Podstawy Sztucznej Inteligencji Monika Michalik – gr.2

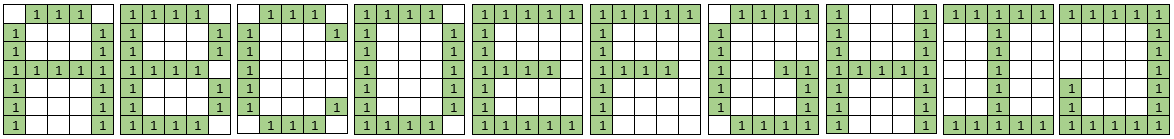
**Scenariusz 3. Budowa i działanie sieci wielowarstwowej.**

**Cel ćwiczenia:**

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

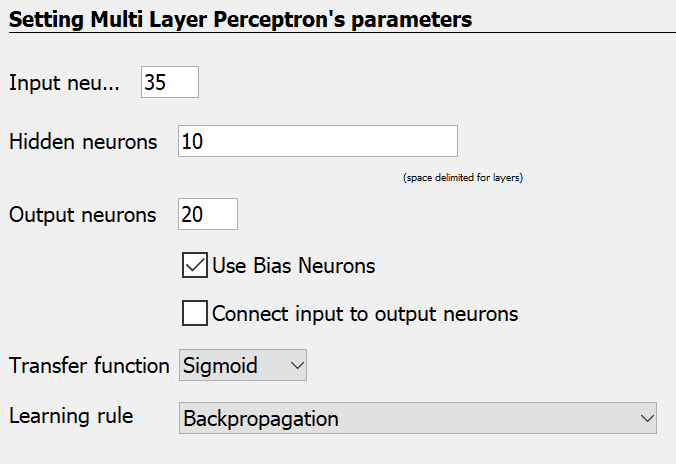
**Zadania do wykonania:**

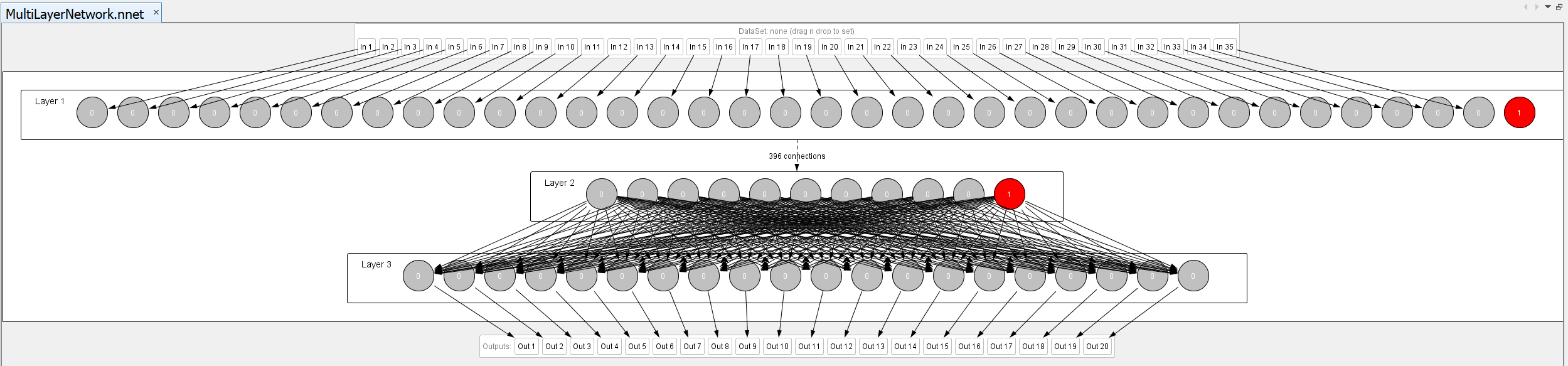
**1. Wygenerowanie danych uczących i testujących, zawierających 20 dużych liter alfabetu w postaci dwuwymiarowej tablicy 5x7 pikseli dla jednej litery.**

****

**2. Przygotowanie wielowarstowej sieci oraz algorytmu wstecznej propagacji błędu z wykorzystaniem gotowych narzędzi.**

Warstwa wejściowa posiada 35 neuronów. Warstwa ukryta posiada 10 neuronów a warstwa wyjściowa 20.



****

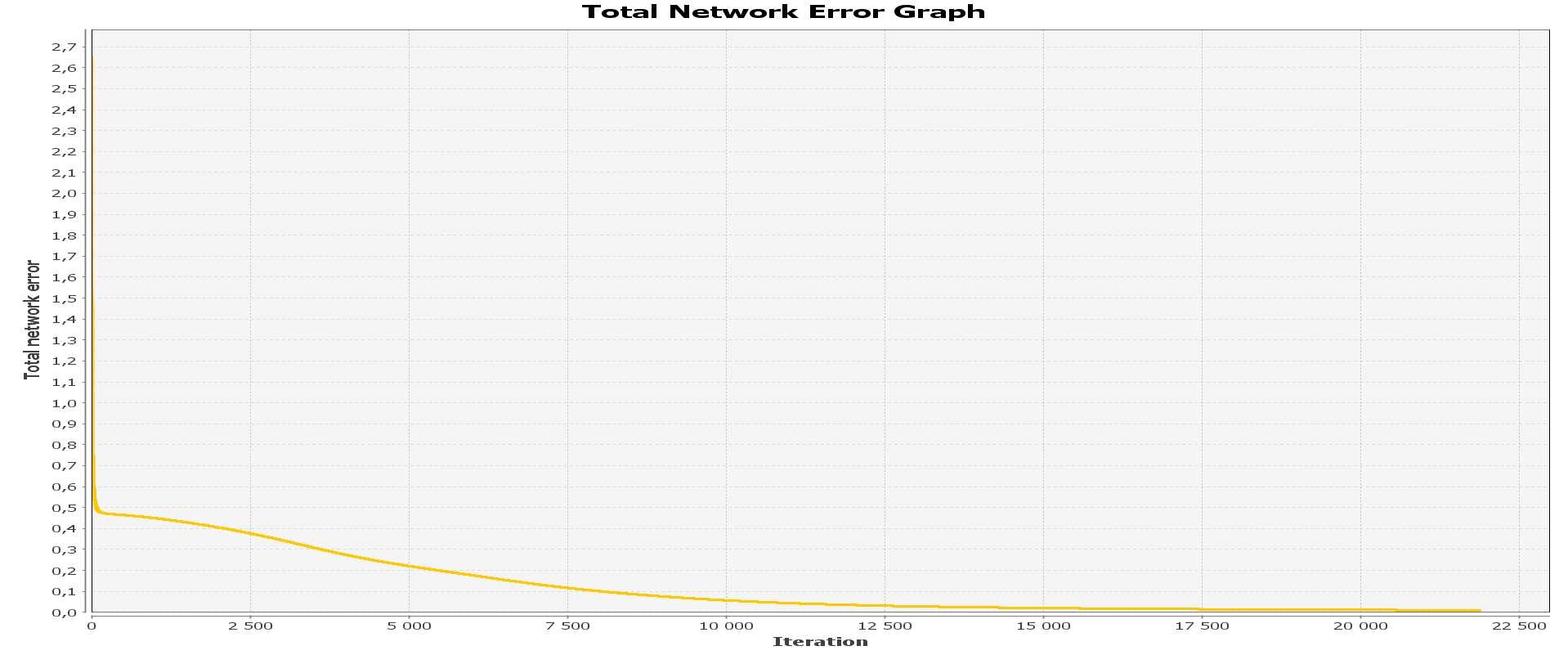
**3. Uczenie i testowanie sieci.**

Max Error = 0.01

**Współczynnik uczenia = 0.01**

Momentum = 0.2

**Ilość iteracji =** **21 886**



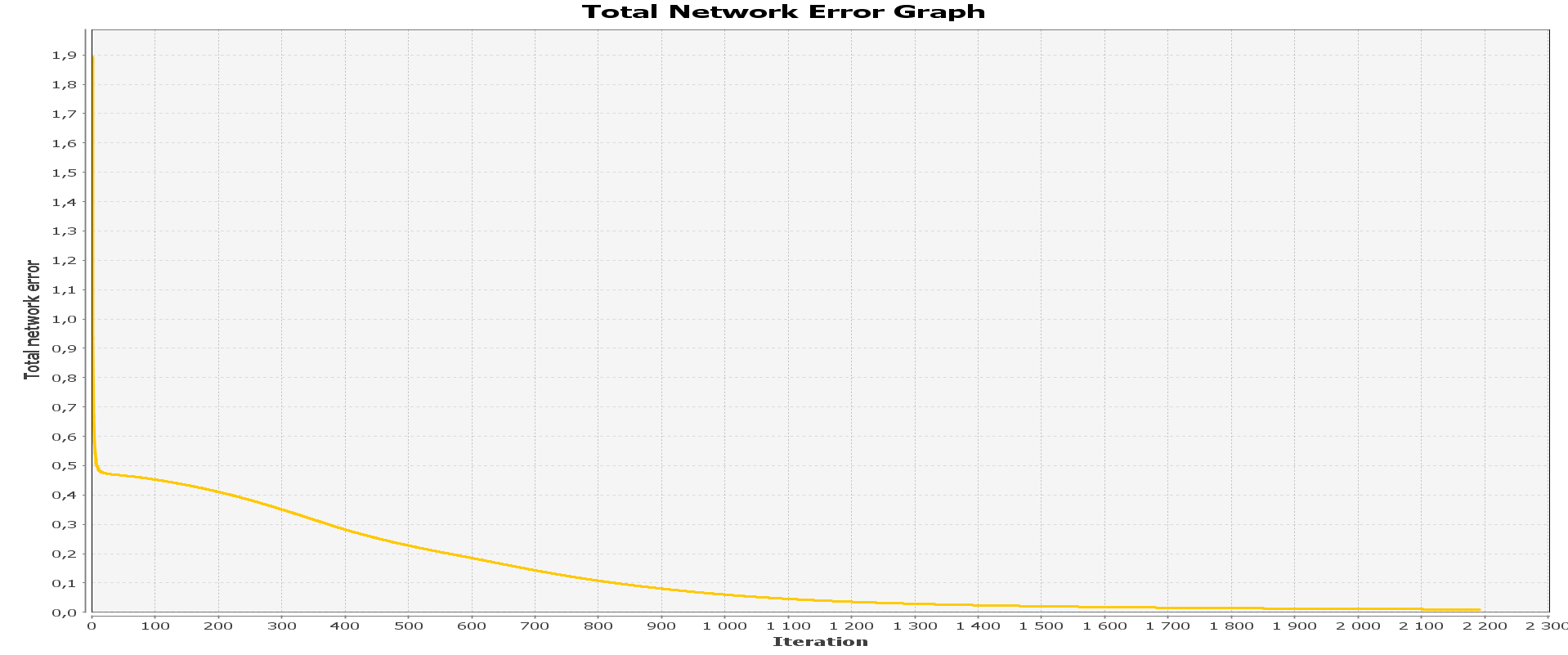
\*\*

Max Error = 0.01

**Współczynnik uczenia = 0.1**

Momentum = 0.2

**Ilość iteracji = 2 193**



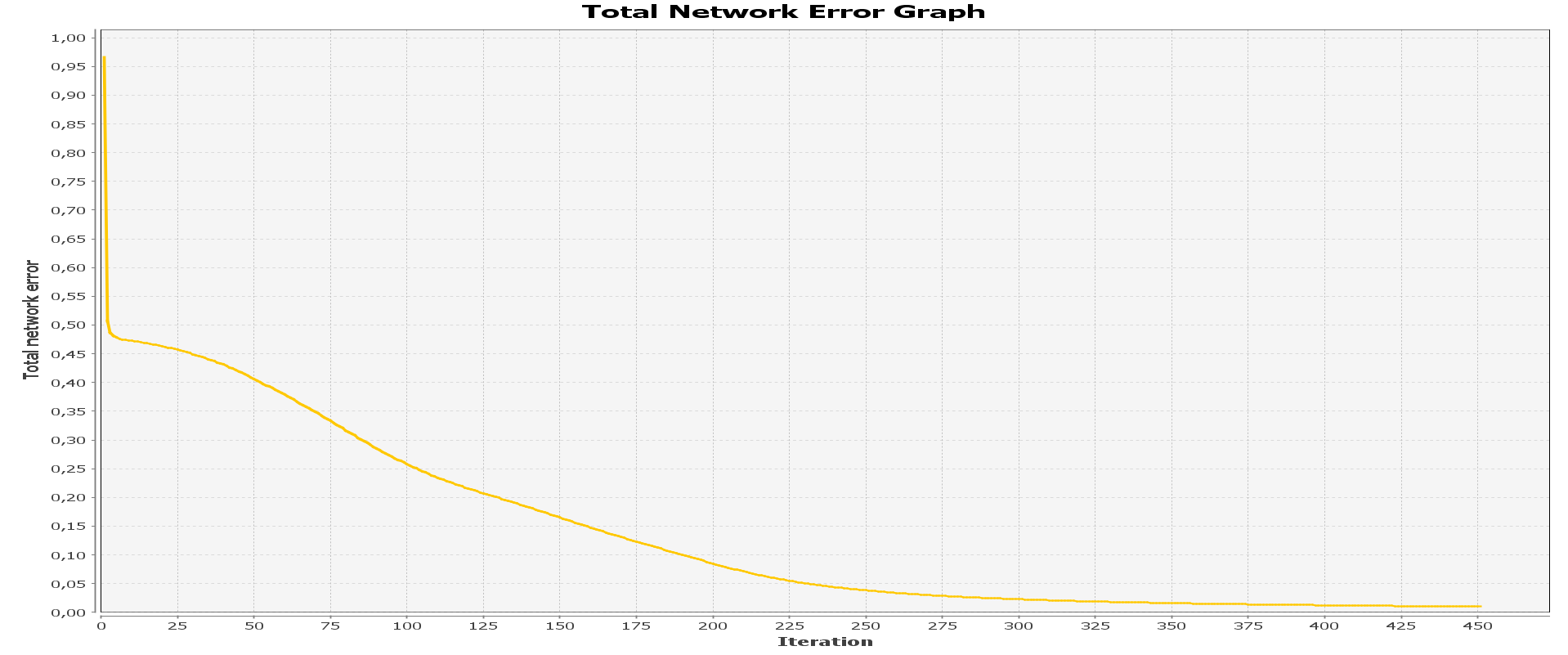
\*\*

Max Error = 0.01

**Współczynnik uczenia = 0.5**

Momentum = 0.2

**Ilość iteracji = 451**

****

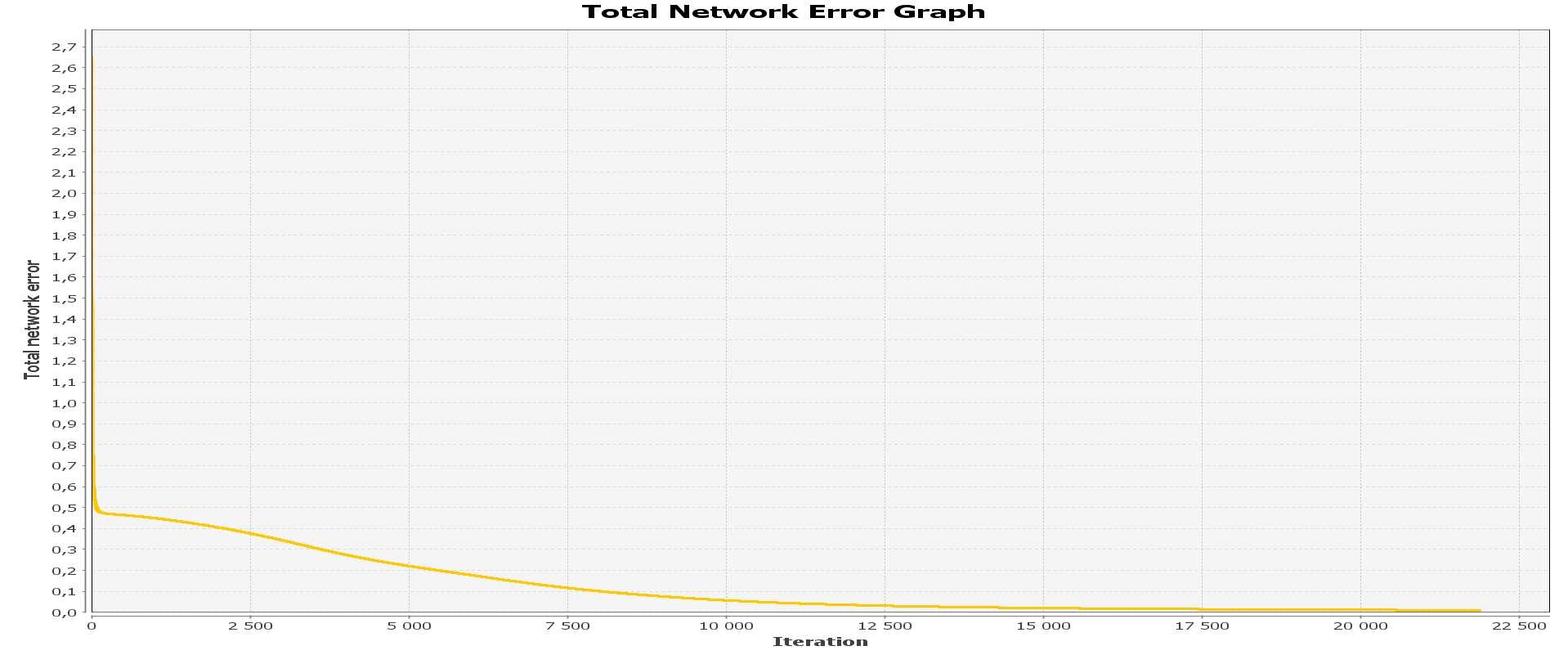
\*\*

Max Error = 0.01

**Współczynnik uczenia = 0.01**

Momentum = 0.7

**Ilość iteracji = 21 886**

****

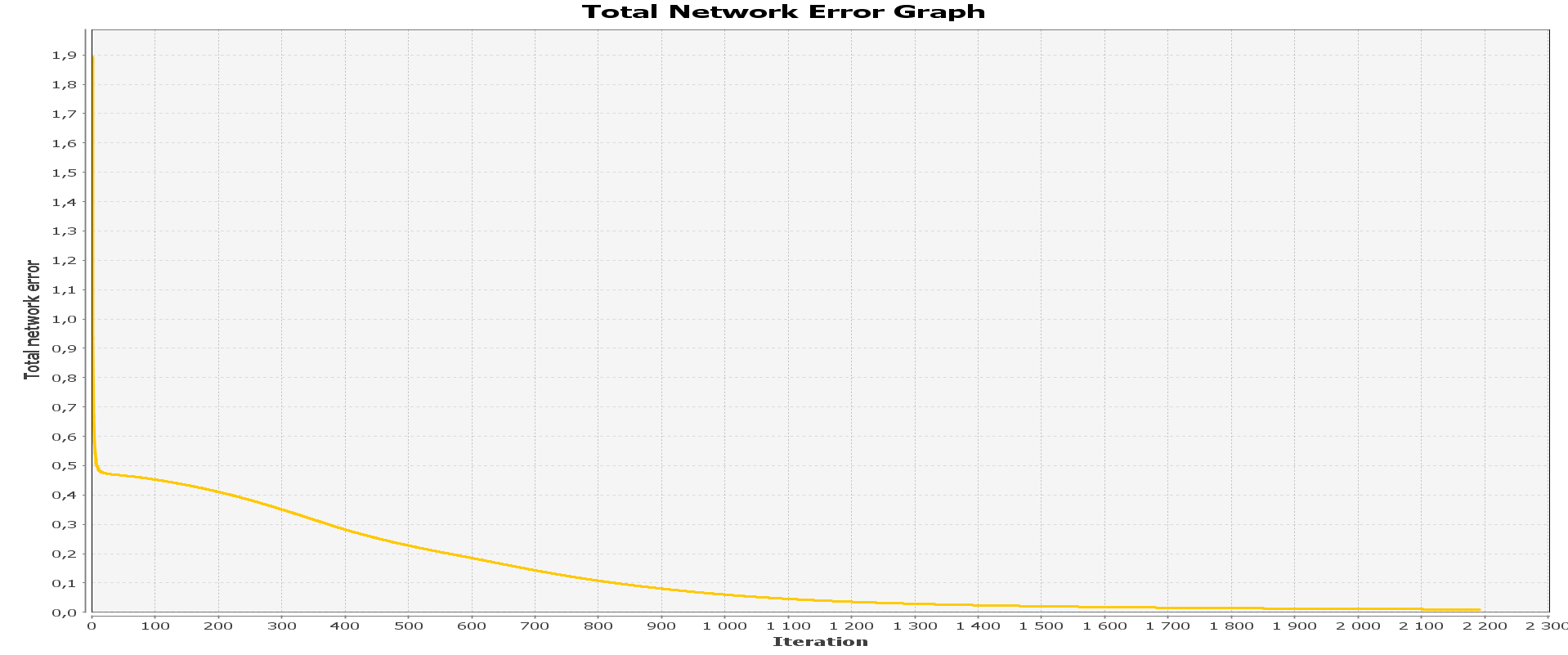
\*\*

Max Error = 0.01

**Współczynnik uczenia = 0.1**

Momentum = 0.7

**Ilość iteracji = 2 193**

****

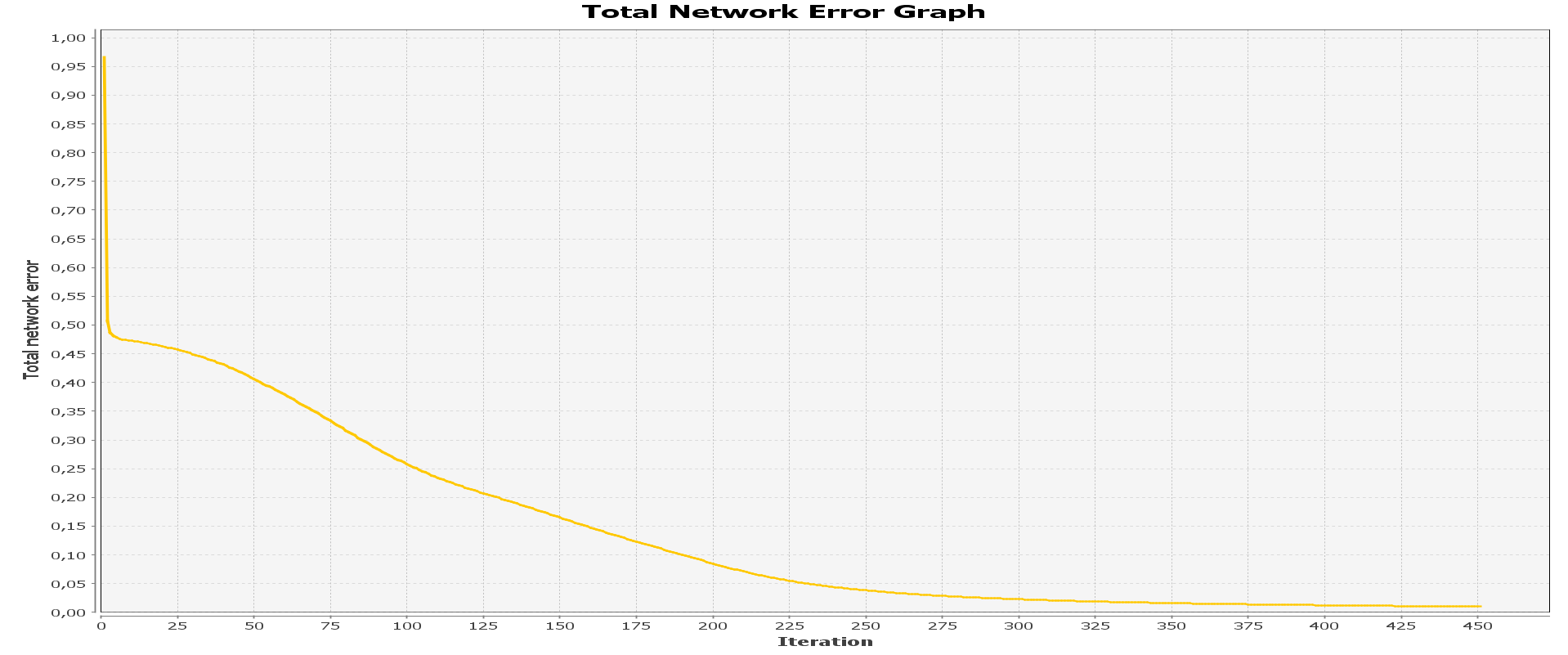
\*\*

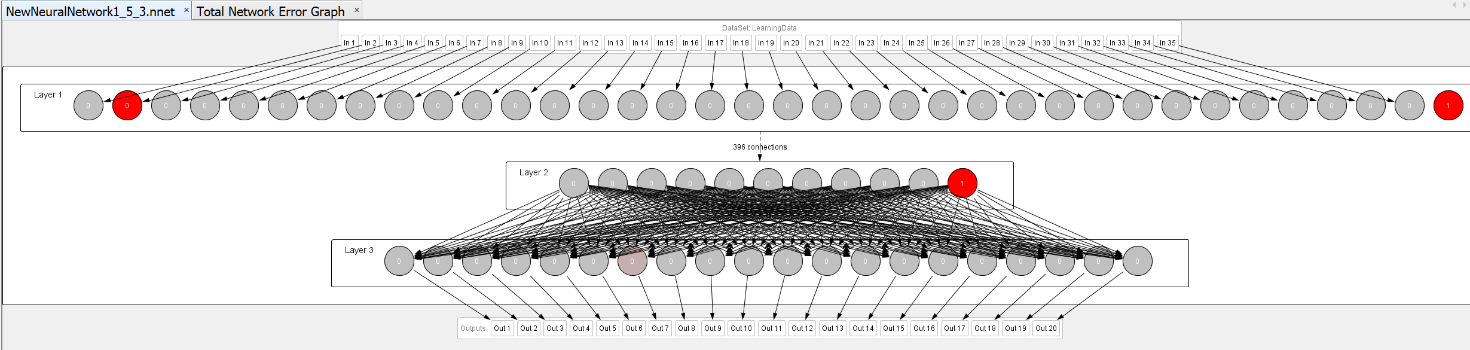
Max Error = 0.01

**Współczynnik uczenia = 0.5**

Momentum = 0.7

**Ilość iteracji = 451**

****

****

**Zestawienie wyników:**

|  |  |  |
| --- | --- | --- |
| **Współczynnik uczenia** | **Momentum** | **Liczba iteracji** |
| **0.01** | **0.2** | **21 886** |
| **0.1** | **0.2** | **2 193** |
| **0.5** | **0.2** | **451** |
| **0.01** | **0.7** | **21 886** |
| **0.1** | **0.7** | **2 193** |
| **0.5** | **0.7** | **451** |

**Obserwacje:** Parametr momentum nie wpływa na liczbę iteracji. Liczba iteracji jest tym większa im mniejszy jest współczynnik uczenia.

Największy spadek wartości błędu ma miejsce w pierwszych iteracjach. W dalszej fazie zmiany błędu nie są już tak gwałtowne jak na początku.

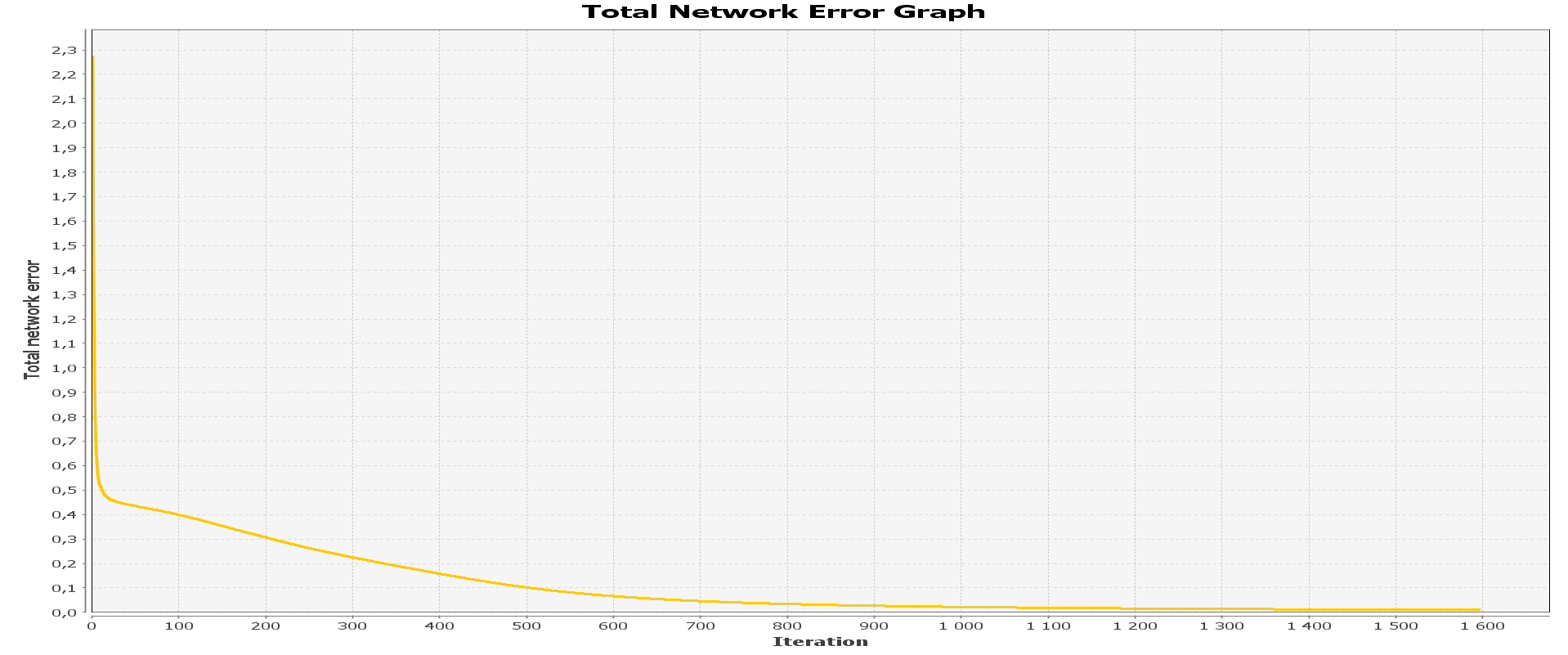
**Wykonanie testu dla 10 liter.**

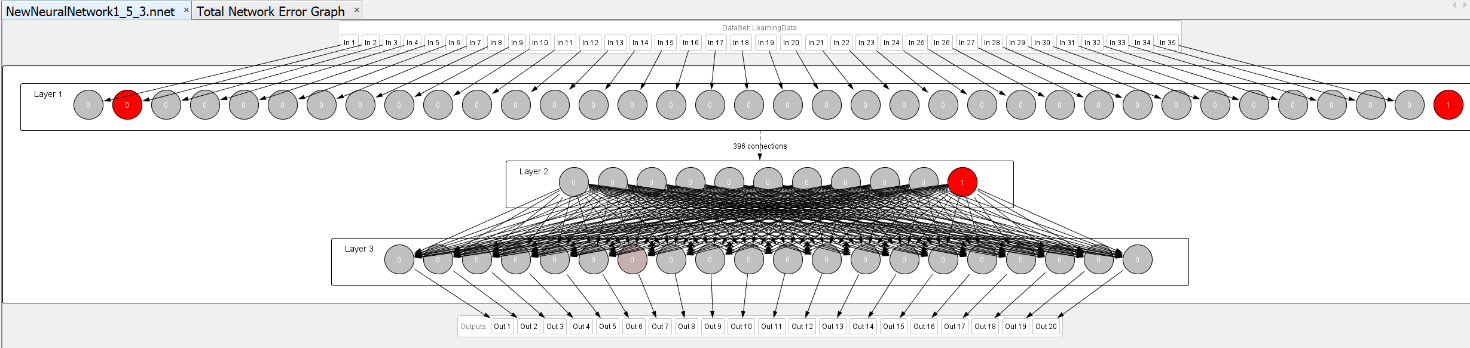
Max Error = 0.01

**Współczynnik uczenia = 0.1**

Momentum = 0.7

**Ilość iteracji = 1 597**

****

****

**4. Wnioski.**

Współczynnik uczenia odpowiada za prędkość uczenia sieci. Wraz z jego wzrostem – uczenie przebiega szybciej, jednak mniej precyzyjnie. Niższe wartości powodują uczenie wolniejsze, ale za to dokładne i precyzyjne.

Im mniejszy współczynnik uczenia, tym więcej ilości iteracji potrzeba do uzyskania poprawnego wyniku.

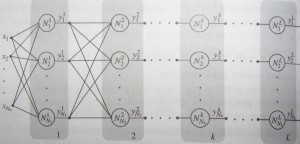
**5. Syntetyczny opis budowy sieci oraz algorytmu uczenia sieci.**

**Algorytm wstecznej propagacji błędów** polega na takiej zmianie wag sygnałów wejściowych kazdego neuronu w każdej warstwie, by wartość błędu dla kolejnych par uczących zawartych w zbiorze uczącym była jak najmniejsza. W tym celu wykorzystywana jest metoda gradientowa najszybszego spadku.  
Schemat krokowy można przedstawić następująco:

1. Inicjalizacja sieci i algorytmu
2. Obliczanie wartości wyjściowej sieci na podstawie danych
3. Obliczanie błędu sieci
4. Korekcja wag
5. Czy sieć nauczona?
   1. TAK – przejdź dalej
   2. NIE – wróć do punktu 2
6. Koniec

Przebieg algorytmu dla wszystkich elementów ciągu uczącego nazywa się **epoką**.  
Algorytm wstecznej propagacji błędów przedstawia sposób uczenia z nadzorem, lub inaczej- z nauczycielem. Określa on sposób, w jaki dobierane są współczynniki wagowe w sieciach wielowarstwowych na podstawie znanej wartości popełnionych błędów.

Zacznijmy od formalnego opisania **schematu wielowarstwowej sieci neuronowej** oraz wprowadzenia odpowiednich oznaczeń.

[](http://web.archive.org/web/20160309093826/http:/ac-it.pl/wp-content/uploads/2012/09/siec.jpg)  
*Schemat wielowarstwowej sztucznej sieci neuronowej*

Naszym zadaniem będzie stworzenie funkcji, w której w rolach zmiennych występują wszystkie wagi wielowarstwowej sieci neuronowej. Oznaczmy ją przez Q(w), gdzie woznacza wektor wszystkich wag sieci. Będziemy dążyć w trakcie uczenia sieci do znalezienia minimum funkcji Qwzgledem wektora w.

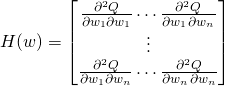
Rozwińmy zatem rozważaną przez nas funkcję w szereg Taylora w najbliższym sąsiedztwie znanego aktualnego rozwiązania w. Rozwinięcie to przedstawimy wzdłuż kierunku pw sposób następujący:

\begin{displaymath} Q(w+p)=Q(w)+[g(w)]^Tp+0,5p^TH(w)p+... \end{displaymath}

gdzie g(w)oznacza wektor gradientu, tzn.

\begin{displaymath} g(w)=[ \frac{\partial{q}}{\partial{w_1}}, \frac{\partial{q}}{\partial{w_2}},...,\frac{\partial{q}}{\partial{w_n}} ]^T \end{displaymath}

natomiast H(w)jest hesjanem, tzn. macierzą drugich pochodnych



Modyfikację wag przeprowadza się następująco:

\begin{displaymath} w(t+1)=w(t)+\eta(t)p(t) \end{displaymath}

gdzie \etajest współczynnikiem uczenia (decydującym o szybkości i precyzji osiąganej zbieżności). Modyfikację wag przeprowadza się tak długo, aż funkcja Q osiągnie minimum lub jej wartość spadnie poniżej założonego progu.  
W kolejnych krokach iteracji oczekujemy spełnienia nierówności Q(w(t+1)) < Q(w(t)). Oznaczmy ją jako **A**

### 1. Ograniczamy szereg Taylora do rozwinięcia liniowego

\begin{displaymath} Q(w+p)=Q(w)+[g(w)]^Tp \end{displaymath}

Q(w)– zależy od wag w kroku t  
Q(w+p)– zależy od wag w kroku t+1

Aby zachodziło **A** wystarczy przyjąć

\begin{displaymath} p(t)=-g(w(t)) \end{displaymath}

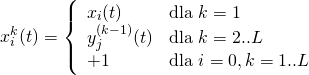
bo g(w(t))^Tp(t)<0. Wstawiająć tą zależność do wzoru na zmianę wag, otrzymujemy:

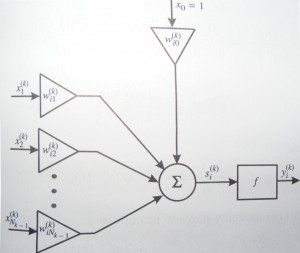
\begin{displaymath} \mathbf{w(t+1)=w(t)-\eta(t)g(w(t))} \end{displaymath}

Ta zależność określana jest w literaturze jako **reguła największego spadku**.

### 2. Definiujemy schemat neuronu, oraz jego wejścia i wyjścia

Sygnał **wejściowy** N^k_ijest powiązany z sygnałem warstwy k-1



[](http://web.archive.org/web/20160309093826/http:/ac-it.pl/wp-content/uploads/2012/09/neuron.jpg)  
*Schemat neuronu N^k_i*

Sygnał **wyjściowy** neuronu N^k_iw chwili t,t=1,2..jest określany jako

\begin{displaymath} y^k_i(t) = f(s_i^{(k)}(t)) \end{displaymath}

przy czym

\begin{displaymath} y=f(\sum_{j=0}^{N_{k-1}}w_{ij}^{(k)}*x_j^{(k)}(t)) \end{displaymath}

Sygnały wyjściowe w warstwie Lto sygnały wyjściowe całej sieci. Są one porównywane z tzw. *sygnałami wzorcowymi sieci*

\begin{displaymath} d^L_1(t),d^L_2(t),...d^L_{N_l}(t) \end{displaymath}

### 3. Wyznaczamy błąd na wyjściu sieci

Błąd na wyjściu sieci Qzdefiniujemy następująco:

\begin{displaymath} Q(t)=\sum_{i=1}^{N_l}(d_i^{(L)}(t) - y_i^{(L)}(t))^2 \end{displaymath}

Korzystając z powyższej zależności, oraz reguły największego spadku, otrzymujemy poniższy wzór (ozn. **B**).

\begin{displaymath} w^k_{ij}(t+1)=w^k_{ij}(t)-\eta(t)\frac{\partial Q(t)}{\partial w^k_{ij}(t)} \end{displaymath}

Prawdą jest, iż

\begin{displaymath} \frac{\partial Q(t)}{\partial w^k_{ij}(t)}  = \frac{\partial Q(t)}{\partial s^k_{i}(t)} * \frac{\partial s_i^k(t)}{\partial w^k_{ij}(t)} =  \frac{\partial Q(t)}{\partial s^k_{i}(t)}*x_j^k(t) \end{displaymath}

Oznaczając

\begin{displaymath} \delta_i^k(t)=-\frac{1}{2}\frac{\partial Q(t)}{\partial s^k_{i}(t)}*x_j^k(t) \end{displaymath}

otrzymujemy równość

\begin{displaymath} \frac{\partial Q(t)}{\partial w^k_{ij}(t)} = -2\delta_i^k(t)x_j^k(t) \end{displaymath}

a zatem algorytm **B** możemy napisać następująco:

\begin{displaymath} \mathbf{w^k_{ij}(t+1)=w^k_{ij}(t)+2\eta\delta_i^k(t)x_j^k(t)} \end{displaymath}

### 4. Definiujemy sposoby wyznaczania wartości \delta_i^{(k)}

#### 4.1 Dla warstwy ostatniej:

\begin{displaymath} \mathbf{\delta_i^{(L)}(t)=Q_i^{(L)}(t)f'(s_i^{(L)}(t))} \end{displaymath}

#### 4.2 Dla dowolnej warstwy k<L:

\begin{displaymath} \delta_i^{(L)}(t)=f'(s_i^{(k)}(t))\sum_{m=1}^{N_{(k+1)}}\delta_m^{(k+1)}(t)w_{mi}^{(k+1)}(t) \end{displaymath}

Alternatywnie, po zdefiniowaniu błędu w warstwie k-tej (z wyjątkiem ostatniej) dla i-tego neuronu jako:

\begin{displaymath} \epsilon_i^{(k)}(t)=\sum_{m=1}^{N_{(k+1)}}\delta_m^{(k+1)}(t)w_{mi}^{(k+1)}(t) \end{displaymath}

otrzymujemy

\begin{displaymath} \mathbf{\delta_i^{(L)}(t)=\epsilon_i^{(k)}(t)f'(s_i^{(k)}(t))} \end{displaymath}

Powyżej został przedstawiony szereg zależności matematycznych, opisujących sposób uczenia wielowarstwowej sieci neuronowej.

Schemat działania:

1. Działanie algorytmu zaczyna się od podania wzorca uczącego na wejście sieci
   1. Najpierw zostaje on przetworzony przez neurony warstwy pierwszej. Przez pojęcie *przetwarzania* rozumiemy wyznaczenie sygnału wyjściowego dla każdego neuronu w danej warstwie
   2. Otrzymane w ten sposób sygnały stają się wejściami dla neuronów warstwy następnej
   3. Cykl ten powtarza się, tzn. ponownie wyznaczamy wartości sygnałów na wyjściach neuronów kolejnej warstwy i przekazujemy je dalej, aż do warstwy ostatniej
2. Znając sygnał wyjściowy warstwy ostatniej oraz sygnał wzorcowy z ciągu uczącego, możemy obliczyć błąd na wyjściu sieci
   1. Błąd na wyjściu sieci definiuje się jako różnica sygnału wzorcowego i wyjściowego
   2. Wykorzystując regułę delta, możemy modyfikować wagi neuronów ostatniej warstwy
3. Następnie, błąd wyjściowy jest propagowany do tyłu (od warstwy wyjściowej do wejściowej) zgodnie z połączeniami neuronów między warstwami i z uwzględnieniem ich funkcji aktywacji.