherwise
        disp('Wystapil blad')
end
```

Opis zmiennych oraz funkcji :

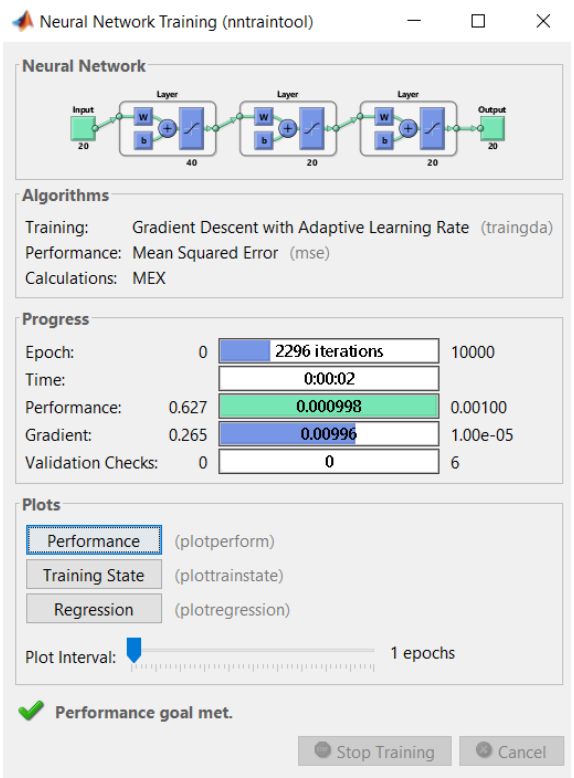
PR - zmienna przechowująca minimalną i maksymalną wartość dla każdego z 20 wejść
S - zmienna przechowująca ilość neutronów w poszczególnych warstwach sieci, 20 stanowi ilość wyjść z sieci, S zawiera 3 wartości zatem sieć będzie 3 warstwowa.
net - zmienna przechowująca sieć neuronową
net = newff(PR,S,{'tansig','tansig','tansig'},'traingda') - wielowarstwowa sieć neutronowa dla wprowadzonego S składającego się z 3 warstw; tansig – parametr określający funkcję aktywacji neuronów w poszczególnych warstwach sieci, tu: tangens hiperboliczny; traingda – funkcja wykorzystywana przy treningu sieci, tu: algorytm propagacji wstecznej błędu
Input - dane wejściowe - 20 dużych liter zapisanych w postaci 0 i 1
Output - dane wyjściowe z wartościami 1 w odpowiedniej dla litery kolumnie
net.trainParam.epochs - maksymalna liczba epochs
net.trainParam.goal - Błąd średniokwadratowy
net.trainParam.mu - Współczynnik uczenia
train(net, Input, Output) - proces uczenia sieci z wykorzystaniem danych wejściowych oraz wyjściowych
efekt = sim(net, A) - symulacja sieci, jako zmienną podajemy literę, która ma zostać rozpoznana
disp(efekt(…)) – Wypisywanie jak podobne są dane litery
switch – pozwala wypisać wykrytą literę

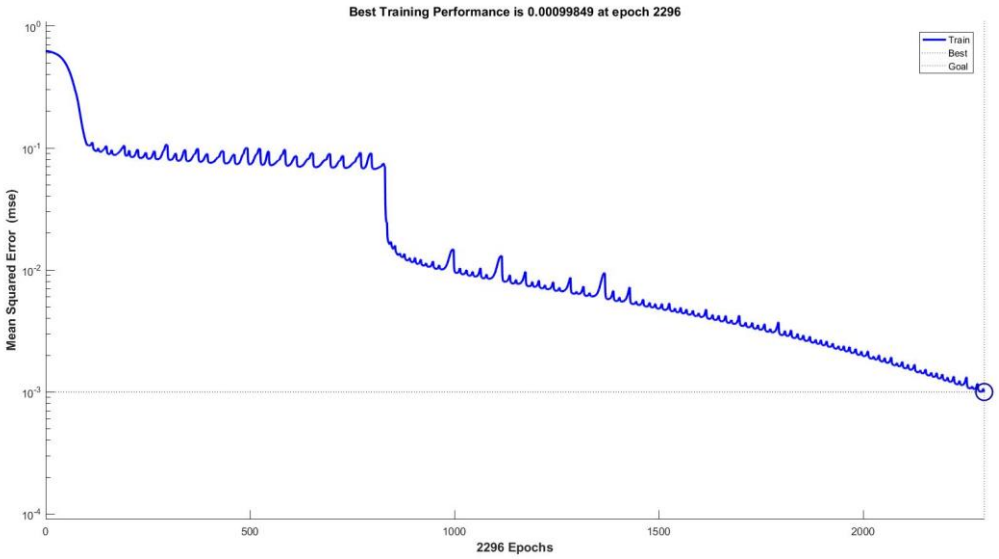
4. Wyniki działania :

```
Sym = sim(net, …); -> Wprowadzamy dużą literę …
Wykresy dla wprowadzanej dużej litery A
Parametry nauki :
net.trainParam.epochs % Max epochs
net.trainParam.mu % Współczynnik uczenia
net.trainParam.goal % Błąd średniokwadratowy
```

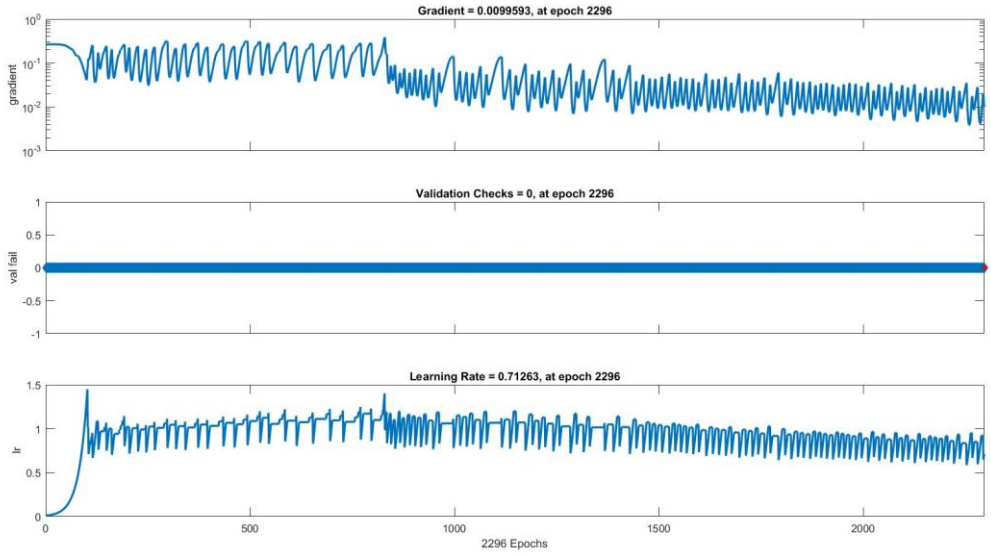
Wprowadzona litera	Współczynnik uczenia	Błąd średniokwadratowy	Podobieństwo litery	Ilość iteracji potrzebna do nauki
A	0.001	0.001	0.9450	3386
	0.01	0.01	0.8869	1656
B	0.001	0.001	0.9503	3640
	0.01	0.01	0.4734	1448
C	0.001	0.001	0.9557	5030
	0.01	0.01	0.7978	1097
P	0.001	0.001	0.7444	5183
	0.01	0.01	0.5775	1388
R	0.001	0.001	0.9257	3761
	0.01	0.01	0.7372	1366
U	0.001	0.001	0.7679	2682
	0.01	0.01	0.9301	1817
Y	0.001	0.001	0.8484	3950
	0.01	0.01	0.8772	1789

Max 10000 epochs
Współczynnik uczenia : 0.001
Błąd średniokwadratowy : 0.001



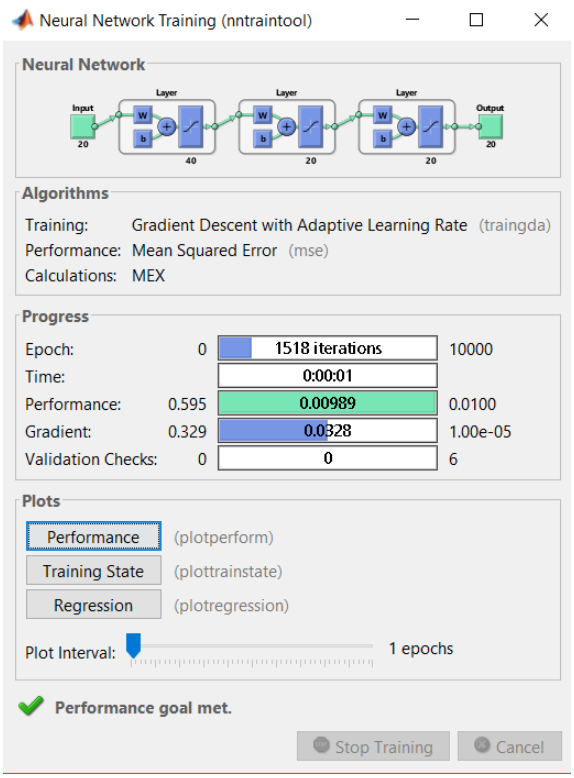


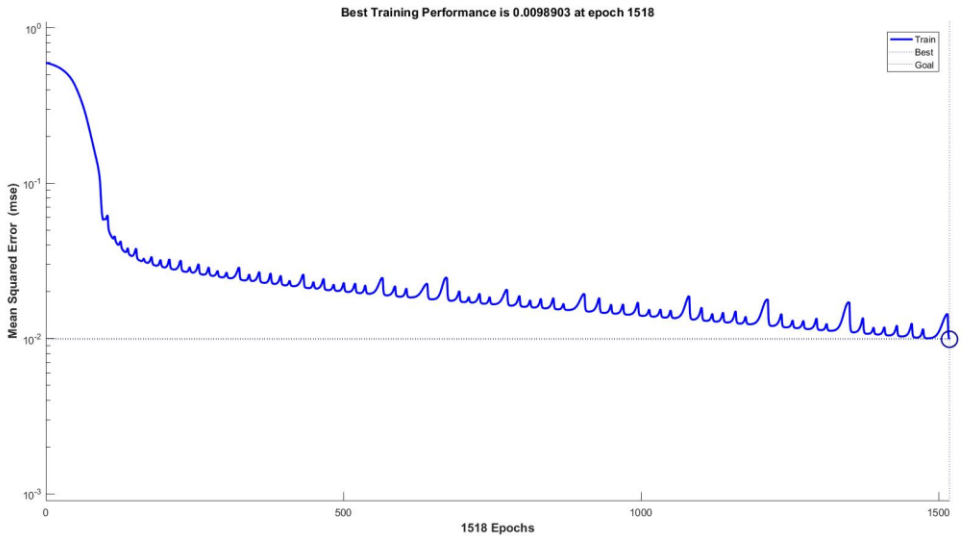
Błąd maleje wraz z liczbą iteracji. Maleje znacznie na początku oraz przy około 800 iteracji. Proces nauki kończy się podczas gdy jest kompletny po 2296 iteracjach



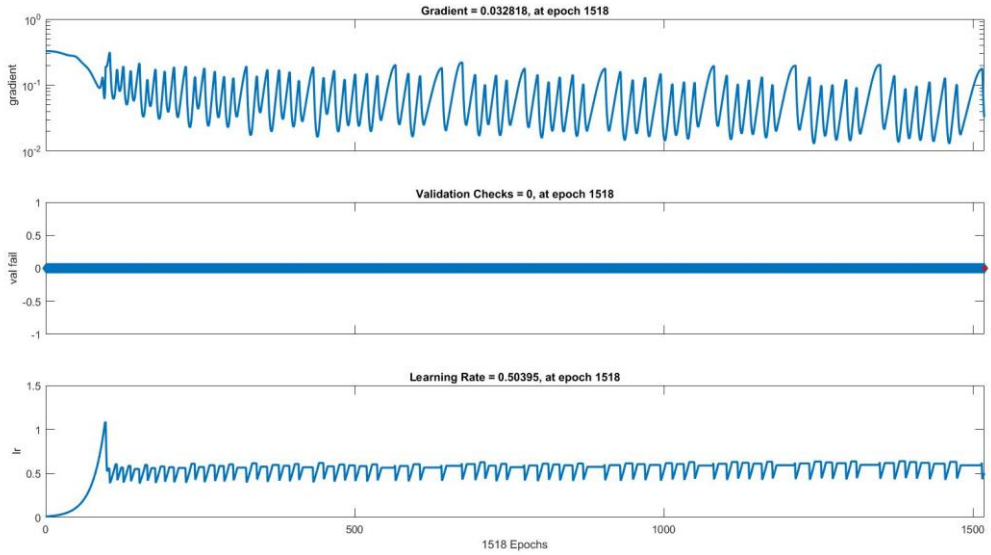
Wskaźnik nauki początkowo rośnie lecz później oscyluje w podobnych wartościach

Max 10000 epochs
Współczynnik uczenia : 0.01
Błąd średniokwadratowy : 0.01





Błąd maleje wraz z liczbą iteracji. Proces nauki kończy się podczas gdy jest kompletny po 1518 iteracjach



Wskaźnik nauki początkowo rośnie lecz później oscyluje w podobnych wartościach

5. Analiza otrzymanych wyników :

Analizując otrzymane wyniki zauważamy, że proces nauki uzależniony jest od współczynnika uczenia oraz błędu średniokwadratowego. Zmieniając te parametry, wpływamy na dokładność nauki oraz szybkość przebiegania tego procesu. Przy mniejszym błędzie średniokwadratowym uzyskiwane podobieństwo liter oscylowało w okolicach 0.9000-1.000 Czyli dokładność nauki była bardzo wysoka. Wraz ze wzrostem dokładności wzrastała ilość iteracji potrzebnych do wykonania tej nauki. Wynosiła ona około między 3000-5000. Natomiast dla większego błędu średniokwadratowego i współczynnika uczenia na poziomie 0.01 podobieństwo rozpoznawanych liter potrafiło wynosić pomiędzy 0.4000-0.7000 lecz czasem dawało dokładność wyższą niż nauka z mniejszym ustalonym błędem. Za to znacznie spadła ilość potrzebnych iteracji i wynosiła ona około 1000-3000. Program zawsze dobrze rozpoznawał wybraną literę przy limicie iteracji ustawionym na 10 000.

Analizując otrzymane wykresy możemy zauważyć, że błąd średniokwadratowy spada wraz z liczbą epok(iteracji) a proces nauki kończy się w momencie osiągniecia najlepszych wyników nie powodując przy tym przeuczenia sieci.
Na ostatnim wykresie zauważamy, że na samym początku mocno wzrasta nam wskaźnik nauki i następnie maleje i przez resztę iteracji oscyluje na podobnym poziomie.

6. Wnioski :

Analizując wykres zmiany wskaźnika nauki podczas treningu sieci zauważamy, że największe wahania wartości wskaźnika występują na początku treningu, po przekroczeniu pewnej ilości iteracji zachodzi zmiana i obserwujemy że jego wartości oscylują w podobnych wartościach. Zaobserwowałem że istnieje punkt, po przekroczeniu którego jakość nauki przestaje już wzrastać. Projektując więc taką sieć i chcąc uzyskać jak najlepsze efekty i wydajność należy mieć na uwadze potrzebę zlokalizowania takiego punktu i przerwania w nim procesu nauki.

Przeprowadzone pomiary potwierdzają to, że sieć jest w stanie bardzo dobrze rozróżniać od siebie litery czy wprowadzone kształty pod warunkiem że otrzyma na wejściu dane dobrej jakości. Również bardzo ważne są parametry nauki. Należy ustalić odpowiednio wysoką maksymalną liczbę epochs aby program nie zakończył procesu nauki, gdy nie jest on kompletny. Zmniejszając możliwy błąd powodujemy wzrost liczby iteracji, lecz skutkuje to dokładniejszą nauką. Natomiast zwiększając możliwość błędu, automatycznie spada nam jakość rozpoznawania liter oraz liczba iteracji do tego potrzebna.
Sieć wielowarstwowa okazała się bardzo skuteczna podczas tego zadania. Zwiększając współczynnik uczenia wpływaliśmy na dokładniejszy proces nauki.