20.11.2017

Akademia Górniczo-Hutnicza
Stanisława Staszica w Krakowie

Podstawy Sztucznej Inteligencji

Sprawozdanie numer 3

Budowa i działanie sieci wielowarstwowej

Inżynieria Obliczeniowa
Urszula Ślusarz
nr. indeksu: 286132

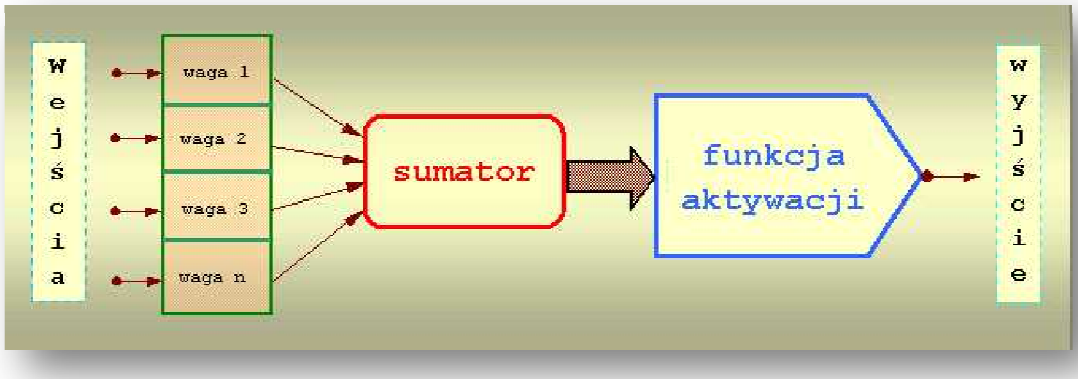
1. Cel ćwiczenia

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

2. Wstęp teoretyczny

Neuron

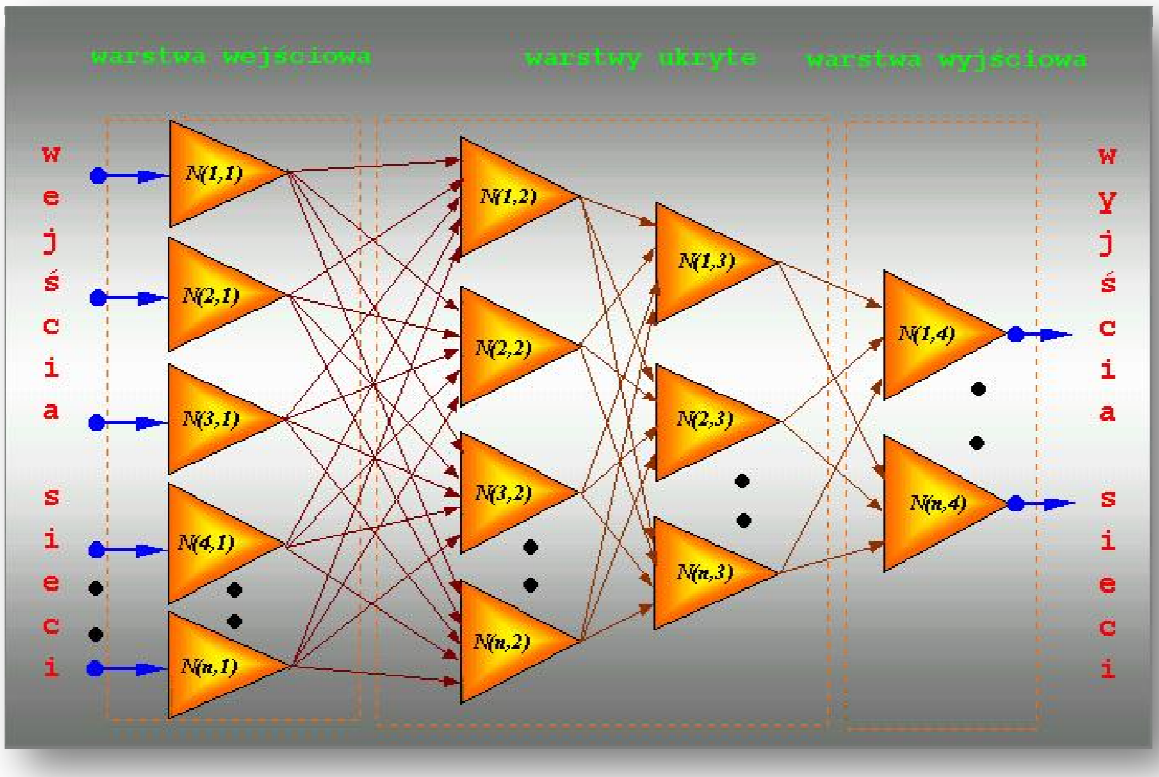
Jego schemat został opracowany przez McCullocha i Pittsa w 1943 roku i oparty został na budowie komórki nerwowej.



Jego działanie jest następujące:
Do wejść doprowadzane są sygnały dochodzące z neuronów warstwy poprzedniej. Każdy sygnał mnożony jest przez odpowiadającą mu wartość liczbową zwaną wagą. Wpływa ona na percepcję danego sygnału wejściowego i jego udział w tworzeniu sygnału wyjściowego przez neuron. Waga może być pobudzająca - dodatnia lub opóźniająca – ujemna; jeżeli nie ma połączenia między neuronami to waga jest równa zero. Zsumowane iloczyny sygnałów i wag stanowią argument **funkcji aktywacji** neuronu.

Sieć wielowarstwowa

Jej schemat jest następujący:



Projektowanie sieci neuronowej zaczyna się już na poziomie analizy sformułowanego problemu. Inaczej mówiąc, jakie i ile danych chcemy lub możemy podać na wejścia sieci (zdeterminuje to wielkość warstwy wejściowej) oraz jaką odpowiedź chcemy uzyskać (ilość wyjść sieci). Pozostanie zatem do określenia ilość warstw ukrytych i neuronów w tych warstwach. Jest to natrudniejszy moment tego etapu pracy. Przyjmuje się, że sieć z jedną warstwą ukrytą powinna nauczyć się rozwiązywania większości postawionych problemów. Nie znane są problemy wymagające do rozwiązania sieci z więcej niż trzema warstwami ukrytymi. Nie ma natomiast dobrej recepty na dobór właściwej ilości neuronów w warstwie ukrytej. Można próbować według wzoru:

$$N_{ww} = \sqrt{N_{wwe} * N_{wwy}}$$

gdzie:

N_{wu} - *ilość neuronów w warstwie ukrytej*
N_{wwe} - *ilość neuronów w warstwie wejściowej*
N_{wwy} - *ilość neuronów w warstwie wyjściowej*

Generalnie jednak uczenie rozpoczyna się z małą ich ilością a następnie, obserwując postępy tego procesu, doświadczalnie zwiększa się ich ilość.

Działanie SSN polega na tym, że sygnały pobudzające (wektor wejściowy) podawane na wejścia sieci, przetwarzane są w poszczególnych neuronach. Po tej projekcji na wyjściach sieci otrzymuje się wartości liczbowe, które stanowią odpowiedź sieci na pobudzenie i stanowią rozwiązanie postawionego problemu. Jednak aby takie rozwiązanie uzyskać, należy przejść żmudną drogę uczeń.

Uczenie metodą wstecznej propagacji błędów

Jest to uczenie z nadzorem lub inaczej – z nauczycielem.

Pierwszą czynnością w procesie uczenia jest przygotowanie dwóch ciągów danych ***uczącego*** i ***weryfikującego***. ***Ciąg uczący*** jest to zbiór takich danych, które w miarę dokładnie charakteryzują dany problem. Jednorazowa porcja danych nazywana jest ***wektorem uczącym***. W jego skład wchodzi ***wektor wejściowy*** czyli te dane wejściowe, które podawane są na wejścia sieci i ***wektor wyjściowy*** czyli takie dane oczekiwane, jakie sieć powinna wygenerować na swoich wyjściach.

Po przetworzeniu wektora wejściowego, nauczyciel porównuje wartości otrzymane z wartościami oczekiwanymi i informuje sieć czy odpowiedź jest poprawna, a jeżeli nie, to jaki powstał błąd odpowiedzi. Błąd ten jest następnie propagowany do sieci ale w odwrotnej niż wektor wejściowy kolejności (od warstwy wyjściowej do wejściowej) i na jego podstawie następuje taka korekcja wag w każdym neuronie, aby ponowne przetworzenie tego samego wektora wejściowego spowodowało zmniejszenie błędu odpowiedzi. Procedurę taką powtarza się do momentu wygenerowania przez sieć błędu mniejszego niż założony. Wtedy na wejście sieci podaje się kolejny wektor wejściowy i powtarza te czynności. Po przetworzeniu całego ***ciągu uczącego***(proces ten nazywany jest ***epoką***) oblicza się błąd dla ***epoki*** i cały cykl powtarzany jest do momentu, aż błąd ten spadnie poniżej dopuszczalnego. Jak to już było zasygnalizowane wcześniej, SSN wykazują tolerancję na nieciągłości, przypadkowe zaburzenia lub wręcz niewielkie braki w zbiorze uczącym. Jest to wynikiem właśnie zdolności do uogólniania wiedzy.

Jeżeli mamy już nauczoną sieć, musimy zweryfikować jej działanie. W tym momencie ważne jest podanie na wejście sieci wzorców z poza zbioru treningowego w celu zbadania czy sieć może efektywnie generalizować zadanie, którego się nauczyła. Do tego używamy ***ciągu weryfikującego***, który ma te same cechy co ***ciąg uczący*** tzn. dane dokładnie charakteryzują problem i znamy dokładne odpowiedzi. Ważne jest jednak, aby dane te nie były używane uprzednio do uczenia. Dokonujemy zatem prezentacji ciągu weryfikującego z tą różnicą, że w tym procesie nie rzutujemy błędów wstecz a jedynie rejestrujemy ilość odpowiedzi poprawnych i na tej podstawie orzekamy, czy sieć spełnia nasze wymagania czyli jak została nauczona.

Wagi początkowe, z którymi sieć rozpoczyna naukę z reguły stanowią liczby wygenerowane przypadkowo. Po nauczeniu sieci zawsze warto dla sprawdzenia otrzymanych wyników powtórzyć całą procedurę od wygenerowania wag początkowych.
Dla dużych sieci i ciągów uczących składających się z wielu tysięcy wektorów uczących ilość obliczeń wykonywanych podczas całego cyklu uczenia jest gigantyczna a więc i czasochłonna. Nie zdarza się także aby sieć została dobrze zbudowana od razu. Zawsze jest ona efektem wielu prób i błędów. Ponadto nigdy nie mamy gwarancji, że nawet prawidłowa sieć nie utknie w minimum lokalnym podczas gdy interesuje nas znalezienie minimum globalnego. Dlatego algorytmy realizujące ***SSN*** wyposaża się mechanizmy dające nauczycielowi możliwość regulacji szybkości i jakości uczenia. Są to tzw. ***współczynniki***: ***uczenia*** i ***momentu***. Wpływają one na stromość funkcji aktywacji i regulują szybkość wpływu zmiany wag na proces uczenia.

3. Wykonane kroki scenariusza

✓ W pierwszej kolejności wygenerowałam dane uczące i testujące, które zawierają 20 dużych liter w postaci dwuwymiarowej tablicy np. 5x7 pikseli dla jednej litery.

Litery wybrane przeze mnie: A C D E F G H K M N O P R S T U W X Y Z

Dane uczące, poniższa tabele przedstawiają matryce 20 liter po konwersji zero-jedynkowej:

```
letterA=   [0 0 1 0 0 ...
            0 1 0 1 0 ...
            0 1 0 1 0 ...
            1 0 0 0 1 ...
            1 1 1 1 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ]';

letterC=   [0 1 1 1 0 ...
            1 0 0 0 1 ...
            1 0 0 0 0 ...
            1 0 0 0 0 ...
            1 0 0 0 0 ...
            1 0 0 0 1 ...
            0 1 1 1 0]';

letterD  = [1 1 1 1 0 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 1 1 1 0 ]';
```

```
letterE  = [1 1 1 1 1 ...
            1 0 0 0 0 ...
            1 0 0 0 0 ...
            1 1 1 1 0 ...
            1 0 0 0 0 ...
            1 0 0 0 0 ...
            1 1 1 1 1 ]';

letterF =  [1 1 1 1 1 ...
            1 0 0 0 0 ...
            1 0 0 0 0 ...
            1 1 1 1 0 ...
            1 0 0 0 0 ...
            1 0 0 0 0 ...
            1 0 0 0 0 ]';

letterG =  [0 1 1 1 0 ...
            1 0 0 0 1 ...
            1 0 0 0 0 ...
            1 0 1 1 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            0 1 1 1 0]';

letterH =  [1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 1 1 1 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1]';

letterK =  [1 0 0 0 1 ...
            1 0 0 1 0 ...
            1 0 1 0 0 ...
            1 1 0 0 0 ...
            1 0 1 0 0 ...
            1 0 0 1 0 ...
            1 0 0 0 1]';

letterM=   [1 0 0 0 1 ...
            1 1 0 1 1 ...
            1 0 1 0 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1]';

letterN=   [1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 1 0 0 1 ...
            1 0 1 0 1 ...
            1 0 0 1 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1]';
```

```
letterO=   [0 1 1 1 0 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            0 1 1 1 0]';

letterP=   [1 1 1 1 0 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 1 1 1 0 ...
            1 0 0 0 0 ...
            1 0 0 0 0 ...
            1 0 0 0 0]';

letterR=   [1 1 1 1 0 ...
            1 0 0 0 1  ...
            1 0 0 0 1 ...
            1 1 1 1 0 ...
            1 0 1 0 0 ...
            1 0 0 1 0 ...
            1 0 0 0 1]';

letterS=   [0 1 1 1 0 ...
            1 0 0 0 1 ...
            1 0 0 0 0 ...
            0 1 1 1 0 ...
            0 0 0 0 1 ...
            1 0 0 0 1 ...
            0 1 1 1 0]';

letterT=   [1 1 1 1 1 ...
            0 0 1 0 0 ...
            0 0 1 0 0 ...
            0 0 1 0 0 ...
            0 0 1 0 0 ...
            0 0 1 0 0 ...
            0 0 1 0 0]';

letterU=  [1 0 0 0 1 ...
           1 0 0 0 1 ...
           1 0 0 0 1 ...
           1 0 0 0 1 ...
           1 0 0 0 1 ...
           1 0 0 0 1 ...
           0 1 1 1 0]';

letterX=  [1 0 0 0 1 ...
           1 0 0 0 1 ...
           1 0 0 0 1 ...
           1 0 0 0 1 ...
           1 0 1 0 1 ...
           1 0 1 0 1 ...
           0 1 0 1 0]';
```

```
letterW=  [1 0 0 0 1 ...
           1 0 0 0 1 ...
           0 1 0 1 0 ...
           0 0 1 0 0 ...
           0 1 0 1 0 ...
           1 0 0 0 1 ...
           1 0 0 0 1]';

letterY=  [1 0 0 0 1 ...
           1 0 0 0 1 ...
           0 1 0 1 0 ...
           0 0 1 0 0 ...
           0 0 1 0 0 ...
           0 0 1 0 0 ...
           0 0 1 0 0]';

letterZ=  [1 1 1 1 1 ...
           0 0 0 0 1 ...
           0 0 0 1 0 ...
           0 0 1 0 0 ...
           0 1 0 0 0 ...
           1 0 0 0 0 ...
           1 1 1 1 1]';
```

Liniowym rozwinięciem tych matryc będą wektory wejściowe ciągu uczącego:

```
189 -    in=[0 0 1 1 1 0 1 1 1 1 0 1 1 0 1 1 1 1 1 1
190          1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 1
191          1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 1
192          1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 1
193          0 0 0 1 1 0 1 1 1 1 0 0 0 0 1 1 1 1 1 1
194          1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0
195          0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
196          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
197          0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
198          1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1
199          1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0
200          0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0
201          0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0
202          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
203          1 0 1 0 0 0 1 0 1 1 1 1 1 1 0 0 1 1 0 0 0
204          1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 0 0
205          1 0 0 1 1 0 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0
206          1 0 0 1 1 1 1 0 0 1 0 1 1 1 1 0 0 1 1 1 1
207          1 0 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0
208          1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 1 0 0 0 0
209          1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 0 0
210          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
211          0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 1 0
212          0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0
213          1 0 1 0 0 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0
214          1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1
215          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
216          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0
217          0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0
218          1 1 1 0 0 1 1 0 1 1 1 0 0 1 0 1 1 1 0 0
219          1 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 0 1
220          0 1 1 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1
221          0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 1 0 0 1 1
222          0 1 1 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1
223          1 0 0 1 0 0 1 1 1 1 0 0 1 0 0 0 0 1 0 1];
224        %A C D E F G H K M N O P R S T U W X Y Z
```

Ciąg uczący składa się z 20 następujących wektorów uczących :

Wektor wejściowy - 35 wartości
Wektor wyjściowy - 20 wartości

Wyjściowa (out) macierz, liczba 1 reprezentuje występowanie danej litery w odpowiedniej kolumnie:

```
out=[ 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 testujące:

```
testujacyA=[0;1;1;1;0;1;0;0;0;1;1;0;0;0;1;1;1;1;1;1;1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;];
          %1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
          %A C D E F G H K M N O P R S T U W X Y Z
testujacyC=[0;1;1;1;0;1;0;0;0;1;1;0;0;0;0;1;0;0;0;0;1;0;0;0;0;1;0;0;0;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
          %A C D E F G H K M N O P R S T U W X Y Z
testujacyD=[1;1;1;1;0;1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;1;1;1;1;0;];
          %0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
          %A C D E F G H K M N O P R S T U W X Y Z
testujacyE=[1;1;1;1;1;1;0;0;0;0;1;0;0;0;0;1;1;1;1;0;1;0;0;0;0;1;0;0;0;0;1;1;1;1;1;];
          %0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
          %A C D E F G H K M N O P R S T U W X Y Z
testujacyF=[1;1;1;1;1;1;0;0;0;0;1;0;0;0;0;1;1;1;1;0;1;0;0;0;0;1;0;0;0;0;1;0;0;0;0;];
          %0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
          %A C D E F G H K M N O P R S T U W X Y Z
testujacyG=[0;1;1;1;0;1;0;0;0;1;1;0;0;0;0;1;0;1;1;1;1;0;0;0;1;1;0;0;0;1;0;1;1;1;0;];
          %0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
          %A C D E F G H K M N O P R S T U W X Y Z
testujacyH=[1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;1;1;1;1;1;1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;];
          %0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
          %A C D E F G H K M N O P R S T U W X Y Z
testujacyK=[1;0;0;0;1;1;0;0;1;0;1;0;1;0;0;1;1;0;0;0;1;0;1;0;0;1;0;0;1;0;1;0;0;0;1;];
          %0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
          %A C D E F G H K M N O P R S T U W X Y Z
testujacyM=[1;0;0;0;1;1;1;0;1;1;1;0;1;0;1;1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;];
          %0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
          %A C D E F G H K M N O P R S T U W X Y Z
testujacyN=[1;0;0;0;1;1;1;0;0;0;1;1;1;0;0;1;1;0;1;0;1;1;0;0;1;1;1;0;0;0;1;1;0;0;0;1;];
          %0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
          %A C D E F G H K M N O P R S T U W X Y Z
```

```
testujacyO=[0;1;1;1;0;1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;0;1;1;1;0;];
            %0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
            %A C D E F G H K M N O P R S T U W X Y Z
testujacyP=[1;1;1;1;0;1;0;0;0;1;1;0;0;0;1;1;1;1;1;0;1;0;0;0;0;1;0;0;0;0;1;0;0;0;0;];
            %0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
            %A C D E F G H K M N O P R S T U W X Y Z
testujacyR=[1;1;1;1;0;1;0;0;0;1;1;0;0;0;1;1;1;1;1;0;1;0;1;0;0;1;0;0;1;0;1;0;0;0;1;];
            %0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
            %A C D E F G H K M N O P R S T U W X Y Z
testujacyS=[0;1;1;1;0;1;0;0;0;1;1;0;0;0;0;0;1;1;1;0;0;0;0;0;1;1;0;0;0;1;0;1;1;1;0;];
            %0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
            %A C D E F G H K M N O P R S T U W X Y Z
testujacyT=[1;1;1;1;1;0;0;1;0;0;0;0;1;0;0;0;0;1;0;0;0;0;1;0;0;0;0;1;0;0;0;0;1;0;0;];
            %0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
            %A C D E F G H K M N O P R S T U W X Y Z
testujacyU=[1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;0;1;1;1;0;];
            %0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
            %A C D E F G H K M N O P R S T U W X Y Z
testujacyW=[1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;1;0;0;0;1;1;0;1;0;1;1;0;1;0;1;0;1;0;1;0;];
            %0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
            %A C D E F G H K M N O P R S T U W X Y Z
testujacyX=[1;0;0;0;1;1;0;0;0;1;0;1;0;1;0;0;0;1;0;0;0;1;0;1;0;1;0;0;0;1;1;0;0;0;1;];
            %0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
            %A C D E F G H K M N O P R S T U W X Y Z
testujacyY=[1;0;0;0;1;1;0;0;0;1;0;1;0;1;0;0;0;1;0;0;0;0;1;0;0;0;0;1;0;0;0;0;1;0;0;];
            %0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
            %A C D E F G H K M N O P R S T U W X Y Z
testujacyZ=[1;1;1;1;1;0;0;0;0;1;0;0;0;1;0;0;0;1;0;0;0;1;0;0;0;1;0;0;0;0;1;1;1;1;1;];
            %0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
            %A C D E F G H K M N O P R S T U W X Y Z
```

Litery są utworzone na matrycy 5 x 7 czyli na **35 polach**. I tyle wejść ma sieć, mogą zostać wypełnione tylko liczbą 0(min) albo 1(max);

```
min_max=[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;
         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];
```

✓ Kolejnym krokiem było przygotowanie (implementacja lub wykorzystanie gotowych narzędzi) wielowarstwowej sieci oraz algorytmu wstecznej propagacji błędu.

Krótki opis i budowa funkcji:

newff - Tworzenie wielowarstwowej jednokierunkowej sieci neuronowej, złożonej z neuronów o nieliniowych funkcjach aktywacji.

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 *R*x*2*, 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, wykorzystywanej 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.

```
10 -    ilosc_wyjsc=[40 20 20]; %ilosc neuronow w kazdej wartswie
11
12      % TFi - nazwa funkcji aktywacji neuronów w i-tej warstwie sieci (zmienna
13      % tekstowa) domyslna = 'tansig' (tangens hiperboliczny);
14      %dopuszczalne wartości parametru TF to: 'tansig' i 'logsig' i 'purelin'
15
16      %traingda metoda propagacji wstecznej błędu z adaptacyjną zmianą stałej szybkości uczenia,
17 -    net = newff(min_max,ilosc_wyjsc,{'tansig','logsig','purelin'},'traingda');
```

W naszym zadaniu wykorzystuję funkcję **traingda**, czyli metodę propagacji wstecznej błędu z adaptacyjną zmianą stałej szybkości uczenia, a warstwy składają się z neuronów o tangensoidalnych funkcjach aktywacji.

Następnie dokonałam uczenia sieci, dla różnych współczynników uczenia, maksymalnej liczby epok trwania treningu oraz danego błędu średniokwadratowego.

```
307 -    net.trainParam.epochs = 2500;
308 -    net.trainParam.goal = 0.01;
309 -    net.trainParam.mu = 0.01;
310
311 -    net= train(net, in, out);
```

```
313      %Wywołujemy przykładową literę z alfabetu, może to być A:
314 -    literka=testujacyA;
315      %Wyświetlamy ją:
316 -    plotchar(literka);
317
318 -    a=sim(net,literka); %testowanie sieci
319 -    maximum=1;
320 -    for i=1:1:20
321 -        if (a(maximum)<a(i))
322 -            maximum=i;
323 -        end;
324 -    end
```

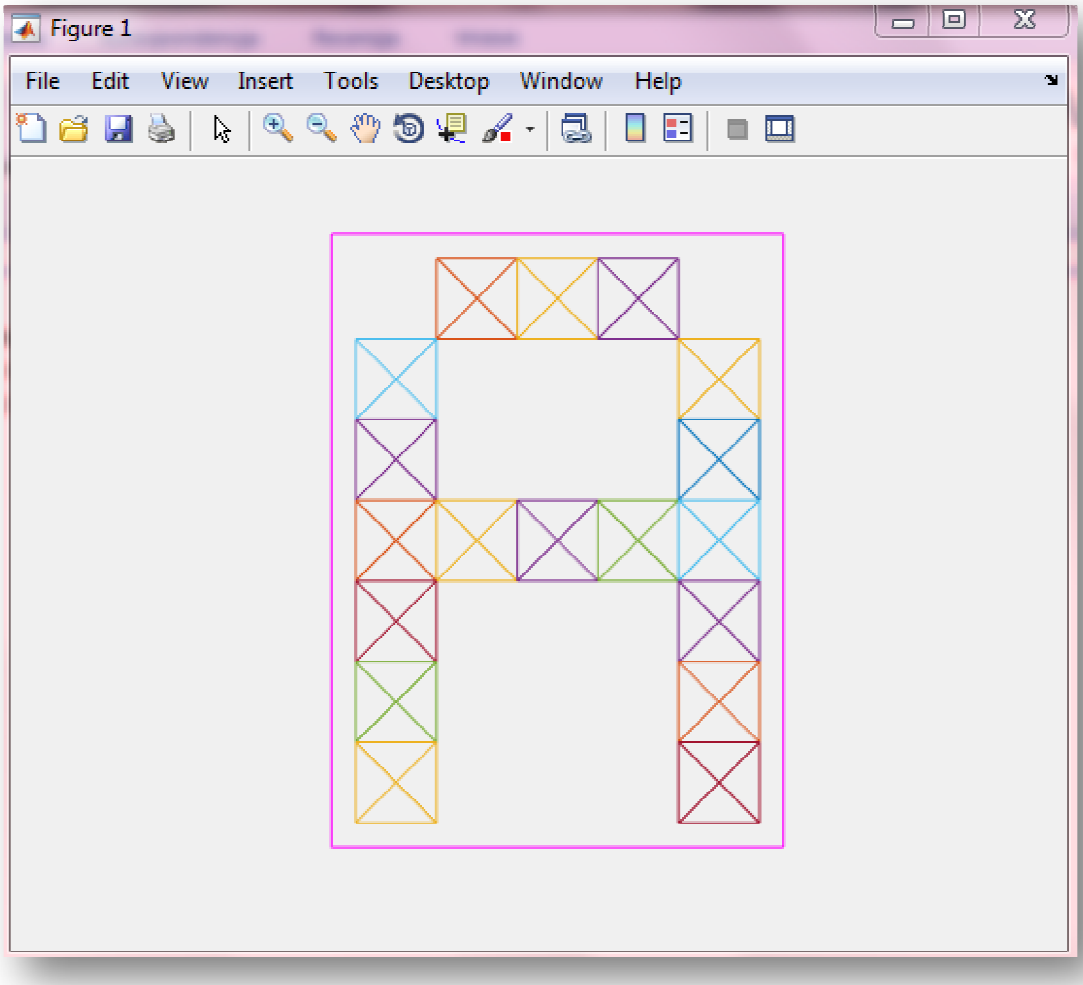
Następnie przy użyciu switcha możemy wyszukać największą wartość przypisaną do litery. Poniżej przedstawię użycie „switcha” dla kilku przykładowych liter.

```
329 -     switch maximum
330 -         case 1
331 -             disp(' litera  A')
332 -             disp('A='),disp(a(1));
333 -         case 2
334 -             disp(' litera  C')
335 -             disp('C='),disp(a(2));
336 -         case 3
337 -             disp(' litera  D')
338 -             disp('D='),disp(a(3));
339 -         case 4
340 -             disp(' litera  E')
341 -             disp('E='),disp(a(4));
```

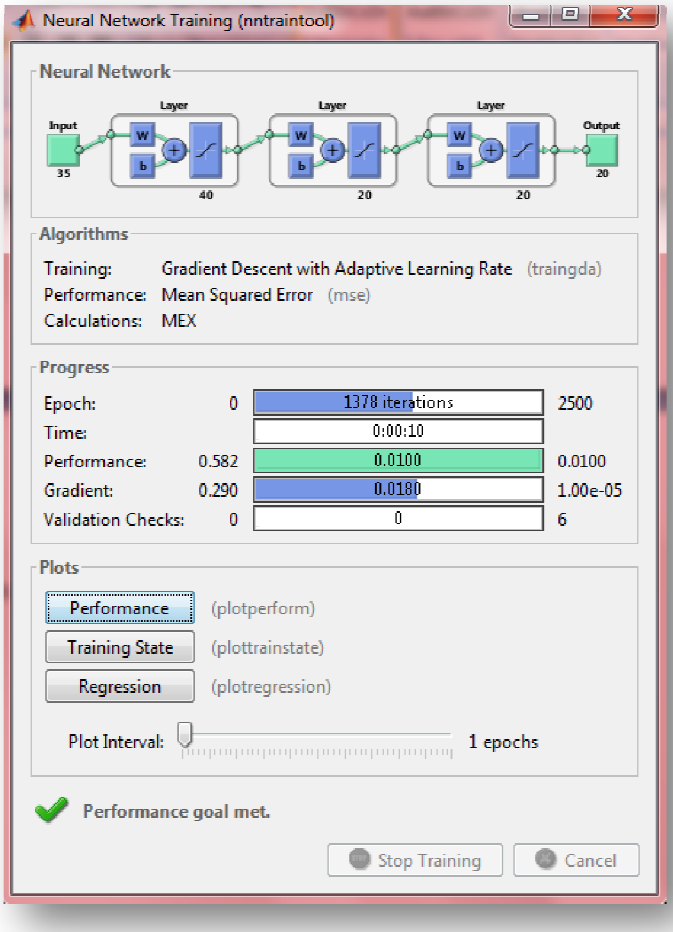
Przedstawiam działanie programu:

1. Wywołuję literkę „A”.

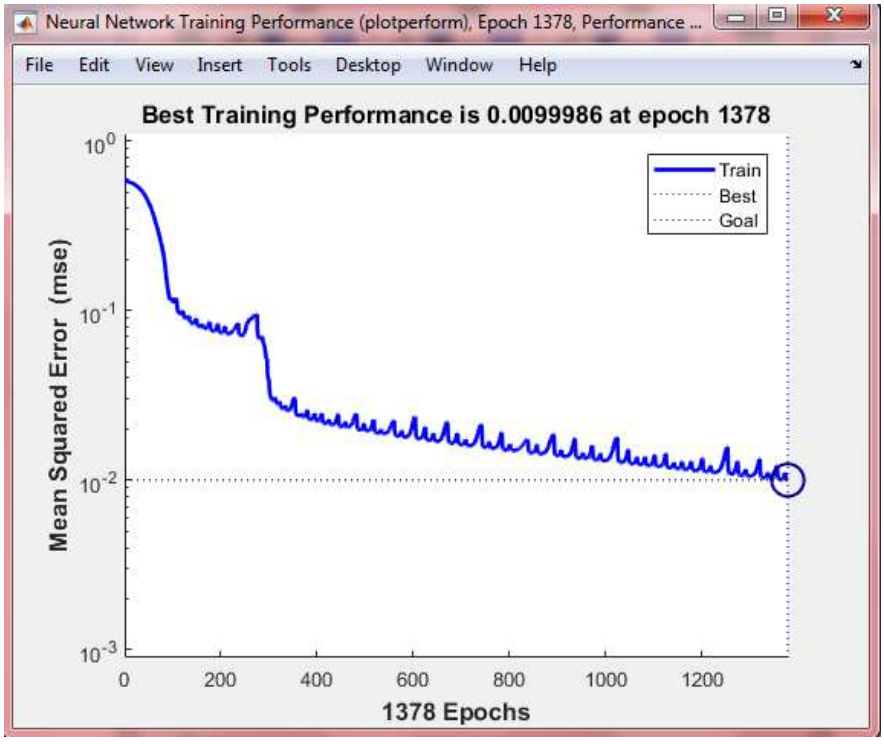
2.Otrzymuję obraz tej literki:



3.Otrzymuję „Neural Network Training”:



4. Plots-> Performance



5. Zebranie wyniki w poniższej tabeli świadczą o tym, którą literę najbardziej przypomina wybrana przeze mnie literka do nauki.

Błąd średniokwadratowy	0.1		
Współ. uczenia:	0,1	0.01	0,001
Liczba epok:	84	127	93
Litera	Dokładność	Dokładność	Dokładność
A	**0.4178**	0.3870	**0.7644**
C	-0.0074	0.1770	-0.0237
D	-0.1829	-0.4415	0.0273
E	-0.0236	0.1257	-0.0525
F	0.1842	0.2838	-0.2954
G	0.0885	-0.0607	0.0808
H	0.2222	-0.1722	-0.1164
K	-0.1432	-0.1799	-0.0168
M	-0.3269	-0.0490	-0.0361
N	0.3833	0.1529	0.1564
O	-0.1992	-0.0497	0.1217
P	0.0632	-0.0304	-0.0061
X		**0.8141**	

Błąd średniokwadratowy	0.01		
Współ. uczenia:	0,1	0.01	0,001
Liczba epok:	1364	1260	1501
Litera	Dokładność	Dokładność	Dokładność
A	**0.4529**	**0.7050**	**0.8004**
C	0.0516	-0.0529	-0.0086
D	0.1387	-0.0094	0.0112
E	0.0272	-0.0428	0.0630
F	-0.0461	-0.0629	-0.0271
G	0.1956	0.0829	0.0151
H	0.1195	0.0439	-0.0227
K	-0.0267	-0.0721	-0.0105
M	0.0110	-0.1782	0.0701
N	-0.0224	-0.0247	-0.0579
O	0.3605	0.0850	-0.0618
P	0.0844	0.1017	-0.0322

Błąd średniokwadratowy	0.001		
Współ. uczenia:	0,1	0.01	0,001
Liczba epok:	2500	2500	2500
Litera	Dokładność	Dokładność	Dokładność
A	**0.7794**	**0.7220**	**0.7510**
C	0.0147	0.0105	-0.0202
D	0.0464	-0.2220	-0.0408
E	0.0128	-0.0255	-0.0851
F	-0.0670	0.0558	-0.0376
G	0.0362	0.0600	0.0141
H	-0.0225	0.0549	-0.0100
K	-0.0621	-0.0162	0.0494
M	0.0111	0.0605	0.0657
N	-0.0641	0.0113	0.0760
O	-0.0095	0.1831	-0.0635
P	0.0600	-0.0062	-0.0426

4. Podsumowanie

Patrząc na umieszczone wyżej tabele działania funkcji newff, widzimy że przy błędzie średniokwadratowym 0,1 i współczynniku uczenia 0,01 wystąpił błąd w określeniu litery, którą przypisaliśmy do testu. Przy błędzie średniokwadratowym 0,01 wszystkie wyniki były prawidłowe, niezależnie od współczynnika uczenia, liczba epok potrzebnych do nauki oscylowała między 1260 a 1501. Ostatnią próbę wykonałam dla błędu średniokwadratowego równego 0,001 w tym przypadku również wyniki wyznaczenia litery były wszystkie prawidłowe i liczba epok do nauki była zawsze równa 2500 niezależnie od współczynnika uczenia. Dużą dokładnością rozpoznawania konkretnych liter charakteryzuje się wyznaczanie przy błędzie średniokwadratowym 0,001, jednak to on potrzebuje największej liczby epok (iteracji) do nauki.