Michał Radziejowski – Scenariusz 2 SPRAWOZDANIE

Budowa i działanie sieci neuronowej

Cel ćwiczenia

Celem ćwiczenia jest poznanie budowy i działania sieci neuronowych oraz uczenie rozpoznawania wielkości liter.

Algorytm uczenia sieci – algorytm propagacji wstecznej

Algorytm wstecznej propagacji błędów, polega na takim dobraniu wag sygnałów wejściowych każdego neuronu w każdej warstwie, aby wartość błędu dla kolejnych par uczących zawartych w zbiorze uczącym była jak najmniejsza. Schemat krokowy algorytmu przedstawia się w następujący sposób:

1. Ustalamy topologię sieci, tzn. 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 modyfikuje wagi na podstawie wartości błędu i wielkości przetwarzanych w tym kroku sygnałów.
7. Powtarzamy od punktu 3. dla kolejnych wektorów uczących.
8. Zatrzymujemy się, gdy średni błąd na danych treningowych zostanie przyjęty jako optymalny.

Algorytm Sieci

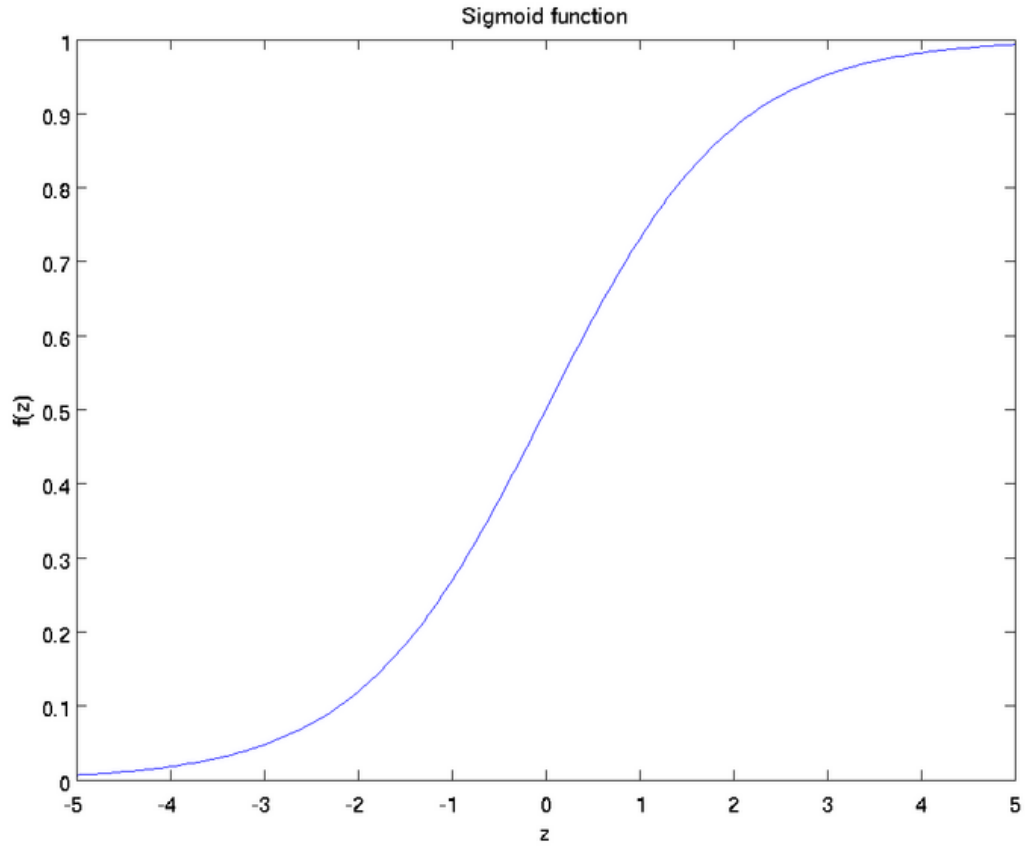
Do stworzenia sieci została użyta biblioteka PyBrain3. Podstawową architekturą była trójwarstwowa sieć stworzona z 35 neuronów wejścia opisywanych funkcją liniową. Następnie znajduje się warstwa ukryta, której wielkość jest zmienna (3 lub 5 neuronów), opisywana funkcją sigmoidalną lub tangens hiperboliczny (podane poniżej), oraz warstwa wyjścia złożona z jednego neurona funkcji liniowej.

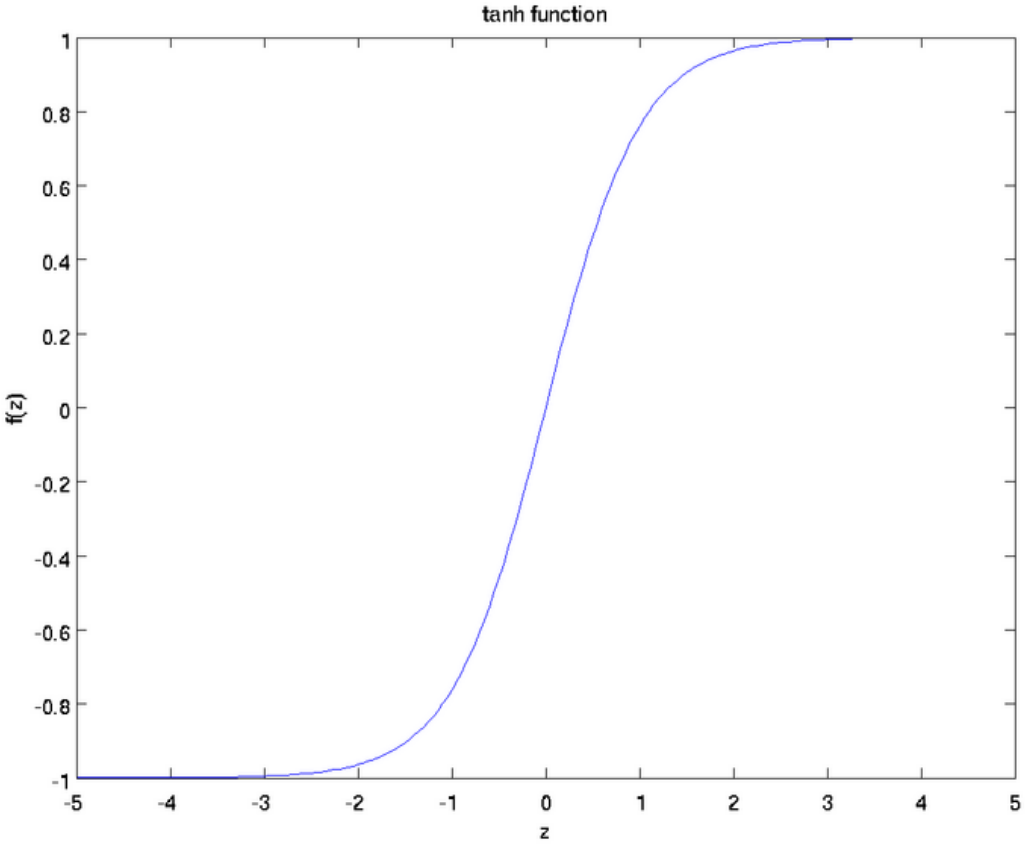
Funkcja sigmoidalna

$$f(z) = \frac{1}{1 + \exp(-z)}$$

Funkcja tangensa hiperbolicznego (tanh)

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$


Sigmoid function
f(z)
z
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-5
-4
-3
-2
-1
0
1
2
3
4
5


tanh function
f(z)
z
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-5
-4
-3
-2
-1
0
1
2
3
4
5

Przykłady działania programu

Program po jego wywołaniu, wypisuje ilość danych treningowych, a następnie przystępuje do treningu sieci według wcześniej zadeklarowanego algorytmu. Z każdą epoką, wypisuje na ekran całkowity błąd uczenia, dzięki któremu jesteśmy w stanie ocenić czy sieć się dobrze zachowuje i ćwiczy na zadanym secie danych.

```
Number of training patterns:  20
Total error: 0.204931585302
Total error: 0.198745991941
Total error: 0.193448919733
Total error: 0.186803446221
Total error: 0.18230086249
Total error: 0.175755213379
Total error: 0.169423421207
```

Po dokonaniu treningu na zadanej wcześniej ilości epok, program zwraca błąd po ostatniej epoce. Można przejść do testowania sieci.

```
Total error: 0.00010808301858
Total error: 0.000108174190322
Total error: 0.000107569070588
Total error: 0.000107786525316
```

Program w pętli wypisuje na ekran literki z wcześniej zadeklarowanej tabeli. Na podstawie danych testowych porównuje wartość output w określaniu wielkości litery, z wcześniej zaimplementowaną na sztywno zmienną porównawczą. Jeżeli wartość output jest większa od zmiennej porównawczej, litera jest określana jako duża.

```
Dla literki F output wynosi
1.01821807642
Literka F jest dużą literą

Dla literki G output wynosi
1.0007430194
Literka G jest dużą literą
```

```
Dla literki f output wynosi
0.0390348605785
Literka f jest małą literą

Dla literki h output wynosi
0.00979638086007
Literka h jest małą literą
```

Wyniki

Wnioski