Mateusz Nogieć

Scenariusz 3-sprawozdanie

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

Opis budowy wykorzystanych sieci i algorytmów uczenia.

Do wykonania projektu wykorzystałem 35 neutronów w pierwszej (wejściowej) warstwie, 16 ukrytych, oraz 20 w warstwie wyściowej. Sieć neuronowa jest typu MLP, a funkcja aktywacji każdego z neuronów jest sigmoidalna.

Poprawne obliczanie błędów popełniane przez neurony warstw ukrytych, których wyjścia nie są bezpośrednio porównywane z oczekiwanymi wartościami zapewnia algorytm wstecznej propagacji błędów. Polega on na swego rodzaju 'wytrenowaniu' wag, tj. znalezieniu optymalnych wartości. Początkowo dla losowych, obliczana jest odpowiedź sieci. Każdy neuron wyjściowy wyznacza bład między oczekiwanym outputem, a odpowiedzią neutronu. Następnie błędy propagowane są do wcześniejszych warstw. Każdy neuron (również w warstwach ukrytych) modyfikuje wagi na podstawie wartości błędu i wielkosci przetwarzanych w tym kroku sygnałów. Dla kolejnych wektorów uczących powtarzamy algorytm od obliczania odpowiedzi sieci(warstwa po warstwie). Gdy wszystkie wektory zostaną użyte, losowo zmieniamy ich kolejność i zaczynamy wykorzystywać powtórnie. 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.

Współczynnik bezwładności (momentum)-wpływa na 'płynność' zmiany wag, co pozwala przyspieszyć uczenie sieci.

Opis danych uczących oraz testujących.

W pliku z danymi uczącymi znajduje się 20 dużych liter. Wektor każdej litery kończy oczekiwany output. Jedna litera to matryca zer i jedynek 5x7.

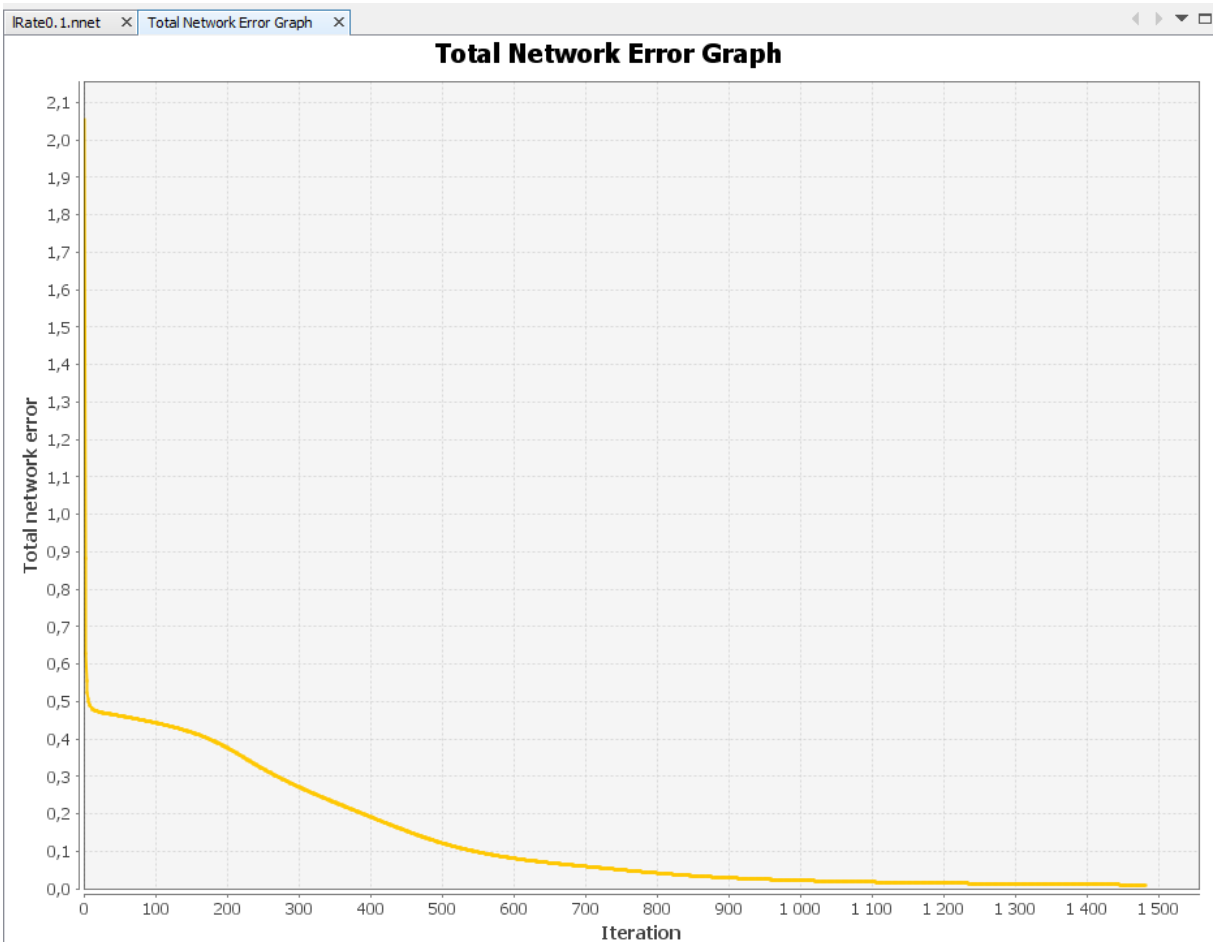
Plik z danymi testującymi to zniekształcona litera, która znajduje się w danych uczących.

Dla każdej sieci wykonałem uczenie dla następujących **współczynników uczenia**: 0.1, 0.5, 0.9 i współczynniku bezwładnowści równym 0.7, a następnie przy wartości wpółczynnika uczenia równej 0.9 przeprowadziłem testy mające na celu sprawdzić wpływ **współczynnika bezwładnośc**i na uczenie sieci. Wartości momentum, jakich użyłem, to 0.1, 0.5, 0.9 .

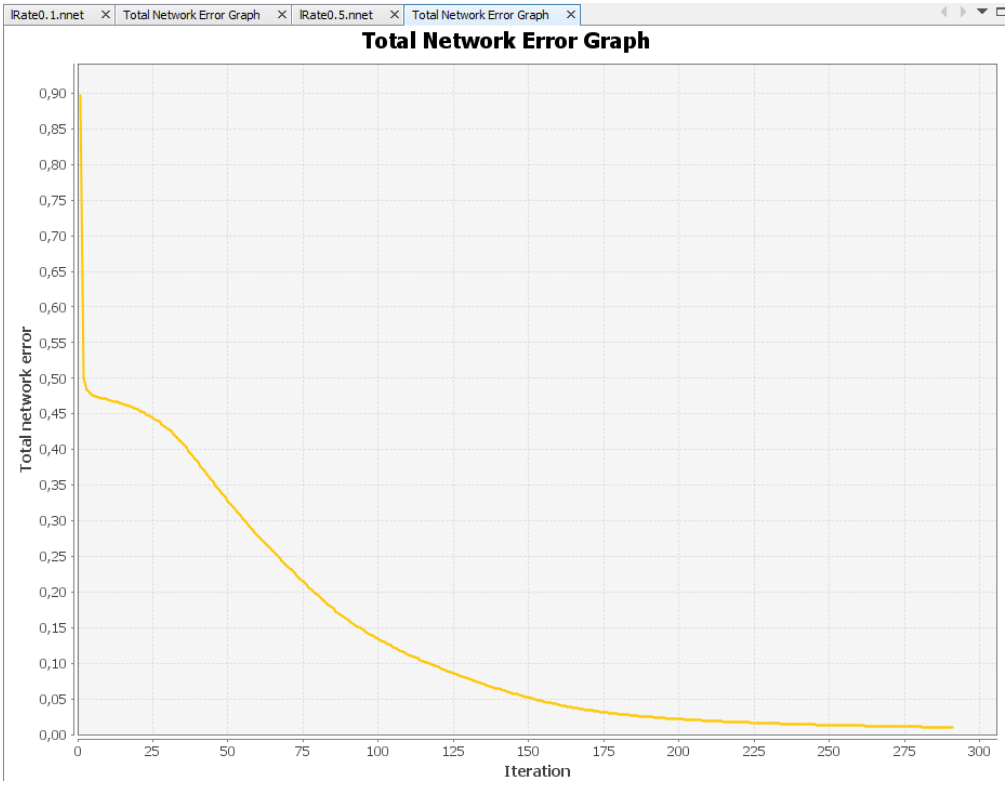
Uczenie oraz testowanie sieci

Stałe momentum=0.7, learning rate kolejno

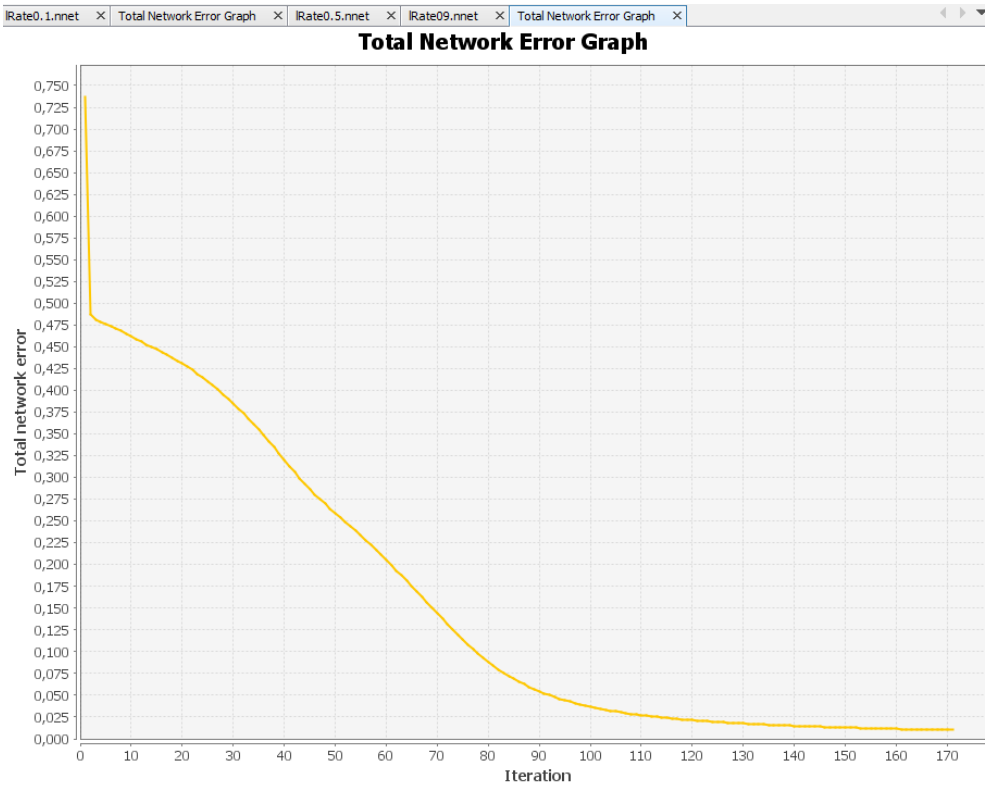
0.1



0.5

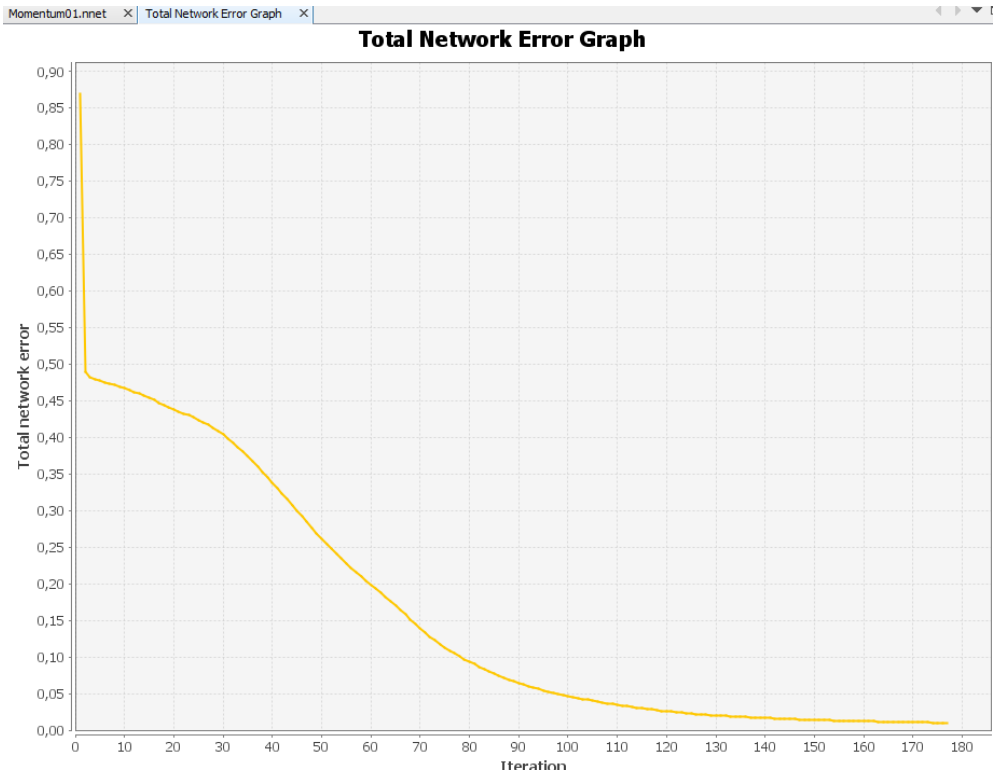

lRate0.1.nnet
Total Network Error Graph
lRate0.5.nnet
Total Network Error Graph
Total Network Error Graph
Total network error
0,90
0,85
0,80
0,75
0,70
0,65
0,60
0,55
0,50
0,45
0,40
0,35
0,30
0,25
0,20
0,15
0,10
0,05
0,00
0
25
50
75
100
125
150
175
200
225
250
275
300
Iteration

0.9

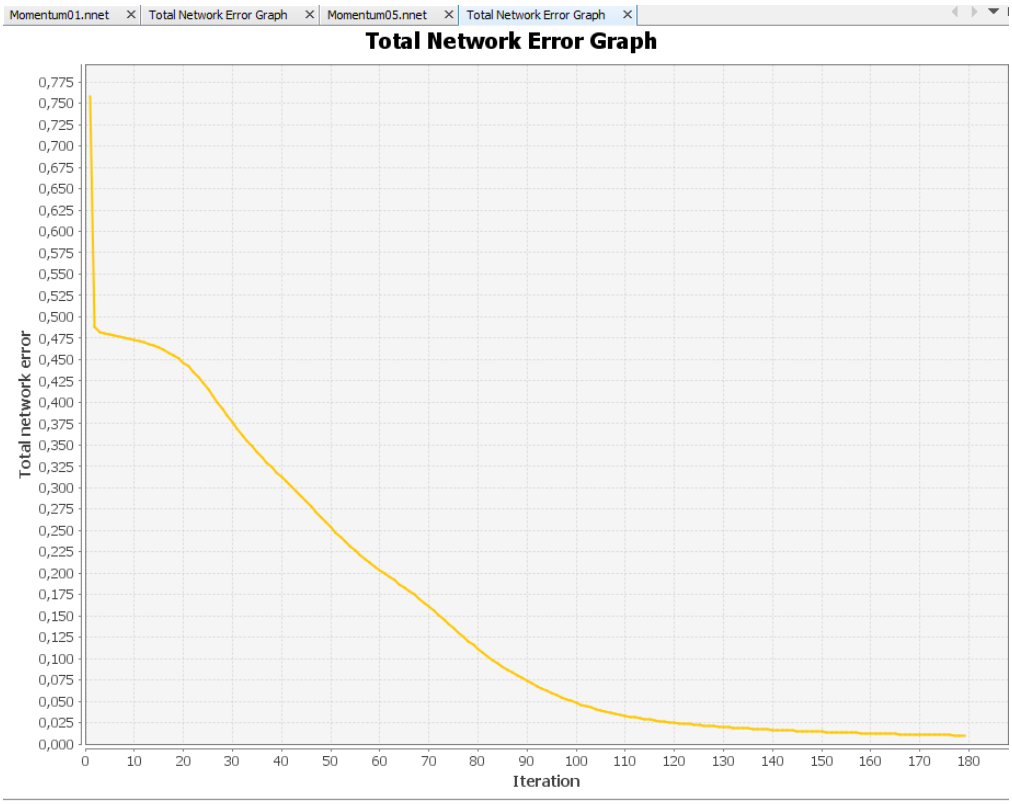

lRate0.1.nnet
Total Network Error Graph
lRate0.5.nnet
lRate09.nnet
Total Network Error Graph
Total Network Error Graph
Total network error
0,750
0,725
0,700
0,675
0,650
0,625
0,600
0,575
0,550
0,525
0,500
0,475
0,450
0,425
0,400
0,375
0,350
0,325
0,300
0,275
0,250
0,225
0,200
0,175
0,150
0,125
0,100
0,075
0,050
0,025
0,000
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
Iteration

Stały współczynnik nauczania, zmienny moment bezwładności

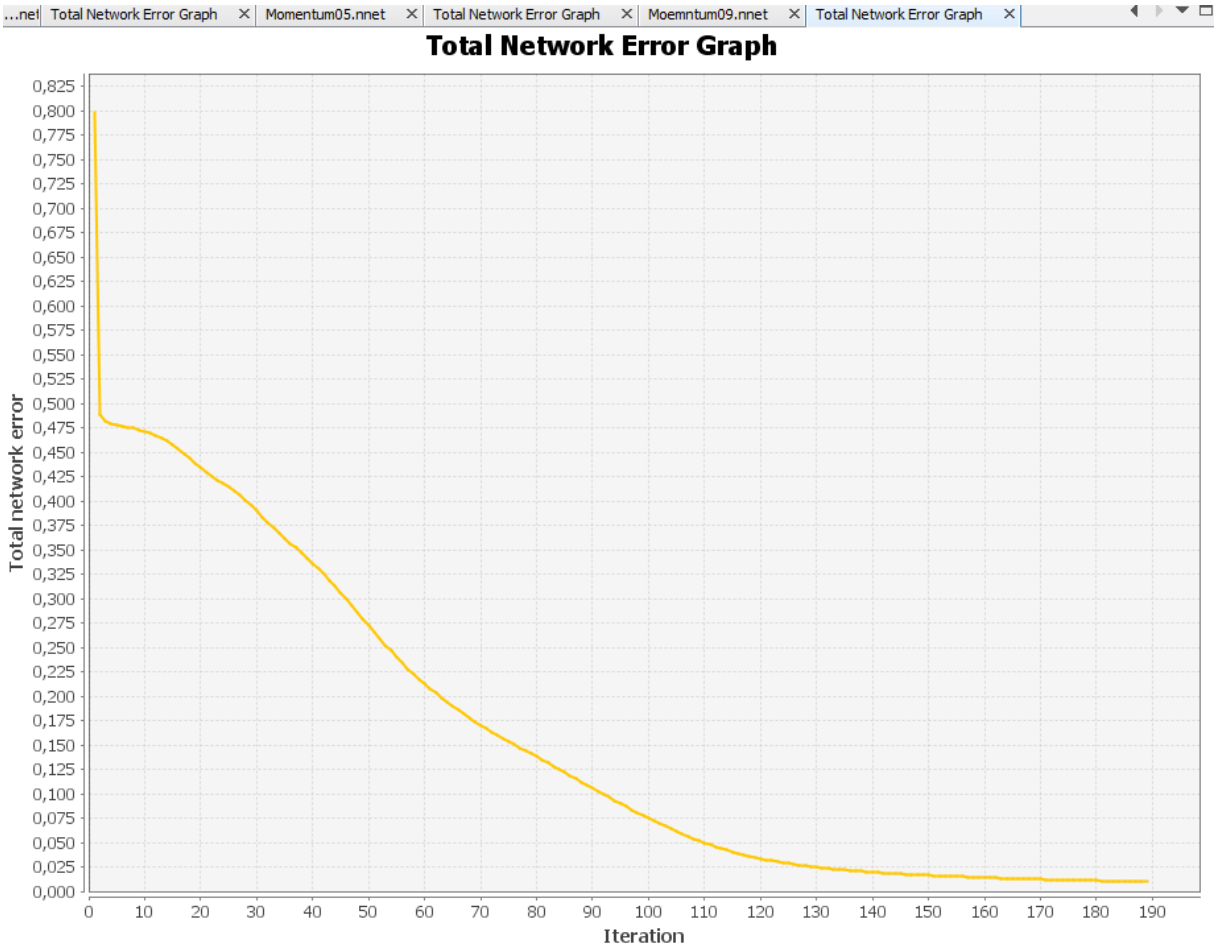
0.1



0.5



0.9



Wnioski.

Algorytm wstecznej propagacji błędów pozwala na budowanie sieci wielowartwowych. Większa ilość neuronów w warstwie ukrytej pozwala na szybsze uczenie się. Niestety wiąże się to ze zwiększonym zapotrzebowaniem na zasoby systemowe.

Współczynnik uczenia znacząco wpływa na szybkość uczenia się naszej sieci, w przeciwieństwie do współczynnika bezwładności, którego wpływ na ilość iteracji jest niewielki, przy czym wyższa wartość współczynnika uczenia pozytywnie wpływa na szybkość uczenia. Różnica między learning rate ustawionym na wartość 0.1 i 0.9 to różnica nieco ponad 1200 iteracji. Choć różnica spowodowana różnymi wartościami momentum jest kosmetyczna, to przemawia za niższą jego wartością.

Za każdym razem testy przebiegły ‘pomyślnie’. Dla współczynnika uczenia 0.5 i 0.9 wynik był zbliżony ~0,74 dla prawidłowej litery. Dla 0.1 było to aż 0.89, jednak ceną za tak wysoką skuteczność 13 krotność iteracji. Biorąc pod uwagę momentum, najskuteczniejsza okazała się wartość 0.5 gdyż dla niej suma wyniosła 0.87, dla 0.1- 0.83, zaś dla 0.9-0.86. Pozwala to twierdzić, że najbardziej optymalna wartość jest zbilansowana.