Kamil Stachowicz

PSI - Sprawozdanie

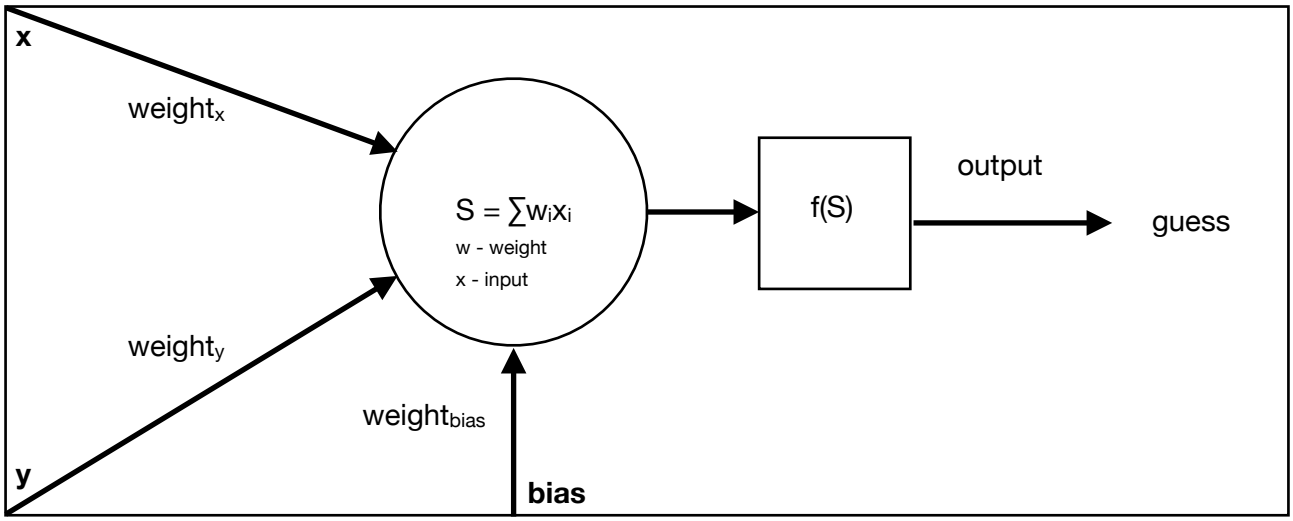
Laboratorium nr 1

Opis wykonywanego ćwiczenia
Celem ćwiczenia jest poznanie budowy i działania perceptronu poprzez implementację oraz uczenie perceptronu realizującego wybraną funkcję logiczną dwóch zmiennych.

Schemat ćwiczenia
1. Zaimplementowanie sztucznego neuronu
2. Wygenerowanie zestawu uczącego dla bramki logicznej AND
3. Uczenie perceptronu za pomocą różnej liczby danych
4. Testowanie perceptronu

Wykonanie ćwiczenia

Perceptron został zaimplementowany według następującego źródła:
https://github.com/shiffman/NOC-S17-2-Intelligence-Learning/blob/master/week4-neural-networks/perceptron.pdf



Perceptron dla tego zadania przyjmował dwie wartości x i y, miał odgadnąć ich iloczyn logiczny. Przyjmuje on trzy wejścia:
- X
- Y
- Bias

Bias jest domyślnie ustawiony na 1, wszystkie trzy wagi wejść mają losową wagę która zmienia się w procesie uczenia. Perceptron może zwracać wartość 1 czyli prawdę oraz -1 czyli fałsz.

Funkcja na podstawie której obliczana jest wartość wyjściowa to unipolarna funkcja progowa, zwraca ona tylko dwie wartości co dobrze obrazuje problem rozwiązywany przez perceprtron.

$$f\left(S\right) = \begin{cases} -1 & dla\ x < 0 \\ 1 & dla\ x \geq 0 \end{cases}$$

Schemat działania:
- Perceptron przyjmuje wejścia których wynik działania jest nam znany
- Próba odgadnięcia - *perceptron zwraca wartość 1 lub -1*
- W razie błędu informujemy o tym perceptron. Wyliczany jest błąd z wzoru

Kamil Stachowicz

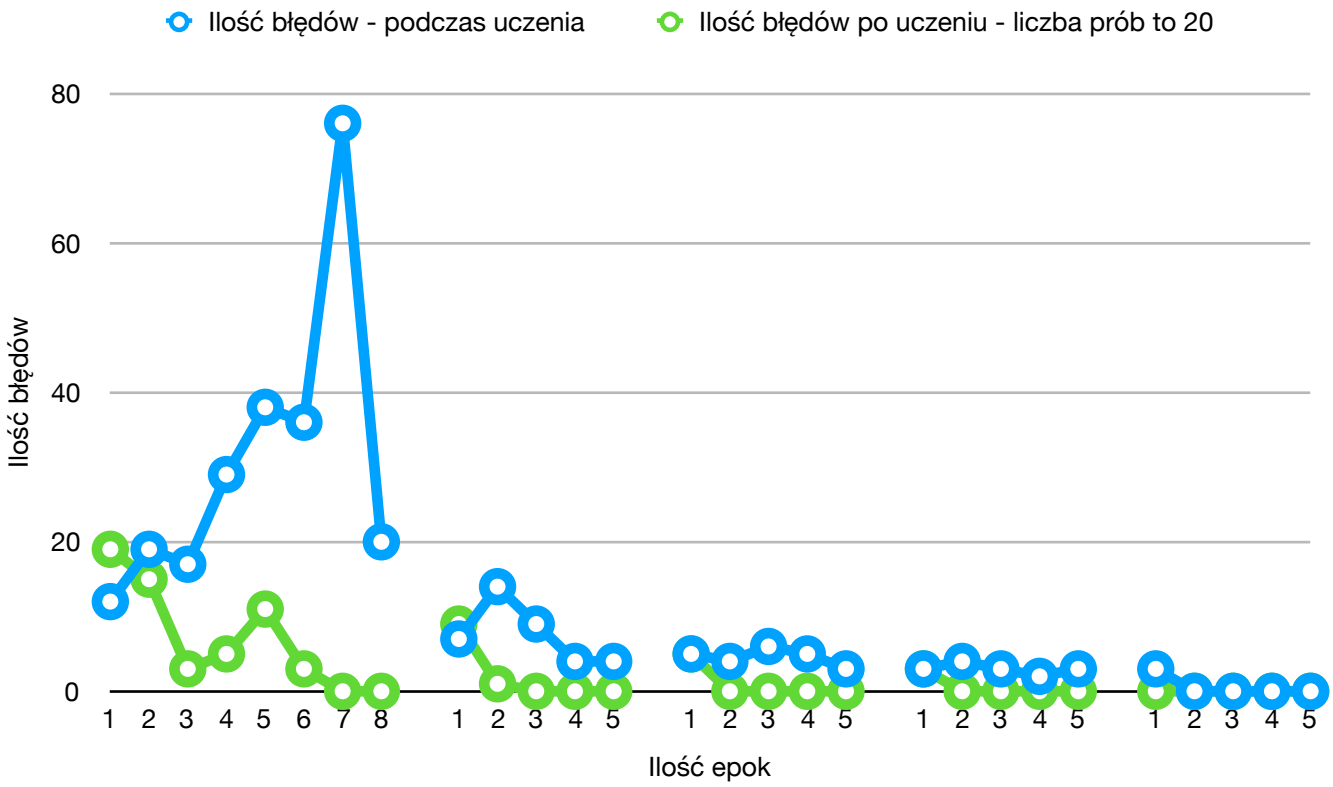
BŁĄD = Oczekiwana wartość - odgadnięta wartość

Ilość epok - uczenie 20 punktami	Ilość błędów	Ilość prób	Ilość błędów po uczeniu - liczba prób to 20	LR
1	12	20	19	
2	19	40	15	
3	17	60	3	
4	29	80	5	0,01
5	38	100	11	
6	36	120	3	
7	76	140	0	
8	20	160	0	
1	7	20	9	
2	14	40	1	
3	9	60	0	0,05
4	4	80	0	
5	4	100	0	
1	5	20	5	
2	4	40	0	
3	6	60	0	0,1
4	5	80	0	
5	3	100	0	
1	3	20	3	
2	4	40	0	
3	3	60	0	0,5
4	2	80	0	
5	3	100	0	
1	3	20	0	
2	0	40	0	
3	0	60	0	1
4	0	80	0	
5	0	100	0	

- Dopasowywane są wagi dla każdego z wejścia:
  Nowa waga = stara waga + błąd * wejście * szybkość uczenia
- Powtórzenie schematu

Zestawienie wyników

Analiza błędów



Każdy z perceptronów uczony był 20 wygenerowanymi losowo zestawami wejść i wyjść, uczenie to było powtarzane (ilość epok).
Learning rate na poziomie równym 0,01 dla tej funkcji jest bardzo nieoptymalny, nauka trwa stosunkowo długo i jak można zauważyć na wykresie (np. dla 7 epok) potrafi być bardzo nieefektywna.
Im większy learning rate tym szybciej i sprawniej przebiega uczenie perceptronu. Dla learning rate równego 1 i tylko jednej epoce perceptron popełnił zaledwie 3 błędy podczas uczenia się i bezbłędnie rozwiązywał problem który mu zadano.

Wnioski

Na podstawie wyników z laboratorium należy wysnuć wniosek że dwoma najważniejszymi aspektami podczas konstruowania zadania i warunków dla sztucznej sieci neuronowej, jest dobranie odpowiedniego learning rate, ilości epok (w związku z epokami ważne jest również dobranie odpowiedniej ilości danych uczących w każdej epoce) oraz odpowiedniej dla problemu - funkcji progowej.

Dobranie odpowiedniego learning rate może spowodować że znacznie zaoszczędzony zostanie czas potrzebny na uczenie perceptronu, w przypadku jednego jak w danym ćwiczeniu, czas wykonania się uczenia, nawet dla znacznej ilości epok jest niski, należy jednak zauważyć że przy bardziej rozległej sieci neuronowej, liczba zadań do wykonania przez jednostki obliczeniowe będzie znacznie wzrastać.

Funkcja progowa zwracająca dwie wartości, w tym zadaniu okazała się bardzo dobrym wyborem, nie jest ona złożona ani obliczeniowo, ani logicznie wobec czego wykonuje się szybko i przynosi

zadowalające efekty - (przy wysokim learning rate i przy małej liczbie epok perceptron stawał się bezbłędny). Niestety na podstawie dwóch wartości nie jest możliwe wyznaczenie na przykład skali podobieństwa dwóch zdjęć czy wytrzymałości materiałów.

Ważnym wyborem jest też język w którym implementujemy sieć neuronową, wybór pythona w projekcie okazał się być dobrym pomysłem ze względu na prosty i czytelny kod oraz stosunkowo szybki czas wykonywania się programu. Dodatkowym aspektem do używania go w przyszłych projektach jest mnogość dodatkowych bibliotek i gotowych zaimplementowanych rozwiązań z zagadnienia AI.

Podsumowując: takie zagadnienie jak nauka perceptronu wykonywania operacji zgodnych z bramką logiczną, jest zagadnieniem stosunkowo prostym. Pokazuje ono jednak na jakich aspektach należy się skupić w przyszłości, podczas tworzenia modelu dla bardziej skomplikowanego zagadnienia.
Należy też zauważyć że jeden perceptron nie jest w stanie nauczyć się rozwiązywać skomplikowanych problemów, niemożliwym dla niego jest nauczenie się logiki bramki XOR, która w teorii nie jest bardziej skomplikowana od bramki AND czy OR.

Kod programu

```
#!/usr/bin/python
import random

class Perceptron:

    def __init__(self, lr):
        self.weights = [0,0,0]
        self.weights[0] = random.uniform(-1,1)
        self.weights[1] = random.uniform(-1,1)
        self.weights[2] = random.uniform(-1,1)
        self.wrong = 0

        self.x = 0
        self.y = 0

        self.bias = 1
        self.learningRate = lr
        print("Learning rate = ", self.learningRate)
    def displayOutput(self): //jest to funkcja odpowiedzialna za wyświetlanie wyniku działania perceptronu
    def guess(self, x, y):
        self.x = x
        self.y = y
        self.output = x*self.weights[0] + y*self.weights[1] + self.bias*self.weights[2]
        if self.output >= 0:
            self.output = 1
        else:
            self.output = -1

        self.displayOutput()
        return self.output

    def train(self, x, y, desiredOutput):
        output = self.guess(x, y)
        error = desiredOutput - output

        self.weights[0] += error * self.x * self.learningRate
        self.weights[1] += error * self.y * self.learningRate
        self.weights[2] += error * self.bias * self.learningRate
```

Źródła:
https://github.com/shiffman/NOC-S17-2-Intelligence-Learning/blob/master/week4-neural-networks/perceptron.pdf