Krzysztof Kaflowski

Podstawy Sztucznej Inteligencji

Sprawozdanie ze Scenariusza nr 2

„Rozpoznawanie wielkości liter”

1. Celem ćwiczenia było stworzenie jednowarstwowej sieci neuronowej, która ma za zadanie rozpoznawać czy litera jest wielka czy mała.

2a)

Sieć neuronowa (sztuczna sieć neuronowa) – ogólna nazwa [struktur matematycznych](https://pl.wikipedia.org/wiki/Struktura_matematyczna) i ich programowych lub sprzętowych modeli, realizujących obliczenia lub [przetwarzanie sygnałów](https://pl.wikipedia.org/wiki/Przetwarzanie_sygna%C5%82%C3%B3w) poprzez rzędy elementów, zwanych [sztucznymi neuronami](https://pl.wikipedia.org/wiki/Sztuczny_neuron), wykonujących pewną podstawową operację na swoim wejściu. Oryginalną inspiracją takiej struktury była budowa [naturalnych neuronów](https://pl.wikipedia.org/wiki/Neuron), łączących je [synaps](https://pl.wikipedia.org/wiki/Synapsa), oraz [układów nerwowych](https://pl.wikipedia.org/wiki/Uk%C5%82ad_nerwowy), w szczególności [mózgu](https://pl.wikipedia.org/wiki/Kresom%C3%B3zgowie).



Problem zrealizowano za pomocą jednego perceptronu. Przygotowano dane uczące zawierające 10 małych liter oraz 10 dużych liter w postaci siatki 5x7 w reprezentacji zero-jedynkowej. Wykorzystano jedynie regułę delta, ponieważ na wykładzie nie zostały zaprezentowane żadne inne metody jak również budowa sieci neuronowych. Z tego powodu jedynym narzędziem który można było wykorzystać był pojedynczy perceptron.

Reguła Delta:

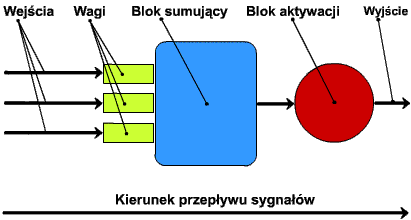
Dobry opis tej reguły znajduje się tutaj:  
http://galaxy.agh.edu.pl/~vlsi/AI/rdelta/  
Krótkie streszczenie zawartości strony:

Reguła delta polega na porównaniu otrzymanego wyniku oraz wyniku, które powinny być otrzymany. Wyliczana jest różnica nazywana błędem dzięki której poprawiane są wagi na dokładniejsze.

Perceptron stworzono na podstawie źródła z Internetu:

<http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad3/w3.htm>  
  
Krótkie streszczenie teorii zawartej na tej stronie:

Budowa perceptronu:



**Perceptronem** nazywamy prosty element obliczeniowy, który sumuje ważone sygnały wejściowe i porównuje tę sumę z progiem aktywacji - w zależności od wyniku perceptron może być albo wzbudzony (wynik 1), albo nie (wynik 0).

Schemat działania perceptronu:

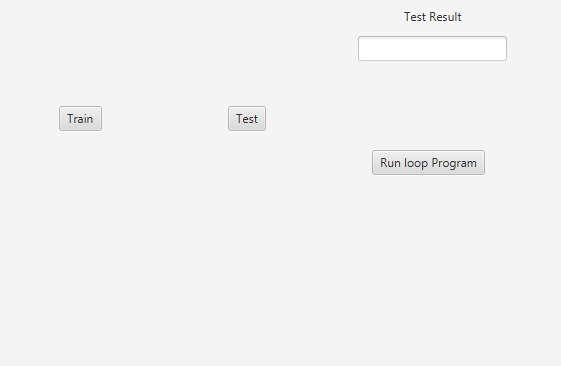
* Inicjujemy wagi losowo.
* Dla każdego przykładu uczącego obliczamy odpowiedź perceptronu.
* Jeśli odpowiedź perceptronu jest nieprawidłowa, to modyfikujemy wagi:

w1 += n \* (d-y) \* x1   
w2 += n \* (d-y) \* x2   
b   += n \* (d-y)

gdzie n jest niewielkim współczynnikiem uczenia (n > 0), d - oczekiwana odpowiedź a y - odpowiedź neuronu.

2b)

Wygląd programu:



Zastosowanie przycisków:

* Learn – nauczanie perceptronu danymi z pliku trainData.txt
* Test – testowanie poprawności programu za pomocą danych z pliku testData.txt
* Run Loop Program – uruchamia naprzemiennie 100 razy Learn oraz Test aby zobaczyć jak zmienia się dokładność programu

Kluczowe metody:

* learn2() – zczytuje wszystkie dane pochodzące z zestawu uczącego w pliku tesktowym, następnie oblicza odpowiedź perceptronu, na końcu sprawdzania każdej pary liczb, poprawia wagi
* adjustWeights() – poprawia aktualne wagi na podstawie otrzymanego błędu obliczania wyniku
* weightSum2() – oblicza średnią ważona używając aktualnych wag oraz pary liczb z danych wejściowych
* test() – oblicza wynik odpowiedzi perceptronu dla każdej pary liczb z plik służacego do walidacji neuronu, następnie wyświetla ile procent estymacji zakończyło się sukcesem
* sigmoid() - oblicza wartości funkcji sigmoidalnej, wzór poniżej

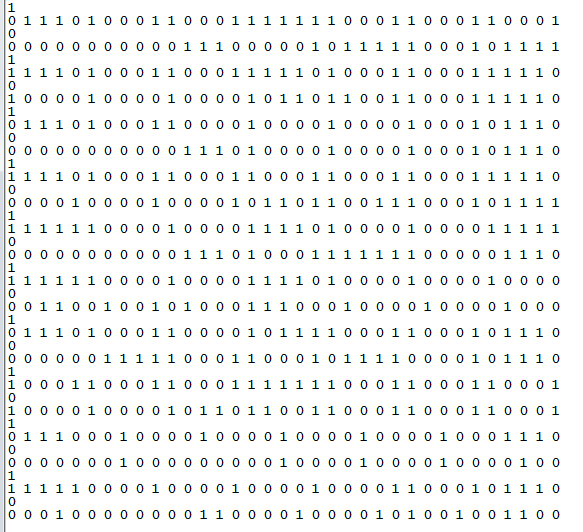
Funkcja sigmoidalna:

Funkcja sigmoidalna jest stosowana do obliczenia końcowej wartości odpowiedzi perceptronu. Wartość zawiera się w przedziale <0,1>. Jeżeli wartość jest bliska 0 to znaczy, że litera jest mała, a jeżeli wartość jest bliska 1 to znaczy, że litera jest wielka.

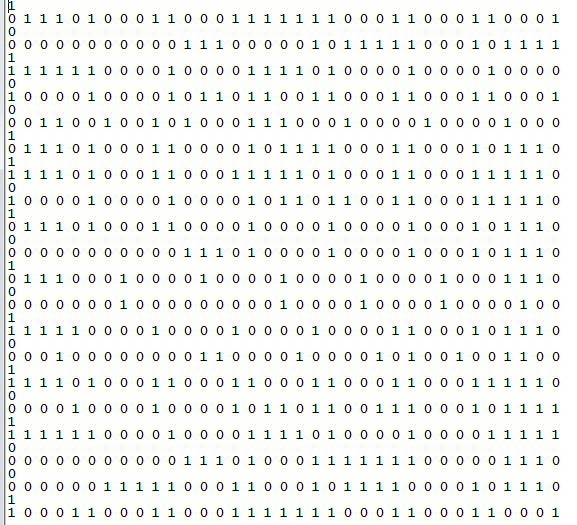
2c)

Program testowano dla 3 współczynników uczenia: 0,1; 0,01; 0,001.

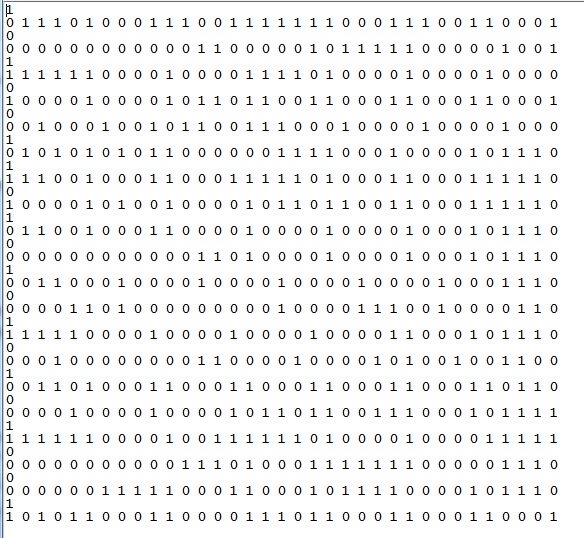
Zawartość pliku trainData.txt stosowany do nauki perceptronu:



Zawartość pliku testData.txt testującego neuron:

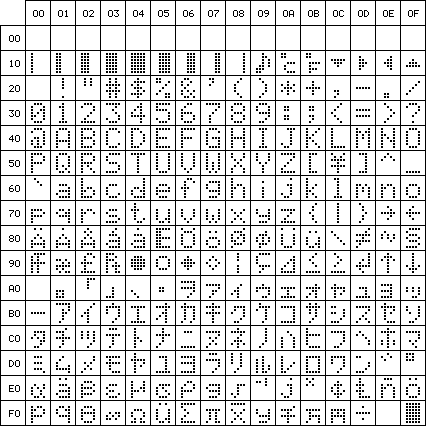


Zawartość pliku testującego zawierający zniekształcone dane distortedData.txt:



Pliki zawierają najpierw wartość docelową czy litera jest duża czy mała, a następnie reprezentacji siatki litery 5x7 w formacie binarnym.

Wykorzystane siatki prezentują się następująco:



Przykład stworzonej siatki dużych liter:

A

0 1 1 1 0

1 0 0 0 1

1 0 0 0 1

1 1 1 1 1

1 0 0 0 1

1 0 0 0 1

1 0 0 0 1

B

1 1 1 1 0

1 0 0 0 1

1 0 0 0 1

1 1 1 1 0

1 0 0 0 1

1 0 0 0 1

1 1 1 1 0

C

0 1 1 1 0

1 0 0 0 1

1 0 0 0 0

1 0 0 0 0

1 0 0 0 0

1 0 0 0 1

0 1 1 1 0

D

1 1 1 1 0

1 0 0 0 1

1 0 0 0 1

1 0 0 0 1

1 0 0 0 1

1 0 0 0 1

1 1 1 1 0

Przykład stworzonej siatki małych liter:

a

0 0 0 0 0

0 0 0 0 0

0 1 1 1 0

0 0 0 0 1

0 1 1 1 1

1 0 0 0 1

0 1 1 1 1

b

1 0 0 0 0

1 0 0 0 0

1 0 0 0 0

1 0 1 1 0

1 1 0 0 1

1 0 0 0 1

1 1 1 1 0

c

0 0 0 0 0

0 0 0 0 0

0 1 1 1 0

1 0 0 0 0

1 0 0 0 0

1 0 0 0 1

0 1 1 1 0

d

0 0 0 0 1

0 0 0 0 1

0 0 0 0 1

0 1 1 0 1

1 0 0 1 1

1 0 0 0 1

0 1 1 1 1

Poniższy wykres przedstawia wyniki 100 iteracji nauczania oraz wyniku procentowego sukcesu testu:

Jasno widać, że dla współczynnika uczenia rownego 0,1 neuron uczy się najszybciej i już po około 140 iteracjach jest maksymalna dokładność rozpoznawania wielkości liter i był bezbłędny. Z drugiej strony współczynnik uczenia równy 0,001 sprawił, że proces nauczania mocno się spowolnił i po wprowadzeniu 2000 danych wejściowych ledwo osiągnął 60% dokładności rozpoznawania wielkośi liter.

Poniższy wykres przedstawia wyniki 100 iteracji procesu nauczania oraz wyniki procentowego sukcesu testu dla zniekształconych danych testujących znajdujących się w pliku distortedData.txt:

Jak widać na wykresie program nigdy nie osiąga 100% dokładności rozpoznawania wielkości liter nawet po wprowadzeniu 2000 danych uczących. Osiągniecie 90% dokładności uważam za satysfakcjonujące.

2d)

Analiza programu i wynikająch błędów:

Na działanie sieci maja wpływ współczynnik uczenia, ilość danych uczących, funkcja aktywacji oraz metoda uczenia.

2e)

Wnioski:

* Perceptron wykorzystany do nauki idealnie nadał się do rozwiązania problemu.
* Proces uczenia dla wyższych współczynników przebiega bardzo szybko.
* Program jest w stanie szybko osiągnąć stuprocentową dokładność rozpoznawania wielkości liter.

2f)

Listing kodu całego programu oraz pliku uczące oraz testujące znajdują się pod linkiem:

<https://github.com/kkaflows/PSI-Scenariusz-2-Rozpoznawanie-wielkosci-liter>

2g)

Uwagi:

* Wykorzystany został tylko jeden algorytm, ponieważ na wykłądzie nie został zaprezentowany żaden inny