

Katedra Informatyki i Automatyki

**Sztuczna inteligencja – projekt**

Realizacja sieci neuronowej uczonej algorytmem wstecznej propagacji błędu z przyspieszeniem metodą adaptacyjnego współczynnika uczenia i metodą momentum uczącą się klasyfikacji jakości nauczania.

Wykonał:

Kamil Król

II EF-DI

Rzeszów 2019

Spis treści

1. **Opis danych i problemu**
   1. **Opis danych**

Dane zawierają ewaluację wyników nauczania 151 asystentów dydaktycznych Wydziału Statystycznego Uniwersytetu Wiscosin-Madison zebrane w ciągu trzech zimowych i dwóch letnich semestrów. Wyniki podzielono na trzy kategorie podobnej wielkości („low”, „medium” i „high”). Zestaw danych nie miał brakujących wartości, zebrał je Wei-Yin Loh z Wydziału Statystycznego UW-Madison.

**Atrybuty:**

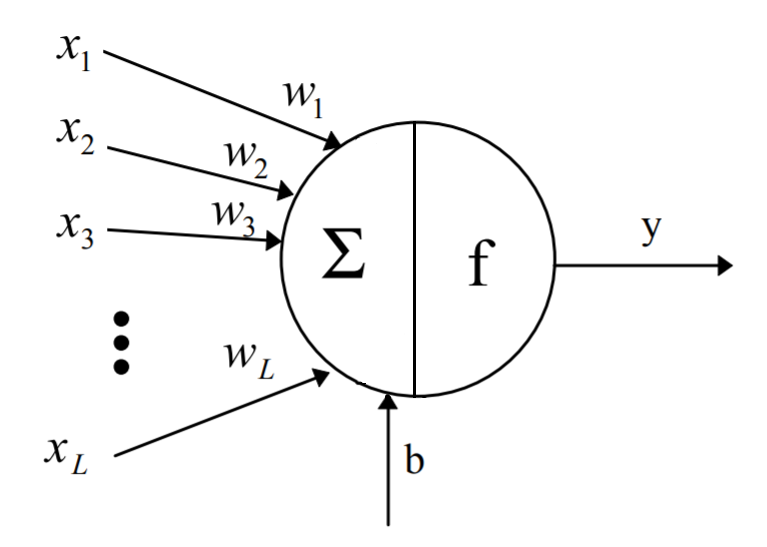
* Czy język angielski jest językiem ojczystym asystenta ( 1 = tak, 2 = nie )
* Instruktor kursu ( 25 kategorii )
* Kurs ( 26 kategorii )
* Rodzaj semestru ( 1 = letni, 2 = zimowy )
* Ilość osób w klasie
* Wynik ewaluacji ( 1 = low, 2 = medium, 3 = high )
  1. **Opis problemu**

Celem projektu było stworzenie sieci neuronowej wykorzystującej metodę wstecznej propagacji błędu z metodą adaptacyjnego współczynnika uczenia oraz metodą momentum, która na podstawie podanych parametrów ( punkt 1.1. ) przewiduje wynik ewaluacji nauczania. W projekcie zostało użyte środowisko Matlab 2019a.

1. **Sztuczne sieci neuronowe – wstęp teoretyczny**

Człowiek w swojej twórczości często czerpie inspirację z natury i jej sprawdzonych sposobów rozwiązywania problemów – tak też jest w przypadku sieci neuronowych, opartych na działaniu komórek nerwowych. Neurony „informatyczne” są zbudowane analogicznie do tych „biologicznych”, podobnie przyjmują na wejście wiele sygnałów, przetwarzają je na jeden sygnał wyjściowy, decydują o jego mocy i przesyłają dalej. Postaram się przedstawić bliżej zasadę jego działania.

* 1. **Model neuronu**

****

Rysunek 1: Model neuronu

Neuron jest podstawowym elementem sieci neuronowych. Składa się on ze skończonej liczby wejść – oznaczmy je jako *L* – do których trafiają sygnały *x1, … , xL* przemnożone przez odpowiadające im wagi *w1, … wL* . Następnie ich wartości są do siebie dodawane w sumatorze, a powstała wartość jest zwiększana o bias ( wartość progową, przesunięcie ), oznaczaną jako *b.*

Zatem na wyjściu sumatora otrzymujemy sygnał *n* wyrażany zależnością:

Wytworzony w ten sposób sygnał podawany jest jako argument funkcji aktywacji *f*, która w zależności od modelu neuronu czy rozwiązywanego problemu może mieć różną postać. W tym przypadku zostaną użyte neurony, których funkcja aktywacji ma postać liniową bądź sigmoidalną.

Ogólne działanie neuronu możemy zatem opisać jako:

* 1. **Funkcje aktywacji**

Działanie neuronów może różnić się od siebie z zależności m.in. od użytych w nich funkcji aktywacji. Dobiera się je w zależności od rozważanego problemu, doświadczenia osoby przeprowadzającej badania, a także od posiadanych danych. Funkcja sigmoidalna, która została użyta w tym projekcie przyjmuje wartości z przedziału [-1 1], nie można zatem użyć jej ( bez normalizacji danych ), gdy chcemy na wyjściu neuronu uzyskać inne wartości. Funkcja sigmoidalna bipolarna określona jest wzorem:

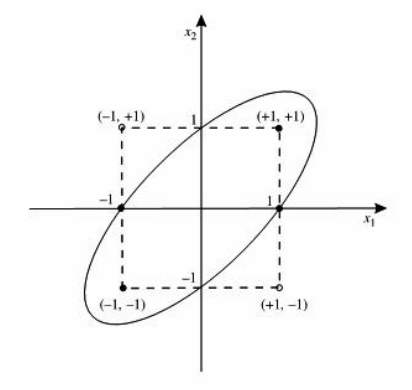
?

1. **Sieci jednokierunkowe wielowarstwowe**
   1. **Budowa i działanie**

Neuron jest przydatnym modelem matematycznym, jednak zastosowanie pojedynczych neuronów jest mocno ograniczone. Często przytaczanym przykładem problemu, który mimo swej prostoty przerasta możliwości jednego neuronu jest funkcja logiczna „wyłączne lub”, czyli XOR. Poniżej przedstawiono ciąg uczący funkcji:

|  |  |  |
| --- | --- | --- |
| x1 | x2 | d = XOR(x1, x2) |
| +1 | +1 | -1 |
| +1 | -1 | 1 |
| -1 | +1 | 1 |
| -1 | -1 | -1 |

Tabela 1 - Ciąg uczący da problemu XOR



Rysunek 2 - Ilustracja zagadnienia XOR

Z opisu i ilustracji wynika, iż nie istnieje taka granica decyzyjna, która mogłaby jednoznacznie podzielić przestrzeń na dwa zbiory rozwiązań – w tym przypadku przykładową granicą może być elipsa. Aby rozwiązać takie problemy stosuje się sieci neuronowe, czyli wiele połączonych ze sobą neuronów, rozmieszczonych w dwóch lub więcej warstwach. Każda sieć zawiera warstwę wejściową, do której podawane są dane uczące, oraz warstwę wyjściową, z której odbierany jest wynik uczenia. Dodatkowo sieć może zawierać jedną lub więcej warstw ukrytych. Wartościami wejściowymi dla neuronów kolejnych warstw są wyjścia wszystkich neuronów warstwy poprzedniej pomnożone przez odpowiadające im wagi. Sieci te nazywane są jednokierunkowymi, ponieważ sygnał przechodzi tylko z wejścia do wyjścia i nie występuje sprzężenie zwrotne, neurony nie mogą także przesyłać sygnału do innych ze swojej warstwy.

Rysunek sieci

* 1. **Algorytm wstecznej propagacji błędów**

Aby obliczyć prawidłową wagę połączeń neuronu wykorzystywaliśmy błąd powstały przy porównaniu wartości wyjściowej z wartością oczekiwaną. Jednak w przypadku sieci wielowarstwowych nauczyciel nie wie jaka powinna być wartość wyjściowa poszczególnych neuronów w warstwach ukrytych. Aby wyznaczyć te wartości używamy metody wstecznej propagacji błędów ( oraz kilku jej modyfikacji ).

1. **Eksperymenty**

Celem eksperymentów było zbadanie dla jakich wartości poszczególnych parametrów sieć uczy się najlepiej i najszybciej. Podczas eksperymentów zastosowano domyślny podział danych ( dividerand ) który dzieli dane na trenujące/walidacyjne/testowe w stosunku 70/15/15.

* 1. **Sprawdzanie wszystkich kombinacji parametrów**

W pierwszej kolejności sprawdzono poprawność klasyfikacji sieci dla wszystkich możliwych kombinacji poniższych parametrów:

|  |  |  |
| --- | --- | --- |
| **Parametr** | **Oznaczenie** | **Wartości wektora** |
| Ilość neuronów w pierwszej warstwie | S1\_vec | [3:1:15] |
| Ilość neuronów w drugiej warstwie | S2\_vec | [3:1:15] |
| Wartość współczynnika zwiększania uczenia | lr\_inc\_vec | [1.01:0.02:1.09] |
| Wartość współczynnika zmniejszania uczenia | lr\_dec\_vec | [0.5:0.1:0.9] |
| Wartość dopuszczalnej krotności przyrostu błędu | er\_ver | [1.01:0.15:1.07] |
| Wartość współczynnika momentum | mc\_vec | [0.1:0.2:0.9 0.95 0.99] |

Z uwagi na niewielką liczbę rekordów ( 151 par ) zdecydowano nie testować sieci dla zbyt wielu neuronów w warstwach, przy czym liczba neuronów w warstwie drugiej zawsze była mniejsza lub równa liczbie neuronów w warstwie pierwszej. Zgodnie z przeprowadzonym eksperymentem, najlepszą poprawnością klasyfikacji było 74.1722% osiągnięte dla 15 neuronów w warstwie pierwszej, 12 neuronów w warstwie drugiej, wartości zmniejszania/zwiększania współczynnika uczenia odpowiednio 0.9/1.01, dopuszczalnej krotności przyrostu błędu 1.07 oraz współczynnika momentum 0.99.

Sprawdzanie sieci neuronowej w ten sposób wydaje się rozsądne, ponieważ bierzemy pod uwagę wszystkie kombinacje parametrów, a sieci neuronowe bywają czułe na zmianę któregokolwiek z nich, jednak jest ono nieopłacalne czasowo. Sieć musiała wykonać niemal 80 000 prób nauczania sieci od początku, co w praktyce zajęło niecałe 24 godziny. Dodatkowo, badając wszystkie parametry, ciężko jest określić wpływ działania jednego z nich na działanie sieci, zatem w kolejnych rozważaniach zastosujemy nieco inne podejście do problemu, dzieląc go na mniejsze aspekty.

* 1. **Badanie optymalnej liczby neuronów w warstwach sieci**