НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи №1-1

із дисципліни «Методи штучного інтелекту»

на тему

*Розробка програмного забезбепечення для реалізації двошарового персептрону з сигмоїдальною функцією активації*

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студентка групи КМ-81 | Доцент |
| *Верзун П. В.* | Терейковська Л.О. |

Київ — 2021

Зміст

[**Постановка задачі**](#_tfvq2xe2h67d) **3**

[**Короткі теоретичні відомості**](#_tqcqjv38h9xn) **4**

[**Код програми з коментарями**](#_mo6hoedsjpn4) **5**

[**Результат роботи програми**](#_wqyujciste2l) **7**

[**Висновки**](#_xaqol1c1o5mc) **8**

# 

# 

# **Постановка задачі**

Розробити програмне забезпечення для реалізації класичного нейрону.

Передбачити режим навчання класичного нейрону на одному навчальному

прикладі та режим розпізнавання.

Навчальний приклад:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x0 | x1 | x2 | x3 | yr |
| 1 | 3 | 5 | 7 | 0.3 |

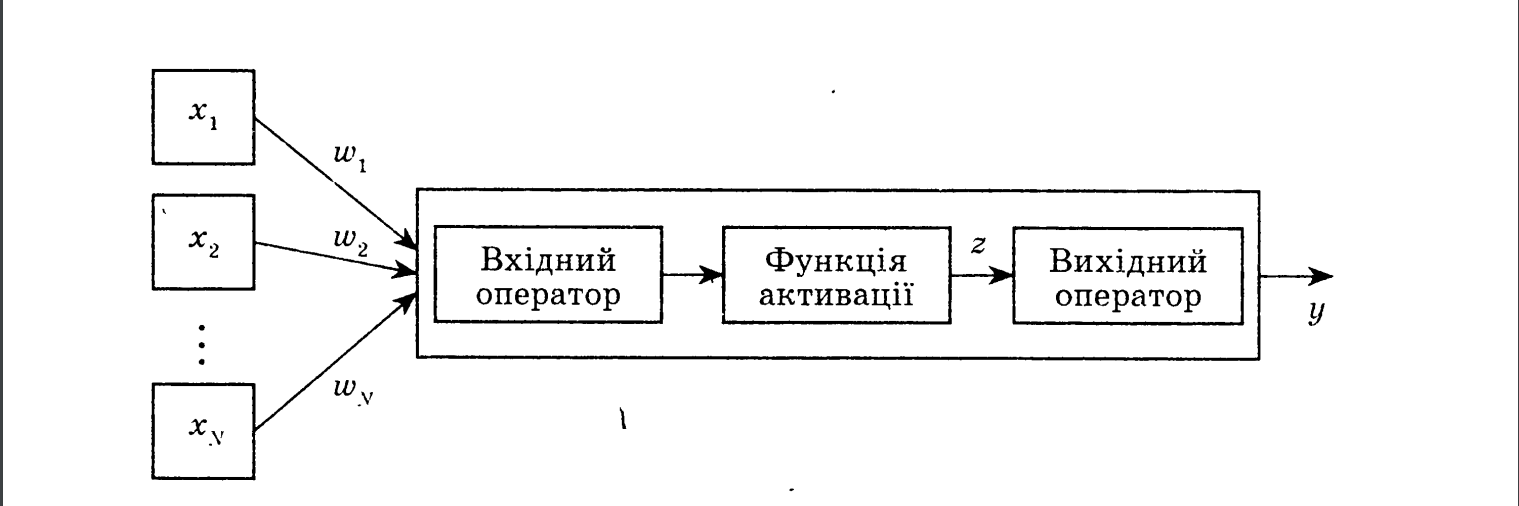
* Вагові коефіцієнти ініціалізовані випадково
* Допустима помилка 0.1

# 

# **Короткі теоретичні відомості**

Штучні нейрони, що також називаються нейронними клітинами, модулями, вузлами, моделюють структуру й функції біологічних нейронів.

Структура штучного нейрона:



Вхідними сигналами штучного нейрона є вихідні сигнали інших нейронів, кожний з яких взятий зі своєю вагою, аналогічною синаптичній силі.

Вхідний оператор перетворює зважені входи й подає їх на оператор активації. Вихідний сигнал у являє собою перетворений вихідним оператором вихідний сигнал оператора активації.

Більш детально по кожному з елементів.

**Вхідний оператор** в даній роботі представлений функцією зважених входів, яка

має наступний вигляд:

**Функція активації** описує перетворення нейрона зі стану 𝑧(𝑘) в момент часу k у

новий стан 𝑧(𝑘 + 1).

В даній роботі вона представлена логістичною або сигмоїдальною функцією:

# **Код програми з коментарями**

Класичний нейрон реалізован за допомогою класу ClassicNeuron, що має всі необхідні методи для роботи з ним.

class ClassicNeuron:

def \_\_init\_\_(self, x, y\_r, dd):

self.\_x = x

self.y\_r = y\_r

self.dd = dd

self.w = [random() for i in range(N)]

self.i = 1

self.y\_i = None

self.x\_s = 0

@property

def x(self):

return self.\_x

@x.setter

def x(self, x):

if len(x) != N:

raise ValueError(f"Dim {N}")

self.\_x = x

def sigmoid(self):

return 1 / (1 + exp(-self.x\_s)) # sigmoid activation

def weighted\_sum(self):

return sum([self.\_x[j] \* self.w[j] for j in range(N)]) # weighted sum

def train(self):

# weights update, training

while True:

self.x\_s = self.weighted\_sum()

self.y\_i = self.sigmoid()

dn = abs((self.y\_r - self.y\_i) / self.y\_r)

print(f"\n Iteration: {self.i}\n y\_i = {self.y\_i} \n dn = {dn}")

if dn <= self.dd:

print(

f"\n New weights: w1:{self.w[0]} w2:{self.w[1]} w3:{self.w[2]} w4:{self.w[3]}"

)

print(f"y = {self.y\_i} \n")

return self.w

else:

q = self.y\_i \* (1 - self.y\_i) \* (self.y\_r - self.y\_i)

for j in range(N):

dw = self.\_x[j] \* q

self.w[j] += dw

self.i += 1

В функції ініціалізації задаються всі необхідні параметри, а саме вхідні сигнали, допустиму похибку, очікуваний результат, випадково задані ваги для вхідних сигналів.

В функції активації реалізована сигмоїда.

Функція train реалізує саме навчання класичного нейрону. Тут ми знаходимо y\_i, яке потім порівнюємо з y\_r та на основі цього робимо висновок тренування завершено чи ні.

print("Task1")

neuron = ClassicNeuron([1, 3, 5, 7], 0.3, 0.1)

neuron.train()

neuron.x = [1, 3.5, 5.4, 7.2]

neuron.weighted\_sum()

print(f"End: {neuron.sigmoid()}")

Тут ми бачимо, що створюється екземпляр класу з вхідними даними. Потім він тренується. І після цього ми запускаємо режим розпізнавання.

# **Результат роботи програми**

Task1

Iteration: 1

y\_i = 0.9993894035372826

dn = 2.331298011790942

Iteration: 2

y\_i = 0.9993671307455259

dn = 2.331223769151753

Iteration: 3

y\_i = 0.9993431895727328

dn = 2.3311439652424433

Iteration: 4

y\_i = 0.9993173867231133

dn = 2.331057955743711

Iteration: 5

y\_i = 0.999289498025335

dn = 2.3309649934177834

Iteration: 6

y\_i = 0.9992592620435063

dn = 2.330864206811688

Iteration: 7

y\_i = 0.9992263720405166

dn = 2.3307545734683885

Iteration: 8

y\_i = 0.9991904657778409

dn = 2.3306348859261368

Iteration: 9

y\_i = 0.9991511124433999

dn = 2.3305037081446667

Iteration: 10

y\_i = 0.9991077957220624

dn = 2.3303593190735414

Iteration: 11

y\_i = 0.9990598916187631

dn = 2.3301996387292108

Iteration: 12

y\_i = 0.9990066390433712

dn = 2.3300221301445707

Iteration: 13

y\_i = 0.9989471002580559

dn = 2.329823667526853

Iteration: 14

y\_i = 0.9988801068873062

dn = 2.3296003562910212

Iteration: 15

y\_i = 0.9988041849840577

dn = 2.329347283280192

Iteration: 16

y\_i = 0.998717449084513

dn = 2.329058163615043

Iteration: 17

y\_i = 0.9986174492836111

dn = 2.32872483094537

Iteration: 18

y\_i = 0.9985009452905598

dn = 2.328336484301866

Iteration: 19

y\_i = 0.9983635636452941

dn = 2.327878545484314

Iteration: 20

y\_i = 0.9981992616837299

dn = 2.3273308722790995

Iteration: 21

y\_i = 0.9979994594367964

dn = 2.3266648647893216

Iteration: 22

y\_i = 0.9977515750197309

dn = 2.3258385833991033

Iteration: 23

y\_i = 0.9974364308454728

dn = 2.3247881028182427

Iteration: 24

y\_i = 0.997023383924278

dn = 2.3234112797475936

Iteration: 25

y\_i = 0.9964605038442014

dn = 2.3215350128140044

Iteration: 26

y\_i = 0.9956528925270804

dn = 2.318842975090268

Iteration: 27

y\_i = 0.994408884979628

dn = 2.3146962832654268

Iteration: 28

y\_i = 0.9922836037180666

dn = 2.3076120123935557

Iteration: 29

y\_i = 0.9880074133004393

dn = 2.2933580443347976

Iteration: 30

y\_i = 0.9764921048622186

dn = 2.254973682874062

Iteration: 31

y\_i = 0.9185037634611766

dn = 2.061679211537255

Iteration: 32

y\_i = 0.18742485713483464

dn = 0.37525047621721785

Iteration: 33

y\_i = 0.4933342646019396

dn = 0.644447548673132

Iteration: 34

y\_i = 0.01652915018081741

dn = 0.9449028327306086

Iteration: 35

y\_i = 0.024153325394825658

dn = 0.9194889153505811

Iteration: 36

y\_i = 0.04098339386433398

dn = 0.8633886871188867

Iteration: 37

y\_i = 0.09132241202768218

dn = 0.695591959907726

Iteration: 38

y\_i = 0.3009036500880601

dn = 0.0030121669602003096

New weights: w1:0.15275359632815289 w2:0.28137098834971264 w3:0.40149473528563806 w4:-0.5496198137220689

y = 0.3009036500880601

End: 0.3009036500880601

# **Висновки**

Розроблено програмне забезпечення, що реалізує логіку класичного нейрона. Кількість ітерацій напряму залежить від вагів, що в нашому випадку, задані рандомно при ініціалізації класу.

На виході отримали y, що дорівнює ~0.300903.