Vilniaus Universitetas

Dirbtinis neuronas

Dirbtinio intelekto pagrindai

1 užduotis

Darbą atliko:

Dovydas Martinkus

Duomenų Mokslas 4 kursas 2 gr.

Vilnius, 2022

Turinys

1 Tikslas ir uždaviniai

Tikslas: išanalizuoti dirbtinio neurono modelį ir jo veikimo principus

Uždaviniai:

Sudaryti dirbtinio neurono modelį.

Pasirinkta programavimo kalba realizuoti dirbtinius neuronus, naudojančius slenkstinę ir sigmoidinę aktyvacijos funkcijas.

Užrašyti kokią nelygybių sistemą reikia spręsti, norint teisingai parinkti svorių ir poslinkio reikšmes ir nubraižyti nelygybių sistemos grafiką.

2 Užduoties ataskaita

Užduotyje naudojami duomenys pateikti 1 lentelėje.

1 lentelė Duomenys klasifikavimui

Duomenys		**Klasė**
x1	***x2***	**t**
-0,3	0,6	0
0,3	-0,6	0
1,2	-1,2	1
1,2	1,2	1

Dirbtinis neuronas realizuotas naudojant „Python“ programavimo kalbą. Svorių ir poslinkio parinkimui buvo naudojama perrinkimo strategija, naudojant paieškos tinklelį, sudarytą visus svorius ir poslinkį keičiant nuo -10 iki 10 imant 0,1 dydžio žingsnį. Duomenų nuskaitymo ir paieškos tinklelio sudarymo kodas pateiktas žemiau.

```
import numpy as np
import math

# turimi duomenys
data = [[-0.3,0.6,0],
        [0.3,-0.6,0],
        [1.2,-1.2,1],
        [1.2,1.2,1]]

# sudaromos svorių kombinacijos kiekvienam kintant nuo -10 iki 10 (viso 200*200*200
galimų variantų)
weights = [(i,j,k)
     for i in np.arange(-10, 10, 0.1)
     for j in np.arange(-10, 10, 0.1)
     for k in np.arange(-10, 10, 0.1)]
```

Žemiau pateiktas programinis kodas, skirtas tinkančių parametrų reikšmių paieškai, naudojant slenkstinę aktyvacijos funkciją.

```
# su kiekviena duomenų eilute išsaugoma su kuriais svorių rinkiniais gaunama teisinga
dirbtinio neurono išėjimo reikšmė
def weight_check_threshold(row,weights):
    valid = []
    for weight in weights:
        a = weight[0] + row[0]*weight[1] + row[1]*weight[2]
        if (row[2] == 0 and a < 0) or (row[2] == 1 and a >= 0):
            valid.append(weight)
    return valid
```

```
# kiekvienai eilutei paduodami visoms prieš tai tikrintoms eilutėms tikę svorių
rinkiniai
valid_weights_threshold = weights
solution = []
for row in data:
    solution = weight_check_threshold(row,valid_weights_threshold)
    valid_weights_threshold = solution

valid_weights_threshold = [(round(i,1),round(j,1),round(k,1)) for (i,j,k) in
solution]


import random
print(len(valid_weights_threshold)) # tikusių svorių rinkinių skaičius
print(len(valid_weights_threshold) / 200**3) # dalis tikusių iš visų tikrintų svorių
rinkinių


random.seed(10)
random.choices(valid_weights_threshold,k=5)

228000
0.0285

[(-3.4, 7.2, -1.3),
 (-4.3, 7.3, 0.8),
 (-3.4, 9.4, -0.6),
 (-6.0, 8.8, 0.8),
 (-1.9, 6.8, 2.5)]
```

Iš viso rasta 228000 skirtingų tinkančių svorių ir poslinkio reikšmių rinkinių. Šis skaičius lygus beveik 3 % visų tikrintų svorių ir poslinkio reikšmių rinkinių. Atsitiktinai parinkti ir lentelėje pavaizduoti 5 tinkančių parametrų reikšmių rinkiniai (2 lentelė).

2 lentelė Tinkančių svorių ir poslinkio reikšmių rinkinių pavyzdžiai (su slenkstine aktyvacijos funkcija)

w0	***w1***	***w2***
-3,4	-7,2	-1,3
-4,3	7,3	0,8
-3,4	9,4	-0,6
-6,0	8,8	0,8
-1,9	6,8	2,5

Beveik toks pat programinis kodas naudotas ir tinkančių parametrų reikšmių paieškai, naudojant sigmoidinę aktyvacijos funkciją. Nesunku pastebėti, kad jeigu sigmoidinės funkcijos reikšmė suapvalinama iki sveiko skaičiaus, gaunami identiški parametrų reikšmių rinkiniai kaip ir naudojant slenkstinę aktyvacijos funkciją. Įdomumo dėlei

pasirinkta ieškoti tik tokių parametrų reikšmių rinkinių, su kuriais klasė 1 priskiriama jei sigmoidinės funkcijos reikšmė didesnė už 0,9 ir 0 jeigu funkcijos reikšmė mažesnė už 0,1.

```
def weight_check_sigmoid(row,weights):
    valid = []
    for weight in weights:
        a = weight[0] + row[0]*weight[1] + row[1]*weight[2]

        a = 1/(1+math.exp(-a))

        if fa > 0.9:
            fa = 1
        if fa < 0.1:
            fa = 0

        if fa == row[-1]:
            valid.append(weight)
    return valid
```

```
valid_weights_sigmoid = weights
solution = []
for row in data:
    solution = weight_check_sigmoid(row,valid_weights_sigmoid)
    valid_weights_sigmoid = solution

valid_weights_sigmoid = [(round(i,1),round(j,1),round(k,1)) for (i,j,k) in solution]
```

```
print(len(valid_weights_sigmoid))
print(len(valid_weights_sigmoid) / 200**3)

random.seed(10)
random.choices(valid_weights_sigmoid,k=5)
```

```
38053
0.004756625

[(-5.0, 9.6, 0.5),
 (-5.5, 7.9, -0.1),
 (-4.9, 7.0, 0.4),
 (-6.5, 8.9, -0.4),
 (-3.9, 7.9, 1.2)]
```

Šį karta rasti tik 38053 tinkantys svorių ir poslinkio reikšmių rinkiniai. Lentelėje pavaizduoti tinkančių svorių ir poslinkio reikšmių rinkinių pavyzdžiai (3 lentelė).

3 lentelė Tinkančių svorių ir poslinkio reikšmių rinkinių pavyzdžiai (su sigmoidine aktyvacijos funkcija)

w0	***w1***	***w2***
-5,0	9,6	0,5
-5,5	7,9	-0,1
-4,9	7,0	0,4
-6,5	8,9	-0,4
-3,9	7,9	1,2

Sudaryta, kokią nelygybių sistemą reiktų spręsti, norint teisingai parinkti svorių ir poslinkio reikšmes turimiems duomenims, kai aktyvacijos funkcija yra slenkstinė:

$$-0.3w_1 + 0.6w_2 + w_0 < 0$$

$$0.3w_1 - 0.6w_2 + w_0 < 0$$

$$1.2w_1 - 1.2w_2 + w_0 >= 0$$

$$1.2w_1 + 1.2w_2 + w_0 >= 0$$

Paprastumo dėlei pasirinkta laikyti, kad $w_2 = 0$. Dėl šios priežasties turimą nelygybių sistemą galima užrašyti paprasčiau:

$$-0.3w_1 + w_0 < 0$$

$$0.3w_1 + w_0 < 0$$

$$1.2w_1 + w_0 >= 0$$

$$1.2w_1 + w_0 >= 0$$

Kadangi trečioji ir ketvirtoji nelygybės sutampa, vieną iš jų pašaliname:

$$-0.3w_1 + w_0 < 0$$

$$0.3w_1 + w_0 < 0$$

$$1.2w_1 + w_0 >= 0$$

Pertvarkę tas pačias nelygybes turime:

$$w_0 < 0.3w_1$$

$$w_0 < -0.3w_1$$

$$w_0 >= -1.2w_1$$

Pasinaudoję nelygybių tranzityvumo savybe, papildomai turime kad:

$$-1.2w_1 \leq 0.3w_1$$

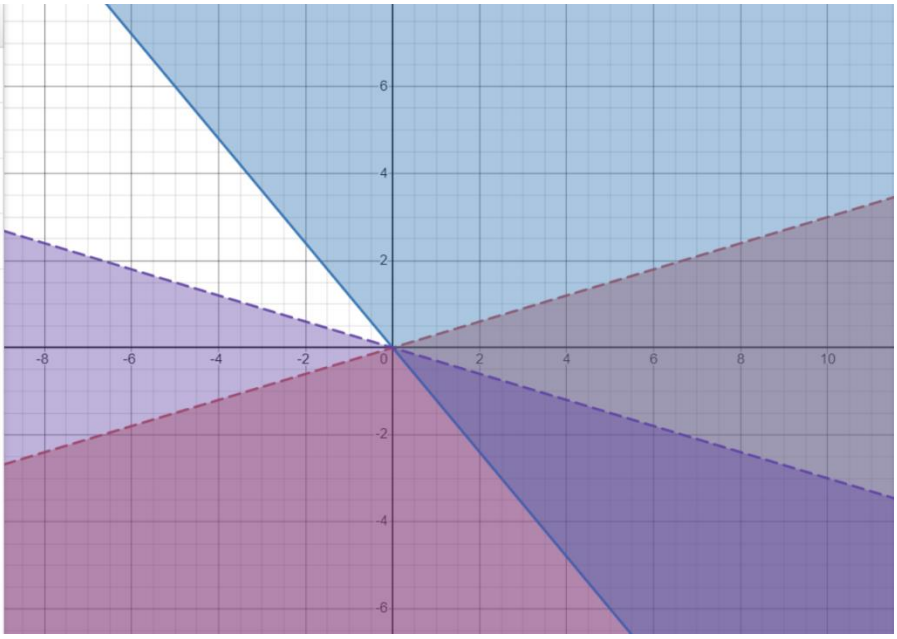
Perrašius gauname sąlygą:

$$w_1 \geq 0$$

Atsižvelgę į gautą sąlygą gauname, kad pirmoji nelygybė prieš tai turėtoje trijų nelygybių sistemoje yra mažiau griežta negu antroji, todėl ją galime pašalinti. Galiausiai gauname:

$$-1.2w_1 \leq w_0 < -0.3w_1$$
$$w_1 \geq 0$$

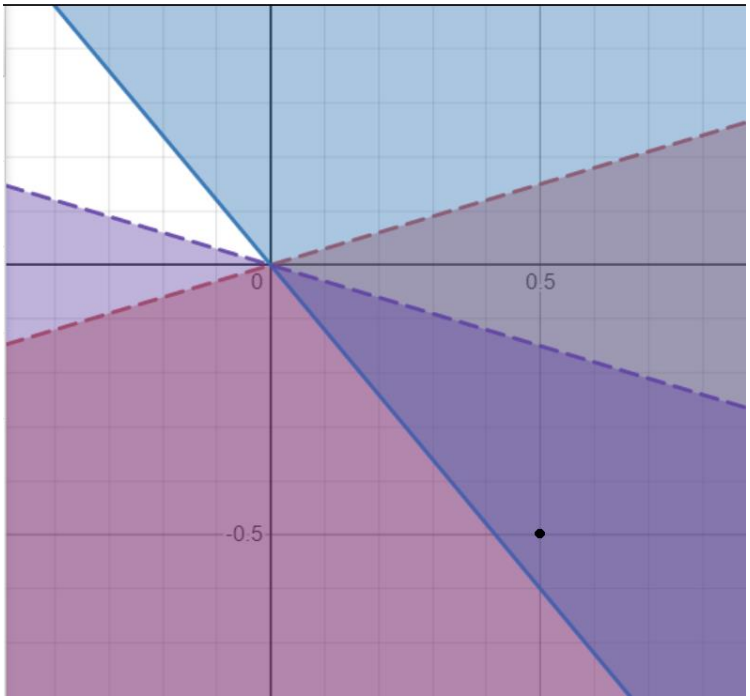
Tokia pati lygčių sistema išspręsta ir grafiniu būdu. Lygčių sistemos sprendinių aibė gaunama visų trijų spalvų susikirtimo zonoje (1 pav.). X ašis atitinka w_1, Y ašis – w_0.



1 pav. Grafinis sudarytos lygčių sistemos sprendimas

Naudojant grafiką galime patikrinti analitiniu būdu gautus rezultatus: w_1 reikmės yra tik neneigiamos, tuo tarpu w_2 reikšmės – neigiamos. Taip pat matoma, kad didėjant w_1 atitinkamai didėja ir galimų w_0 reikšmių intervalo dydis.

Iš grafiko galima nesunkiai pastebėti, kad sprendinių aibei priklauso taškas (0.5, -0.5) (2 pav.). Akivaizdu, kad $w_1 > 0$, be to, įstatę šią reikšmių porą į nelygybę $-1.2w_1 <= w_0 < -0.3w_1$ gauname teisingą nelygybę $-0.6 \leq -0.5 < -0.15$. Tai reiškia, kad grafiniu būdu gautas sprendinys tikrai yra nelygybių sistemos sprendinys.



2 pav. Grafiniu būdu gautas galimos nelygybių sistemos sprendimas.