VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

Praktinė užduotis Nr. 1 (Dirbtinis neuronas)

Atliko: Programų sistemų 4 k. 1 gr. stud. Rokas Petrauskas

VILNIUS, 2023

Turinys

Tikslas

Šios užduoties tikslas yra išanalizuoti dirbtinio neurono modelio veikimo principus jį realizuojant kompiuteriniu modeliu.

Įvestys ir siekiamas rezultatas

Dirbtinis neuronas naudoja 2 įėjimus (x_1, x_2), jiems parinkti atitinkami svoriai (w_1, w_2) bei poslinkis (w_0), kad modelis grąžintų reikiamą klasę (t).

Duomenys		**Klasė**
x_1	x_2	t
−0,2	0,5	0
0,2	−0,7	0
0,8	−0,8	1
0,8	1	1

1 lentelė. Duomenys klasifikavimui

Programos kodas

```
import numpy as np

class ArtificialNeuron:
    """
    ArtificialNeuron Klasė realizuojanti dirbtinį neuroną.
    """

    def __init__(self):
        """
        __init__ metodas, šiuo metu nepriskiriame pradinių svorių ir poslinkio.
        """
        pass

    def activate(self, inputs, activation_fun, weights, bias):
        """
        activate metodas sumuoja svorių ir įėjimų sandaugas ir prideda poslinkio
vertę
        po to parenkama norima aktyvacijos funkcija bei grąžinama numatoma klasė.
        """
        weighted_sum = np.dot(inputs, weights) + bias
        if activation_fun == "sigmoid":
            output = self.sigmoid(weighted_sum)
        elif activation_fun == "step":
            output = self.step(weighted_sum)
        return output
```

```
    def sigmoid(self, x):
        """
        sigmoid metodas naudojamas activate funkcijoje, tam, kad apibrėžtų išeities
reikšmę pagal logistinę funkciją.
        """
        return 1 / (1 + np.exp(-x))

    def step(self, x):
        """
        step metodas naudojamas activate funkcijoje, tam, kad apibrėžtų išeities
reikšmę pagal slenkstinę funkciją.
        """
        return (1 if x >= 0 else 0)

#duomenys klasifikavimui: x1 ir x2 vertės
inputs = [[-0.2, 0.5],
          [0.2, -0.7],
          [0.8, -0.8],
          [0.8, 1.0]]

#siekiamos klases
target_classes = [0, 0, 1, 1]
# gaunamų rinkinių [w0, w1, w2] skaičius
num_sets = 5

def validate_weights_bias(inputs, target_classes, activation_fun, num_sets):
    """
        Tikrina ir grąžina nurodytą kiekį tinkamų svorių ir poslinkio rinkinių,
kurie tenkina duomenų klasifikavimo sąlygas, pasitelkiant antrą užduotyje aprašytą
metodą (atsitiktinai generuojamos reikšmės)
    """
    valid_results = []

    for _ in range(num_sets):
        valid_weights = None
        valid_bias = None

        while True: # generuojamos atsitiktinės reikšmės intervale (-1, 1)
            weights = np.random.uniform(-1, 1, 2)
            bias = np.random.uniform(-1, 1)

            neuron = ArtificialNeuron()
            # paduodame generuotas svorio ir poslinkio reikšmes funkcijai
            all_correct = True
            for i, input_data in enumerate(inputs):
                x = neuron.activate(input_data, activation_fun, weights, bias)
                if round(x) != target_classes[i]: # tikriname ar neurono gauta
klasė sutampa su užduotyje pateikiama klase
```

```
                    all_correct = False
                    break  # jei bent vienas neteisingas rezultatas, nutraukiame
tikrinimą

            if all_correct:
                valid_weights = list(weights)
                valid_bias = bias
                break

        valid_results.append((valid_weights, valid_bias))

    return valid_results

valid_results = validate_weights_bias(inputs, target_classes, "sigmoid", num_sets)

testing_neuron = ArtificialNeuron()

# Išvedame tinkamas reikšmes
for i, (weights, bias) in enumerate(valid_results):
    print(f"Rinkinys {i + 1}:")
    print("w0 (poslinkis):", bias)
    print("w1 (x1 svoris):", weights[0])
    print("w2 (x2 svoris):", weights[1])
    print()

    print("Neurono prognozė su step funckija: ")
    for i1 in range(4):
        x = testing_neuron.activate(inputs[i1], "step", [weights[0], weights[1]],
bias)
        print("vertė: " + str(x))
    print()

    print("Neurono prognozė su sigmoid funckija: ")
    for i1 in range(4):
        x = testing_neuron.activate(inputs[i1], "sigmoid", [weights[0],
weights[1]], bias)
        print("-------------------------")
        print("vertė: " + str(x))
        print("klasė: " + str(round(x)))
    print()
```

Pavyzdinė programos išvestis:

```
Rinkinys 1:

w0 (poslinkis): -0.6495941112570369
w1 (x1 svoris): 0.9893957400343105
w2 (x2 svoris): 0.002643853429404208

Neurono prognozė su step funckija:
vertė: 0
vertė: 0
vertė: 1
vertė: 1

Neurono prognozė su sigmoid funckija:
------------------------
vertė: 0.30024082604235275
klasė: 0
------------------------
vertė: 0.3885133334947147
klasė: 0
------------------------
vertė: 0.5348950295114313
klasė: 1
------------------------
vertė: 0.5360787699639543
klasė: 1
```

Svorių ir poslinkio pasirinkimas

Svorių ir poslinkių nustatymas buvo darytas dvejais būdais. Pirmasis – matematiškas sprendimas, antrasis – realizuotas programiniame kode.

Matematiškas svorių ir poslinkio radimas

Svoriai ir poslinkis buvo nustatytas naudojant 4 lygčių sistemą su 3 nežinomais kintamaisiais. Lygtys sudarytos įstatant x_1, x_2 reikšmes iš 1 lentelės bei sudaugintos su nežinomaisiais w_1, w_2, prie jų pridėjus nežinomąjį w_0 ir sulyginant reikšmę su klasės reikšme kuri būtų gauta naudojant slenkstinę funkciją. Lygtys atvaizduotos 1 pav., kur kintamieji a = w_1, b = w_2, c = w_0.

$-0.2\,a + 0.5\,b + c < 0$
$0.2\,a - 0.7\,b + c < 0$
$0.8\,a - 0.8\,b + c \geq 0$
$0.8\,a + b + c \geq 0$

Pav. 1. Lygčių sistema

Lygčių sistemos rezultatai yra intervalai aprašyti 2 pav. Pasirinkus a reikšmę galima išskaičiuoti kitų intervalų reikšmes.

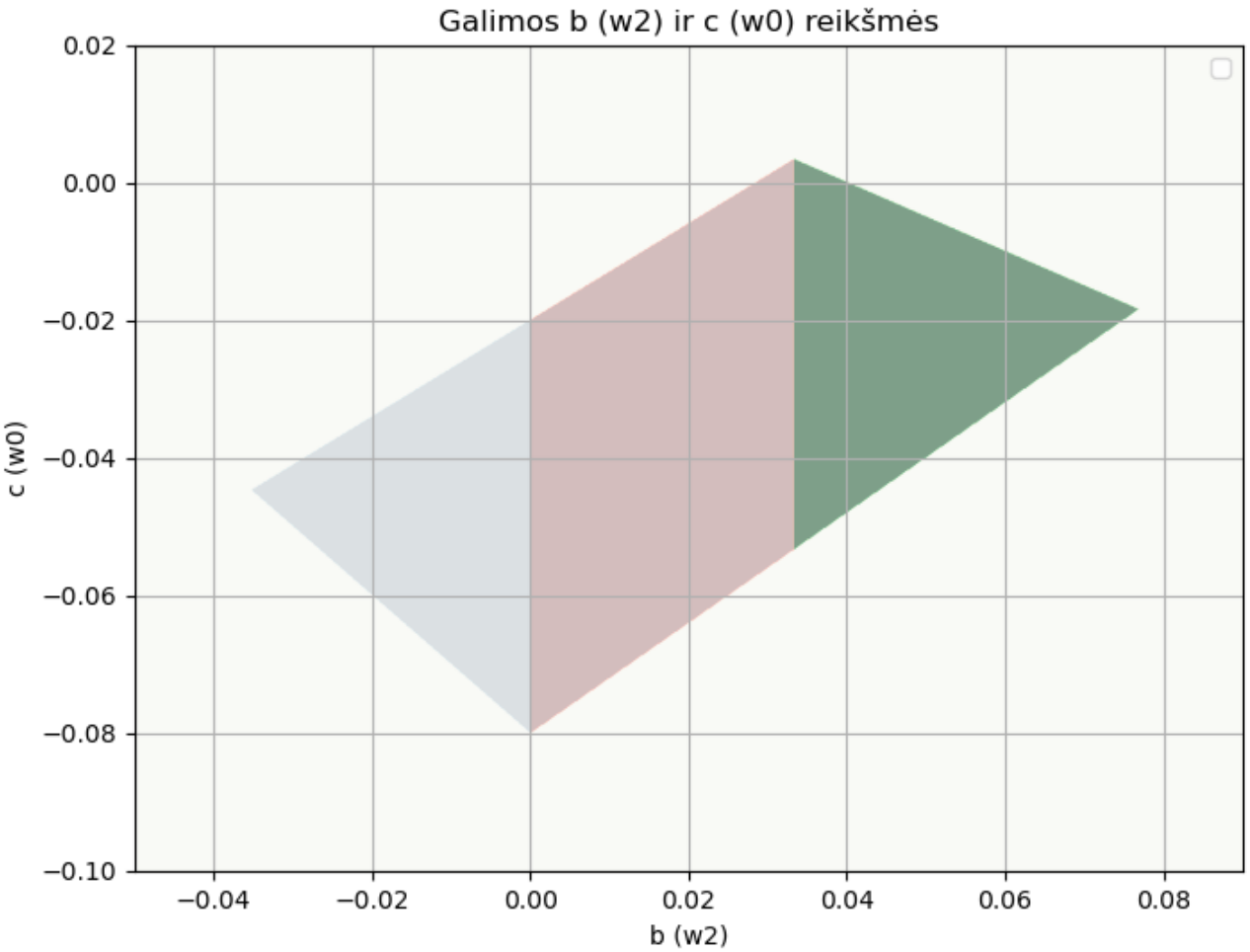
$$a > 0 \text{ and } -\frac{6a}{17} < b \leq 0 \text{ and } \frac{1}{5}(-4a - 5b) \leq c < \frac{1}{10}(7b - 2a)$$

$$a > 0 \text{ and } 0 < b \leq \frac{a}{3} \text{ and } -\frac{4}{5}(a - b) \leq c < \frac{1}{10}(7b - 2a)$$

$$a > 0 \text{ and } \frac{a}{3} < b < \frac{10a}{13} \text{ and } -\frac{4}{5}(a - b) \leq c < \frac{1}{10}(2a - 5b)$$

Pav. 2. Lygčių sistemos rezultatai

Lygčių sistemos rezultatus galima atvaizduoti grafiškai. Fiksavau kintamąjį a (w_1) su reikšme 0.1. Nelygybes tenkinantys kintamieji b (w_2) ir c (w_0) atvaizduoti 3 paveikslėlyje. Pilka spalva atitinka pirmąją nelygybių sistemą, rusva – antrąją, žalia – trečiąją.



Pav. 3. Galimos b (w_2) ir c (w_0) reikšmės kai a (w_1) = 0,1

Norėdamas įsitikinti, kad grafikas teisingas galiu įsistatyti apibrėžtas reikšmes į gautas nelygybes pvz.: imkime tašką (0.02, -0.04). Šis taškas yra antros nelygybių sistemos plote, todėl taškus statau į ją.

0,1 > 0 ∧ 0 < 0,02 ≤ 0,1/3 ∧ -4/5 (0,1 - 0,02) ≤ -0,04 < 1/10 (7 · 0,02 - 2 · 0,1)

Nesunku įsitikinti, kad ši sąlyga teisinga. Galiu pridėti testavimo funkciją anksčiau aprašytai programai:

```
m_testing_neuron = ArtificialNeuron()

for i1 in range(4):
    x = testing_neuron.activate(inputs[i1], "sigmoid", [0.1, 0.02], -0.04)
    print("-------------------------")
    print("vertė: " + str(x))
    print("klasė: " + str(round(x)))
print()
```

Aktyvuodamas neuroną su tomis pačiomis w_2, w_0 reikšmėmis (0.02, -0.04), tuo tarpu w_1 fiksuotas kaip 0.1, gaunu tokią išvestį:

```
-------------------------
vertė: 0.4875026035157896
klasė: 0
-------------------------
vertė: 0.49150081873868723
klasė: 0
-------------------------
vertė: 0.5059997120165879
klasė: 1
-------------------------
vertė: 0.51499550161941
klasė: 1
```

Programiškas svorių ir poslinkio radimas

Programoje svoriai ir poslinkiai randami juos generuojant atsitiktinai intervale (-1, 1). Gautos reikšmės naudojamos su kitomis įvestimis aktyvacijos funkcijoje, gautas rezultatas tikrinamas su lentelėje pateiktomis klasėmis, jei visos eilutės tenkinamos, šios vertės išsaugomos.

Galimos svorių ir poslinkių reikšmės

2 Lentelėje pateikti poslinkių ir svorio reikšmių rinkiniai leidžia teisingai klasifikuoti tiek naudojant sigmoidinę, tiek slenkstinę aktyvacijos funkcijas.

	a (w_1)	**b (w_2)**	**c (w_0)**
1.	0.7213457556606508	0.1704504761452037	-0.11092095756492926
2.	0.5671743065028112	0.25342664526801917	-0.2276034623364065
3.	0.31059769841942453	-0.029204211457044993	-0.11574811158695142
4.	0.12908954188250998	0.06726440845010595	-0.022173818779343435
5.	0.7673572413543792	-0.15164490523848895	-0.41903029711583617
6.	0.8225533569611885	0.0617744802594975	-0.24939167090975967
7.	0.7773584507774902	0.0727781659781801	-0.45037468303095607
8.	0.4350794113445733	0.19336822704220302	-0.0761308539578649
9.	0.36603237604413486	0.023935343415510957	-0.06667825084189238
10.	0.7544663239132896	0.32496186973815444	-0.08050382545723256

2 lentelė. Svorių ir poslinkio reikšmių rinkiniai