Dokumentacja projektu NAI

Sieć neuronowa. Wariant uczący numeru indeksu „11027” przy pomocy sieci z dwoma warstwami ukrytymi w macierzami na wejściach.

Autor: Bartosz Kalinowski s11027, Data: 2014-12-31

Źródła: https://github.com/kelostrada/neuron-network

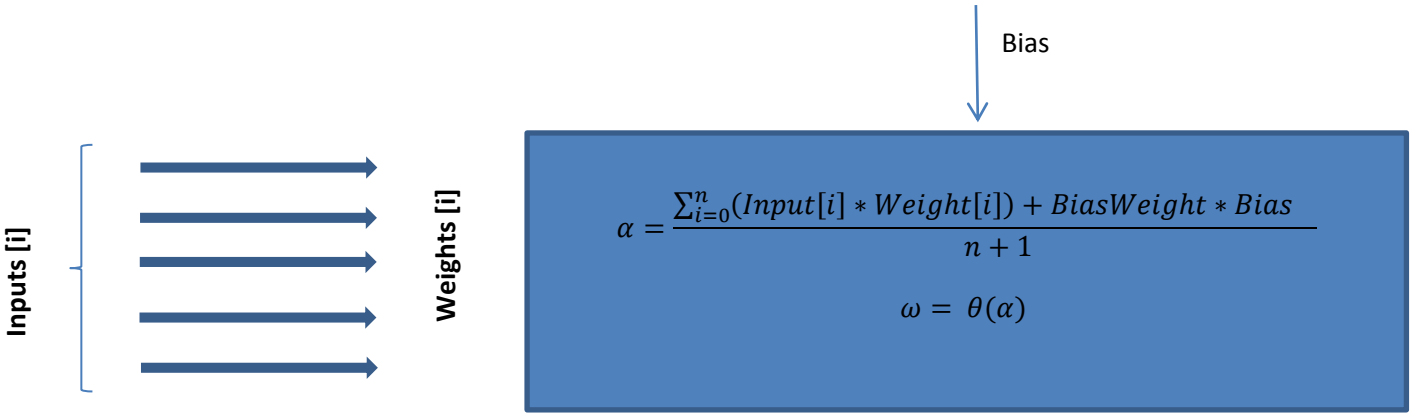
Spis treści

1. Opis sieci neuronowej

1.1 Budowa perceptronu

W omawianym rozwiązaniu zastosowany został standardowy perceptron wyliczający średnią ważoną ze wszystkich wejść oraz biasu.

Schemat budowy:



W pseudokodzie perceptron zbudowany został przy pomocy prostej klasy implementującej następujące interefejsy:

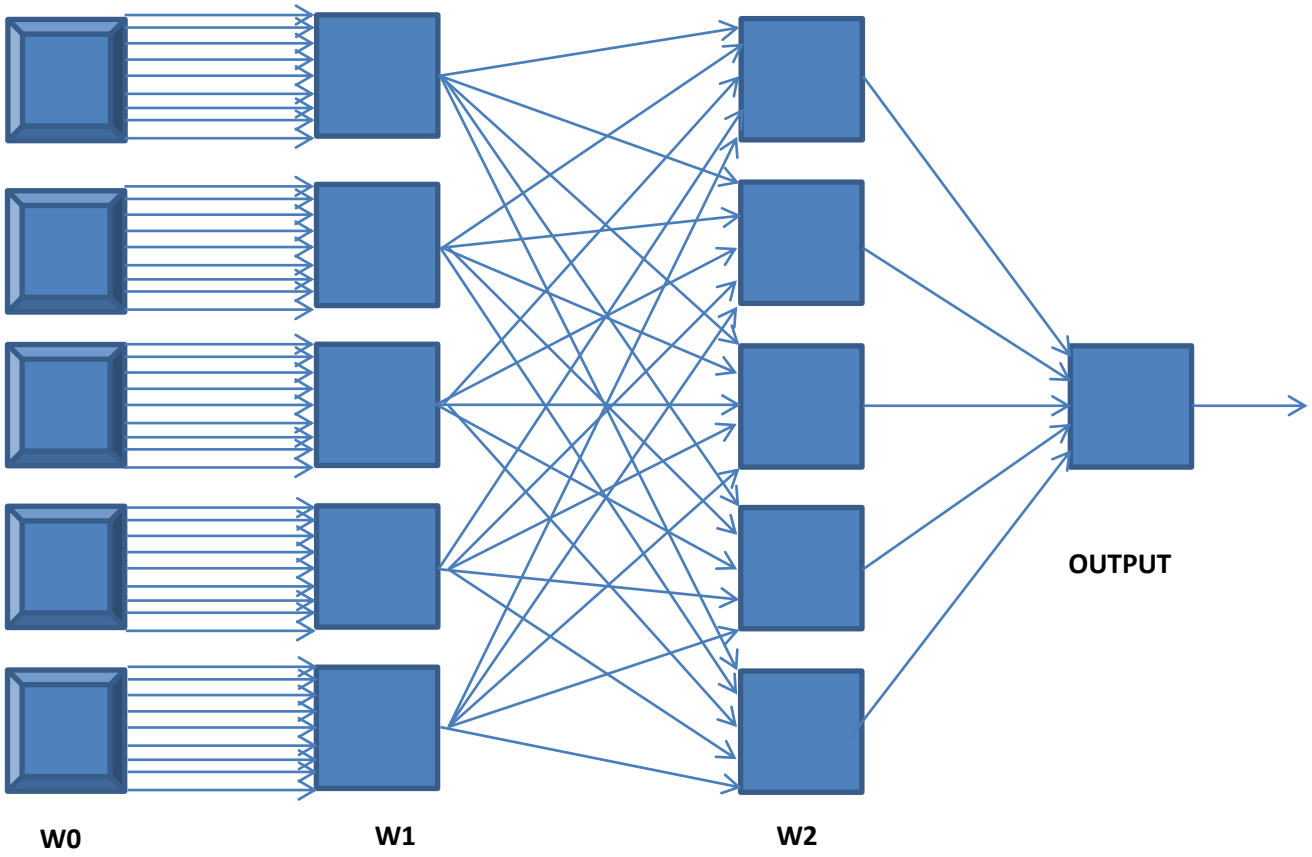
```
public interface IOutput
{
    double OutputValue { get; }
    List<IInput> Outputs { get; }
}
```

```
public interface IInput
{
    double Error { get; }
    double GetWeight(IOutput input);
    List<IOutput> Inputs { get; }
    void AddInput(IOutput input);
}
```

Interfejsy te pozwalają na stworzenie połączeń bezpośrednio pomiędzy implementującymi je klasami (poprzez referencje). Podczas procesu propagacji wstecznej potrzebna jest informacja zarówno o neuronach podłączony „z lewej” jak i „z prawej” strony. Stąd potrzeba wystawienia listy inputów i outputów. Ponadto potrzebna jest możliwość wyliczenia współczynnika błędu – w tym celu właśnie wystawiona jest właściwość Error.

1.2 Struktura sieci

Zastosowana została sieć składająca się z warstwy wejściowej (macierze podające sygnały), dwóch warstw ukrytych składających się każda z 5 perceptronów, oraz warstwy wyjściowej składającej się z jednego neuronu. Warstwy po kolei połączone są na zasadzie „każdy do każdego”. Wyjście sieci należy interpretować jako prawda/fałsz odnośnie zadanego problemu (wyniki bliskie zeru wskazują na fałsz, a bliskie jedynce wskazują naprawdę).

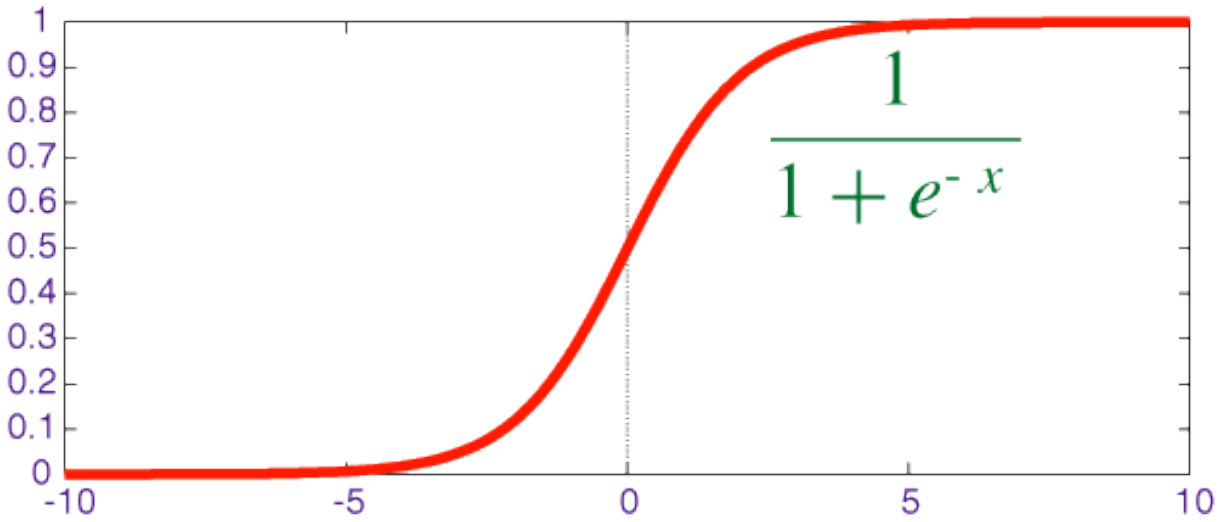


Warstwa W0 jest warstwą składającą się z binarnych macierzy 3x3 z których każda podaje po 9 inputów dla warstwy W1.

Wagi wejść wszystkich neuronów generowane są losowo.

1.3 Funkcja aktywacji

Zastosowana została funkcja aktywacji – **sigmoidalna**. Funkcja ta używana jest na każdym wyjściu z każdego perceptronu (tzn każdy wynik perceptronów przepuszczany jest przez tę funkcję). Ponadto przy propagacji wstecznej wykorzystywana jest pochodna tej funkcji której wzór jest bardzo prosty, a mianowicie: $\Theta'(x) = \Theta(x)(1-\Theta(x))$



Pełen wzór zastosowanej funkcji to:

$$y = \frac{1}{1+e^{-\mu x}}$$

Gdzie μ to parametr stromości. W dalszej części dokumentu pokazane zostanie, że modyfikacja tego parametru może przysłużyć się do przyspieszania (lub uniemożliwiania) procesu uczenia się sieci.

2. Opis zadania

2.1 Wstęp

Rozważmy zbiór wszystkich możliwych macierzy binarnych wymiaru 3×3. Suma elementów dowolnej macierzy z tego zbioru odpowiada pewnej cyfrze systemu dziesiętnego, np. macierz

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

reprezentuje w tym modelu zależności cyfrę 4. Oczywiście ten sposób reprezentacji cyfr nie jest jednoznaczny dla każdej cyfry systemu dziesiętnego innej niż 0oraz 9. Dla przykładu cyfrę 4można zapisać na $\binom{9}{4}=126$sposobów, np. macierze

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

także reprezentują cyfrę 4.

2.2 Zadanie

Zaprojektuj, zaimplementuj (dowolny język programowania) i wytrenuj sieć neuronową, która dla zadanego zestawu wejściowego czterech macierzy $IN=\{M_1,M_2,M_3,M_4\}$dokona jego klasyfikacji ze względu na to, czy liczba $M_1M_2M_3M_4$jest numerem Twojego indeksu na PJWSTK (jeżeli Twój numer indeksu składa się z inne liczby cyfr dziesiętnych niż cztery, dostosuj właściwie liczbę macierzy zestawu wejściowego). Jako algorytmu uczenia zaprojektowanej sieci neuronowej zastosuj metodę wstecznej propagacji błędów dla samodzielnie dobranego zbioru trenującego macierzy wymiaru 3×3.

Zbadaj kolejno:

- efektywność procesu nauki sieci ze względu na wybór typu dostępnych perceptronów oraz wartość współczynnika kroku nauki $0\leq\eta\leq1$,
- efektywność klasyfikacji danych wejściowych w zależności od wartości parametrów sterujących funkcjami aktywacji.

3. Wnioski

3.1 Badane dane

Utworzoną sieć próbowano nauczyć rozpoznawania numeru „11027”. Badania rozpoczęto z próbą kilku danych testowych bez zmian parametrów η oraz μ (tzn ustawiono ich wartości na domyślne – czyli 1). Uczenie sieci w takim podstawowym przypadku nie powiodło się, dlatego przystąpiono do wygenerowania dużej próby danych testowych. Założono, że wszystkie dane będą losowe, jednak żeby ułatwić sieci uczenie się, połowa danych to zestawy macierzy „poprawnych” tzn takich które spełniają oczekiwane warunki, a druga połowa to dane błędne.

Okazało się, że w takich warunkach sieć zaczęła uczyć się na poziomie 4000-5000 iteracji (epok) z błędem na poziomie 0,1. Był to już wynik pozwalający na rozpoczęcie szczegółowych badań.

3.2 Wykresy wydajności

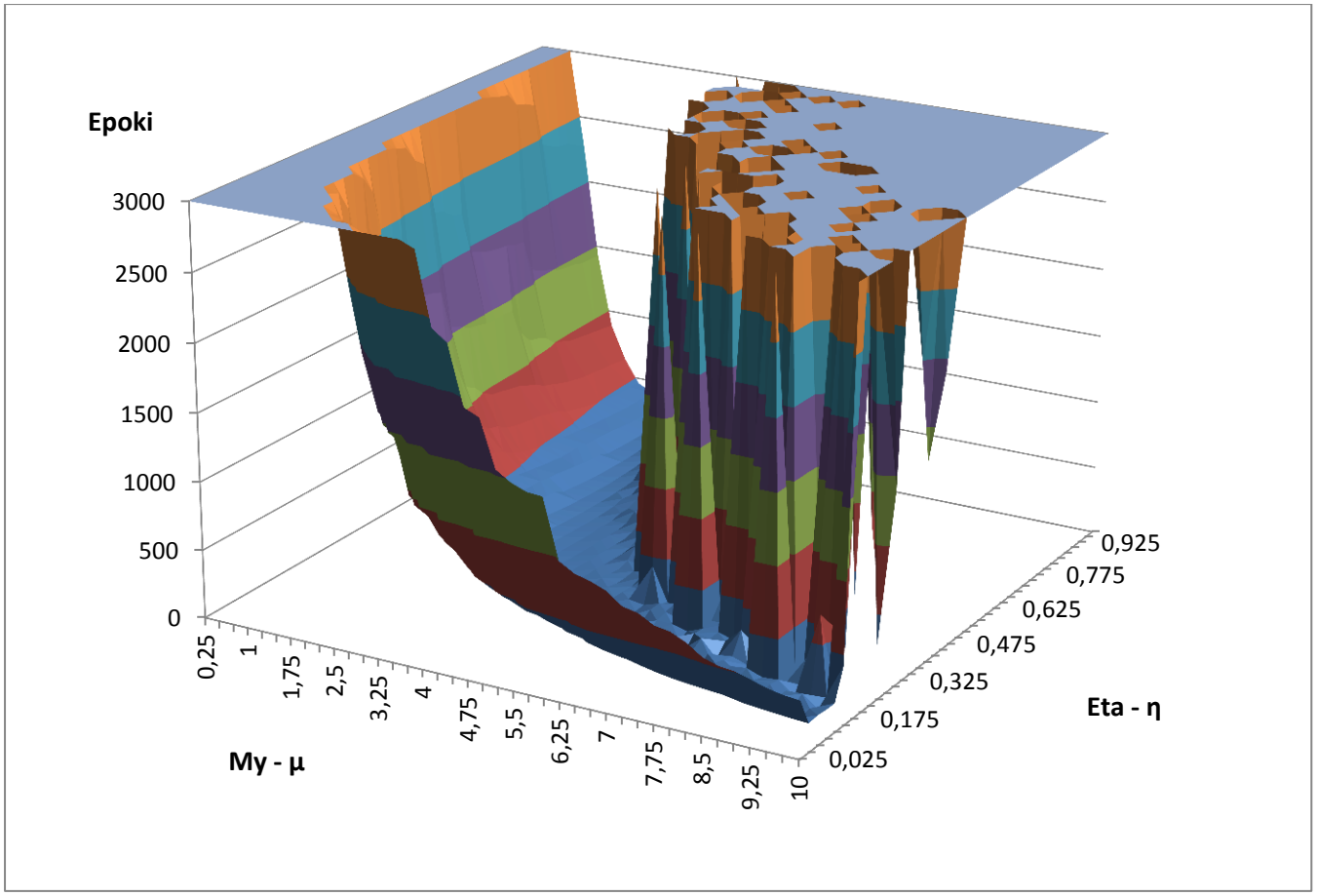
Aby dowiedzieć się więcej na temat wydajności uczenia się omawianej sieci wprowadzono ruchome zmienne η oraz μ. Zmienna μ odpowiedzialna za stromość funkcji aktywacji przyjmować będzie wartości z przedziału (0,20], a zmienna η odpowiedzialna za szybkość uczenia się przyjmować będzie wartości oczywiście od [0,1].

Próbka danych do uczenia się zostaje wylosowana na początku działania programu i nie zmienia się już do końca (każda zmiana jednego z dwóch wcześniej wymienionych parametrów wiąże się ze zresetowaniem stanu sieci i podaniem danych do nauki od początku).

Wagi zostają wylosowane jednorazowo i każdy restart sieci wiąże się z przepisaniem ich na nowo (w ten sposób wiarygodniej można porównać poszczególne warianty uczenia się).

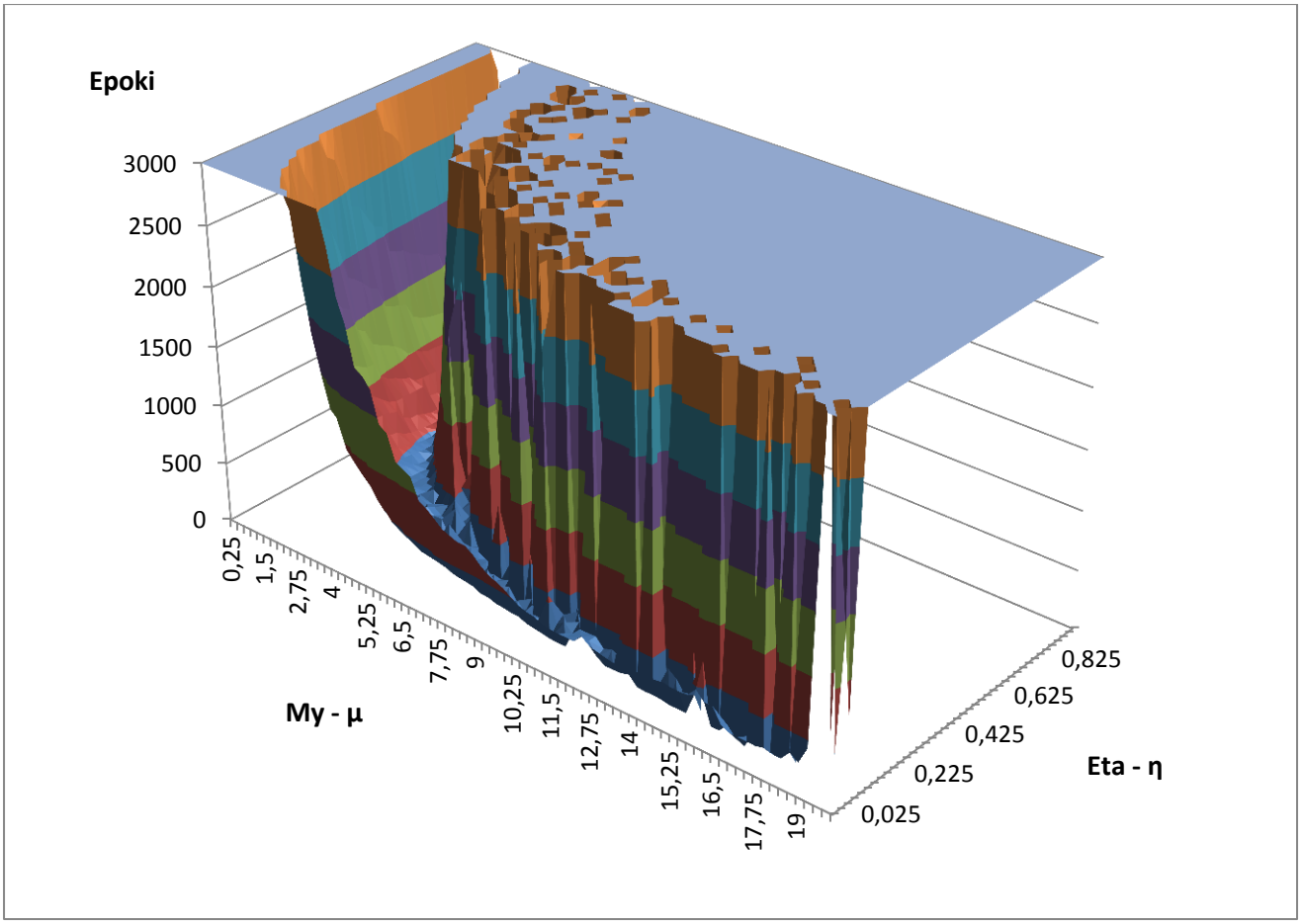
Warunek stopu to 3000 epok/iteracji lub błąd na poziomie 0,1. Sprawdzono sieć także dla mniejszego dopuszczalnego błędu, ale problem polega na tym, że należało wygenerować bardzo duże zbiory danych a wykresy które powstawały w ten sposób zachowywały się niemalże identycznie jak przy słabszym określaniu błędu. Poziom błędu sprawdzany jest na podstawie ostatnich 100 iteracji (tak, żeby nie było możliwości wystąpienia losowego poprawnego układu już na początku nauki sieci).

Zastosowano wykresy płaszczyznowe trójwymiarowe – w ten sposób oba wymagane z zadania podpukty zostają przedstawione na jednym wykresie.



Wykres z parametrem μ z przedziału (0,10]

Jak widać na wykresie, tworzy się tutaj swoista „rzeka” lub „kanion”. Dół tego kanionu oznacza miejsca dla parametrów η oraz μ gdzie sieć uczy się najszybciej. Wnioskuje się, że dla takiego układu sieci neuronowej najszybciej uczy się ona gdzieś dla parametru μ o wartości około 3, a dla parametru η o wartości 0,4.



Wykres z parametrem μ z przedziału (0,20]

Wykres ten obrazuje podobną sytuację jak poprzedni, ale widać na nim jeszcze, że „kanion” z przedziałem szybkiego uczenia się bardzo się zwężył dla wartości μ większych od 10. Oznacza to, że dalsze zwiększanie tego parametru raczej już nie pomoże w przyspieszeniu procesu nauki sieci neuronowej.

3.3 Ostateczne wnioski

Okazało się, że sieć dla zadanego problemu osiąga wartości błędu rzędu 0,1 w najlepszych przypadkach przy około 120 epokach. Wydaje się to być zaskakująco dobrym wynikiem i daje spore nadzieje, na wykorzystanie takiej sieci przy innych bardziej skomplikowanych problemach.