**Urszula Król,**

**grupa projektowa 2**

**IS WIMiIP**

**Sprawozdanie nr 3**

**1. Temat ćwiczenia:**

Budowa i działanie sieci wielowarstwowej.

**2. Cel ćwiczenia:**

Celem ćwiczenia jest poznanie budowy i działania wielowarstwowych sieci neuronowych

poprzez uczenie z użyciem algorytmu wstecznej propagacji błędu rozpoznawania

konkretnych liter alfabetu.

**3. Zadania wykonane w ramach ćwiczenia:**

a) Wygenerowano dane uczące zawierające 20 dużych liter dowolnie wybranego alfabetu w postaci dwuwymiarowej tablicy 5x7 pikseli dla jednej litery oraz stworzono dane testowe.

b) Przygotowano wielowarstwową sieć z wykorzystaniem gotowego narzędzia Neuroph Studio.

c) Przeprowadzono uczenie sieci dla różnych współczynników uczenia i bezwładności.

d) Przeprowadzono testowanie sieci.

**4. ​Syntetyczny opis budowy oraz wykorzystanych sieci i algorytmów uczenia:**

Algorytm wstecznej propagacji błędu

Za pomocą metody obliczeniowej zwanej wsteczną propagacją błędu możemy wytrenować wagi, czyli znaleźć ich optymalny zestaw. Ta metoda umożliwia modyfikację wag w sieci o architekturze warstwowej we wszystkich jej warstwach.

Ogólny schemat procesu wygląda następująco:

1. Ustalamy topologię sieci, czyli liczbę warstw, liczbę neuronów w warstwach.
2. Inicjujemy wagi losowo.
3. Dla danego wektora uczącego obliczamy odpowiedź sieci warstwa po warstwie.
4. Każdy neuron wyjściowy oblicza swój błąd, oparty na różnicy pomiędzy obliczoną odpowiedzią *y* oraz poprawną odpowiedzią *t*.
5. Błędy propagowane są do wcześniejszych warstw.
6. Każdy neuron, także w warstwach ukrytych, modyfikuje wagi na podstawie wartości błędu i wielkości przetwarzanych w tym kroku sygnałów.
7. Powtarzamy od punktu 3. dla kolejnych wektorów uczących. Gdy wszystkie wektory zostaną użyte, losowo zmieniamy ich kolejność i zaczynamy wykorzystywać powtórnie.
8. Zatrzymujemy się, gdy średni błąd na danych treningowych przestanie maleć. Możemy też co jakiś czas testować sieć na specjalnej puli nieużywanych do treningu próbek testowych i kończyć trenowanie, gdy błąd przestanie maleć.

Wzór na zmianę konkretnej wagi wygląda następująco:

http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad3/back_wzor2.gif

Powyższe wyprowadzenie jest prawidłowe tylko dla ostatniej warstwy, kiedy wiemy, jaka powinna być prawidłowa wartość wyjścia. Dla wcześniejszych warstw takiej informacji bezpośrednio nie mamy. Zamiast tego błąd ten będziemy przybliżać, przenosząc go czyli propagując z kolejnych warstw.

Oznaczenia:

*w* - waga wejścia neuronu,

*z* - sygnał wchodzący do neuronu danym wejściem,

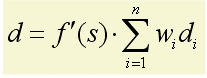
*d* - współczynnik błędu obliczony dla danego neuronu,

*s* - wartość wzbudzenia (suma wartości wejściowych pomnożonych przez wagi) dla danego neuronu

Pomocniczy współczynnik błędu *d* zdefiniujemy dla ostatniej warstwy jako:

http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad3/back_wzor3.gif

a dla pozostałych warstw:



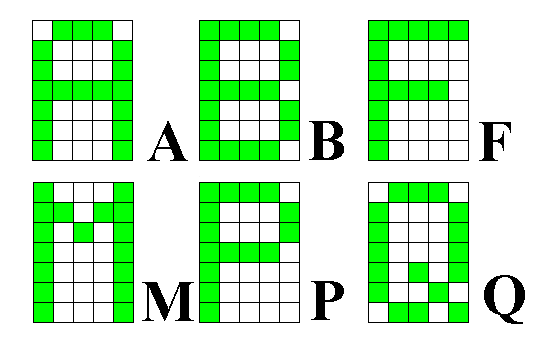
Zmiana wag połączeń następuje jednocześnie we wszystkich warstwach, po fazie propagacji błędu i odbywa się według wzoru:

http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad3/back_wzor5.gif

Powyższy wzór jest uogólnieniem wzoru wyprowadzonego wcześniej dla ostatniej warstwy.

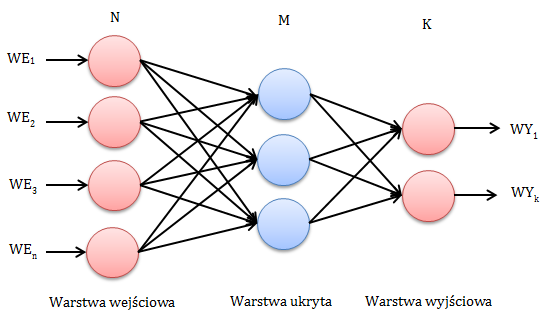
Format danych

Matryca 5x7 przykładowych liter:



Pola zaznaczone na zielono odpowiadają wartości 1, a pola w kolorze białym odpowiadają wartości 0 po konwersji zero-jedynkowej.

Schemat budowy wielowarstwowej sieci neuronowej:



Do stworzenia sieci wielowarstwowej skorzystano z narzędzia Neuroph Studio. Neuroph jest strukturą sieci neuronowej Javy, która pozwala tworzyć popularne architektury sieci neuronowych.

Stworzona sieć posiada 3 warstwy. Warstwa wejściowa ma 35 neuronów wejściowych, warstwa ukryta składa się z 10 neuronów, natomiast ostatnia warstwa, czyli warstwa wyjściowa posiada 20 neuronów.

**5. ​Zestawienie otrzymanych wyników:**

Tabela 1. Wykonane próby:

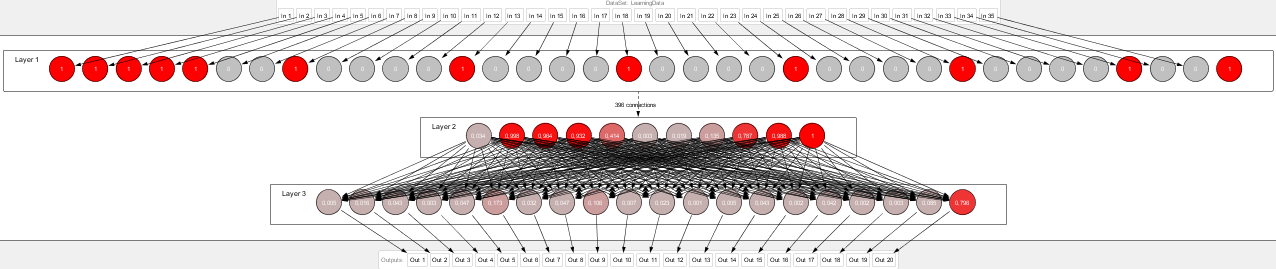
|  |  |  |  |
| --- | --- | --- | --- |
| Lp. | Learning Rate | Momentum | Error |
| 1 | 0.1 | 0.7 | 0.01 |
| 2 | 0.05 | 0.7 | 0.01 |
| 3 | 0.2 | 0.7 | 0.01 |
| 4 | 0.1 | 0.7 | 0.1 |
| 5 | 0.05 | 0.7 | 0.1 |
| 6 | 0.2 | 0.7 | 0.1 |
| 7 | 0.1 | 0.2 | 0.1 |
| 8 | 0.1 | 0.5 | 0.1 |
| 9 | 0.3 | 0.7 | 0.01 |
| 10 | 0.4 | 0.7 | 0.01 |
| 11 | 0.5 | 0.7 | 0.01 |

Tabela 2. Zestawienie prób ze stałą wartością Momentum i Error i różnymi wartościami Learning Rate:

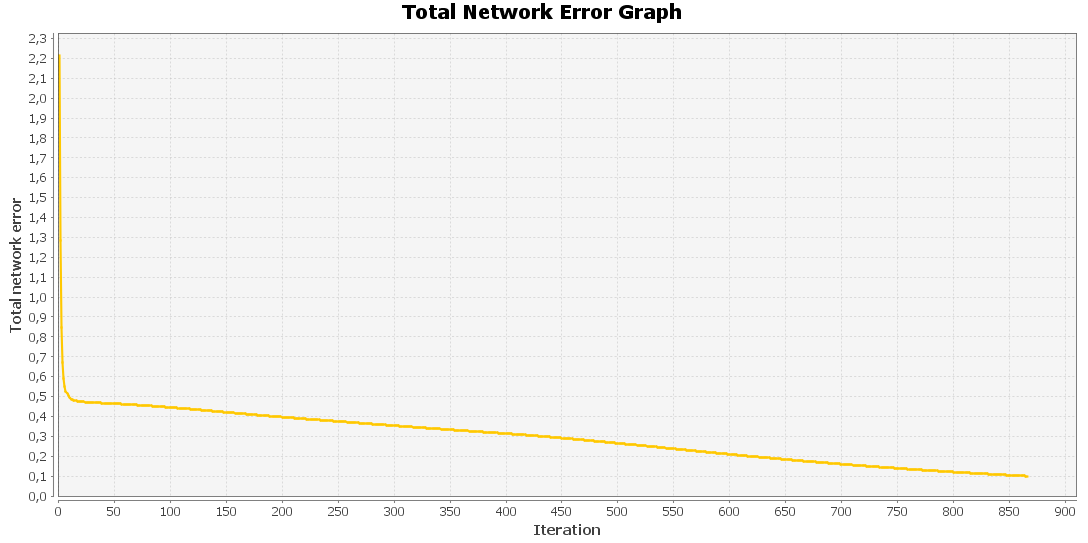
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Lp. | Learning Rate | Momentum | Error | Total Error | Iterations |
| 1 | 0.1 | 0.7 | 0.01 | 0.0010313272341431346 | 2400 |
| 2 | 0.05 | 0.7 | 0.01 | 0.0010443964892938471 | 4900 |
| 3 | 0.2 | 0.7 | 0.01 | 0.0010103371025861534 | 1200 |
| 9 | 0.3 | 0.7 | 0.01 | 0.0010002224036313376 | 790 |
| 10 | 0.4 | 0.7 | 0.01 | 0.0009980287526368515 | 590 |
| 11 | 0.5 | 0.7 | 0.01 | 0.0010001333925108934 | 470 |

Wyniki wykonanych testów i wykresy dla każdej próby znajdują się w pliku otrzymane\_wyniki.xlsx w repozytorium.

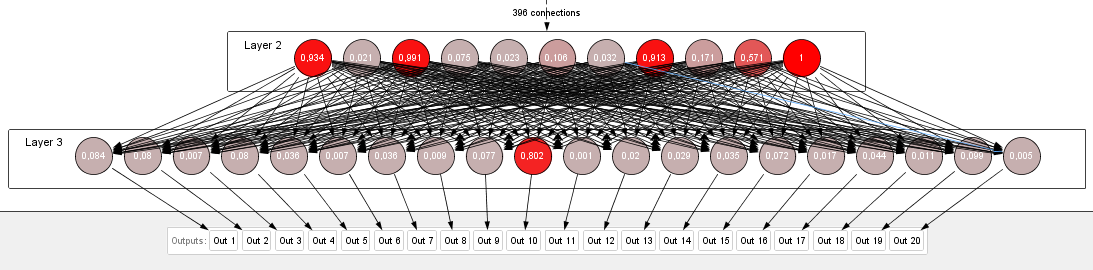
Przykład dla learning\_rate=0.1, momentum=0.7, error=0.1



Wykres zmian wartości błędów w zależności od iteracji dla powyższego przykładu:



Przykład po wykonaniu testu:



**6.​ Analiza i dyskusja otrzymanych wyników:**

- Analiza wykresu zmian wartości błędów w zależności od iteracji dla każdej z prób pozwala zauważyć, że największy spadek wartości błędu ma miejsce w pierwszych iteracjach. W dalszej fazie zmiany błędu nie są już tak gwałtowne jak na początku.

- Wykres ten ukazuje również zależność liczby iteracji od współczynnika uczenia: im mniejszy współczynnik uczenia tym większa liczba iteracji, co potwierdza analiza prób 1, 2 i 3, gdzie dla najmniejszego współczynnika w próbie numer 2 liczba iteracji wyniosła prawie 5000.

- Porównanie prób 1-3 i 4-6 różniących się wielkością dopuszczonego błędu pozwala zauważyć, że próby 4-6 gdzie dopuszczalny błąd był większy wymagały mniej iteracji, ale jednocześnie całkowity błąd w tych próbach był 9-krotnie większy niż w przypadku prób 1-3. Wyniki w próbach 4-6 nie były tak poprawne i jednoznaczne jak w próbach 1-3.

- Analiza prób w tabeli 2, gdzie stałe były wartości momentum i error, a jedynie wartości learning rate się zmieniały, ukazuje, że im mniejszy współczynnik uczenia tym więcej iteracji oraz wskazuje na to, że dobór odpowiedniego współczynnika uczenia ma decydujący wpływ na jakość działania sieci – dla współczynnika równego 0.4 wartość całkowitego błędu była najmniejsza przy jednoczesnej niewielkiej liczbie iteracji, czyli właśnie ta wartość współczynnika jest optymalna.

- Próby numer 4, 7, 8 różniące się jedynie wartością momentum wskazują, że nie ma ono wpływu na uczenie sieci, gdyż uzyskane wyniki w tych próbach były takie same.

**7. Wnioski:**

- Współczynnik uczenia ma decydujący wpływ na szybkość procesu uczenia sieci wielowarstwowej: Im mniejszy współczynnik uczenia tym większa liczba iteracji, a co za tym idzie, wydłuża się czas trwania całego procesu.

- Prawidłowy dobór współczynnika uczenia jest głównym czynnikiem wpływającym na jakość i poprawność uczenia: gdy wartość ta jest zbyt mała zwiększa się jedynie liczba iteracji, bez wpływu na dokładność wyników.

- Wielkość dopuszczalnego błędu wpływa na poprawność wyników: im większy jest dopuszczalny błąd tym mniej dokładne otrzymujemy rezultaty.

- Proces propagacji błędu powtarzany jest dla każdego przypadku uczącego, co jest dość kosztowne obliczeniowo - jeśli sieć jest rozbudowana i przypadków uczących jest dużo, cały proces może zająć bardzo dużo czasu.