I užduotis (Dirbtinis neuronas)

Pijus Petkevičius

2023 m. lapkričio 27 d.

Turinys

1 Įvadas

1.1 Tikslas

Užduoties tikslas- išanalizuoti dirbtinio neurono modelį, jo veikimo principus, susipažinti su dirbtinių neuronų pagrindais.

1.2 Uždaviniai

- Parašyti pasirinkta programavimo kalba dirbtinio neurono realizaciją. Programa turi rasti poslinkio ir svorių reikšmes, naudojant slenkstinę ir sigmoidinę funkcijas, kad 1 lentelėje esantys duomenys būtų tinkamai klasifikuoti.
- Užrašyti nelygybių sistemą, kurią reikia spręsti, norint parinkti tinkamas poslinki ir svorių reikšmes, kai aktyvacijos funkcija yra slenkstinė.
- Patikrinti grafiniu būdu nubraižytą nelygybių sistemą, ar sprendiniai, gauti iš grafiko, tenkina nelygybių sistemą.

2 Klasifikavimo duomenys ir klasė

Šiame darbe 1 lentelėje atvaizduotiems duomenims ieškomas poslinkis ir svoriai, su kuriais būtų teisingai klasifikuojami.

1 lentelė: Duomenys klasifikavimui

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

3 Programos kodas

```python
import numpy as np

# Paimamos įvesties ir svorių reikšmės, sudauginama i-oji įvesties reikšmė su i-ojo svorio reikšme ir taip su visomis įvesties reikšmėmis, jos susumuojamos ir grąžinama reikšmė
# (arba galima panaudoti scaliarinią sandaugą)
def countA(inputValues: [float], weights: [float] ) -> float:
    return np.dot(inputValues, weights)

# Apskaičiuojama slenkstinės funkcijos reikšmė
def threshold(x) -> int:
    return 1 if x >= 0 else 0

# Apskaičiuojama sigmoidinės funkcijos reikšmė
def sigmoid(x) :
    return round(1 / (1 + np.exp(-x)))

# Paimamas įvesties duomenų masyvas, atskiriama įvesties reikšmė nuo išvesties reikšmės. Prie įvesties pridedama 1(poslinkio reikšmę norint traktuoti kaip svorį).
def separateInputOutputAndAddBias(inputValues: np.ndarray) -> (np.ndarray, [float]):
    output = inputValues[:, -1]
    removedOutput = np.delete(inputValues, -1, axis=1)
    inputValues = np.concatenate((np.array([1 for _ in range(removedOutput.shape[0])])[:, np.newaxis], removedOutput), axis=1)
    return inputValues, output

# Šioje funkcijoje kiekvienai įvesties reikšmei pritaikoma countA funkcija kartu su aktyvacijos funkcija ir tikrinama ar gauta reikšmė sutampa su išvesties reikšme.
def applyWeightstoAllInputs(inputValues: np.ndarray, output: [float], weights:[float], activationFunction: callable) -> bool:
    countMatches = 0
    for i in range(inputValues.shape[0]):
        if activationFunction(countA(inputValues[i], weights)) == output[i]:
            countMatches += 1

    return countMatches == inputValues.shape[0]

def readDataFromFile(filename: str) -> np.ndarray:
```

```
33      return np.loadtxt(filename, delimiter=',',skiprows=1)
34
35  # Sugeneruojami visi galimi svorių variantai intervale [-10, 10]
    ↪  su žingsniu 1
36  def generateWeigths() -> [float, float, float]:
37      values = np.arange(-10, 10, 1)
38      weights = []
39      for i in values:
40          for j in values:
41              for k in values:
42                  weights.append([i, j, k])
43      return weights
44
45  # Funkcija paima visus sugeneruotus svorių kombinacijas, pritaiko
    ↪  įvesties reikšmėms, patikrina ar applyWeightstoAllInputs
    ↪  grąžina True, jei taip, tai prideda svorių kombinaciją prie
    ↪  tinkamų svorių masyvo
46  def weightSearchFunction(input: np.ndarray, output: float,
    ↪  weights: [float, float, float], function: callable) ->
    ↪  [float, float, float]:
47      validWeights = []
48      for weight in weights:
49          if applyWeightstoAllInputs(input, output, weight,
            ↪  function):
50              validWeights.append(weight)
51      return validWeights
52
53
54  inputData = readDataFromFile('input.csv')
55  input, output = separateInputOutputAndAddBias(inputData)
56  generateWeigths = generateWeigths()
57
58  print("Threshold")
59  print(weightSearchFunction(inputData, output, generateWeigths,
    ↪  threshold))
60
61  print("Sigmoid")
62  print(weightSearchFunction(inputData, output, generateWeigths,
    ↪  sigmoid))
```

4 Svorių ir poslinkių parinkimas

4.1 Parinkimo strategija

Su python programavimo kalba sukurta **generateWeights()** funkcija, kuri sugeneruoja visas intervale [−10,10) reikšmes(eil. 36-43). Vėliau su **weightSearchFunction(...)** funkcija kiekviena svorių kombinacija yra pritaikoma įvesties duomenims, patikrinama ar pritaikius aktyvacijos funkciją(slenkstinė eil. 9-10 arba sigmoidinė eil. 13-14, visos gautos reikšmės sutampa su tikėtina klasės reikšme (eil. 46-51).

4.2 Rezultatai

Su slenkstine aktyvacijos funkcija gauti 52, o su sigmoidine 72 svorių ir poslinkių rinkiniai(Jei sigmoidinė funkcija nebūtų apvalinama, būtų gaunama 0 svorių ir poslinkio reikšmių kombinacijų),

4.2.1 Rezultatai su slenkstine aktyvacijos funkcija

2 lentelėje pateiktos svorių ir poslinkio reikšmės, su kuriomis slenkstinė aktivacijos funkcija tinkamai suklasifikavo duomenis.

2 lentelė: Poslinkio ir svorių rinkiniai su slenkstine aktyvacijos funkcija

w_0	w_1	w_2
3	1	-1
3	1	0
3	1	1
3	1	2
6	2	-1

4.2.2 Rezultatai su sigmoidine aktyvacijos funkcija

3 lentelėje pateiktos svorių ir poslinkio reikšmės, su kuriomis sigmoidinė aktivacijos funkcija tinkamai suklasifikavo duomenis.

Šioje užduotyje sigmoidinės funkcijos reikšmė yra apvalinama iki sveikojo skaičiaus vienetų tikslumu(0 arba 1). Dėl šios priežasties naudojant sigmoidinės aktyvacijos funkciją, buvo gauta daugiau svorių ir poslinkio reikšmių; didesnė paklaida.

3 lentelė: Poslinkio ir svorių rinkiniai su sigmoidine aktyvacijos funkcija

w_0	w_1	w_2
5	2	2
5	2	3
5	2	4
5	2	5
6	2	4

4.3 Nelygybių sistema

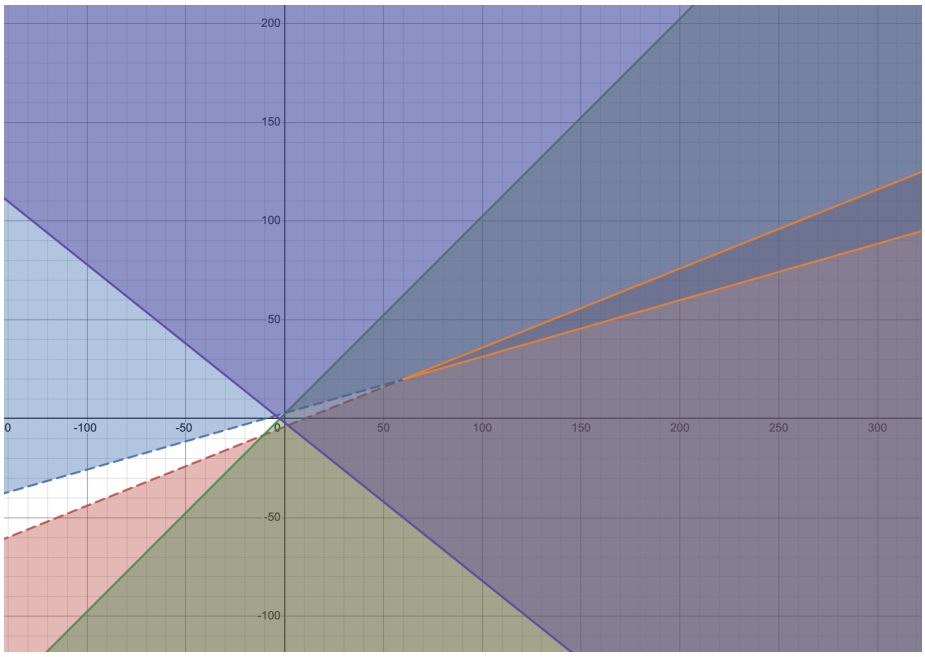
Norint teisingai parinkti svorių ir poslinkio reikšmes, kai aktyvacijos funckija yra slenkstinė, reikėtų išspręsti tokią nelygybių sistemą:

$$w_0 - 0{,}2w_1 + 0{,}5w_2 < 0,$$
$$w_0 + 0{,}2w_1 - 0{,}7w_2 < 0,$$
$$w_0 + 0{,}8w_1 - 0{,}8w_2 \geq 0,$$
$$w_0 + 0{,}8w_1 + 1w_2 \geq 0$$

Kadangi nelygybių sistema sprendžiama grafiškai(nenorime naudoti 3d erdvės), parinkime w_0 reikšmę $w_0 = 2$ Gauta tokia nelygybių sistema:

$$2 - 0{,}2w_1 + 0{,}5w_2 < 0,$$
$$2 + 0{,}2w_1 - 0{,}7w_2 < 0,$$
$$2 + 0{,}8w_1 - 0{,}8w_2 \geq 0,$$
$$2 + 0{,}8w_1 + 1w_2 \geq 0$$

Šios nelygybių sistemos sprendinys matomas 1 pav. pažymėta oranžine spalva.



1 pav.: Nelygybių sistemos sprendinys

4.4 Nelygybių sistemos sprendinio patikrinimas

. Iš 1 pav. parinkti 2 sprendiniai, tenkinantys nelygybių sistemą:

Kai $w_1 = 150$ ir $w_2 = 50$:

$$2 - 0{,}2 \cdot 150 + 0{,}5 \cdot 50 = 2 - 30 + 25 = -3 < 0,$$
$$2 + 0{,}2 \cdot 150 - 0{,}7 \cdot 50 = 2 + 30 - 35 = -3 < 0,$$
$$2 + 0{,}8 \cdot 150 - 0{,}8 \cdot 50 = 2 + 120 - 40 = 82 \geq 0,$$
$$2 + 0{,}8 \cdot 150 + 1 \cdot 50 = 2 + 120 + 50 = 172 \geq 0$$

Kai $w_1 = 250$ ir $w_2 = 75$:

$$2 - 0{,}2 \cdot 250 + 0{,}5 \cdot 75 = 2 - 50 + 37.5 = -10.5 < 0,$$
$$2 + 0{,}2 \cdot 250 - 0{,}7 \cdot 75 = 2 + 50 - 52.5 = 0.5 < 0,$$
$$2 + 0{,}8 \cdot 250 - 0{,}8 \cdot 75 = 2 + 200 - 60 = 142 \geq 0,$$
$$2 + 0{,}8 \cdot 250 + 1 \cdot 75 = 2 + 200 + 75 = 277 \geq 0$$

5 Išvados

- Naudojant sigmoidinę aktyvacijos funkciją, galime gauti reikšmes intervale (0,1), kas tam tikruose uždaviniuose būtų žymiai tiksliau nei slenkstinė aktyvacijos funkcija.
- Nors ir parinkus intervala $[-10,10)$ su žingsniu 1, slenkstinės ar sigmoidinės aktyvacijos funkcijos pagalba gaunama 52 ir 72 reikšmės. Didinant intervalą ar mažinant žingsnio reikšmę, galime tikėtis didesnio svorių ir poslinkių kiekio.