VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Dirbtinis neuronas

1-oji skaitmeninio intelekto ir sprendimų priėmimų dalyko užduotis

Atliko: 4 kurso 5 grupės studentė
Gabrielė Žielytė (parašas)

Darbo vadovas: Prof., Dr. Olga Kurasova (parašas)

Vilnius – 2020

TURINYS

Tikslas

Šio darbo tikslas yra:

1. Sukurti dirbtinio neurono modelį. Į neuroną turi būti paduodamos įėjimų (angl. *input*) reikšmės, nurodoma aktyvacijos funkcija (realizuotos slenkstinė ir sigmoidinė funkcijos). Neuronas turi paskaičiuoti išėjimo reikšmę (angl. *output*).

2. Parašyti programą, kurioje keičiant svorių (w_1, w_2) ir slenksčio (w_0) reikšmes, naudojant slenkstinę bei sigmoidinę aktyvacijos funkcijas, nustatyti tokias svorių ir slenksčio reikšmes, kad gautųsi lentelėje (1 pav.) pateikto klasifikatoriaus rezultatas.

3. Išsiaiškinti kokią nelygybių sistemą reikia spręsti, norint teisingai parinkti svorių ir slenksčio reikšmes, kai aktyvacijos funkcija yra slenkstinė. Išspręsti šią sistemą grafiniu būdu.

x_1	x_2	**Norima reikšmė t (klasė)**
–0,2	0,5	0
0,2	–0,5	0
0,8	–0,8	1
0,8	0,8	1

1 pav. Klasifikatoriaus įėjimo ir išėjimo reikšmės

1. Programos kodas

Pilnas kodas Python kalba:

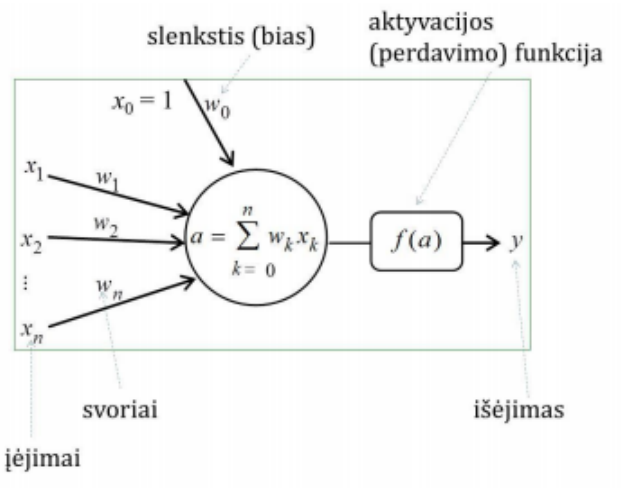
```
import numpy as np


class Neuronas:
    def __init__(self, data, weights, data_outputs):
        self.data = data
        self.data_outputs = data_outputs
        self.weights = weights

# slenkstinė funkcija
    def slenkstine(self, a):
        return 1 if a > 0 else 0

# sigmoidinė funkcija
    def sigmoidine(self, a):
        return 1 / (1 + np.exp(-a))

# įėjimo reikšmių ir svorių sandaugų suma
    def suma(self, data):
        a = np.dot(data, self.weights)
        return a

# mokymo paklaida ir naujų svorių gavimas
    def paklaida(self, y, i, l_rate=1):
        error = self.data_outputs[i] - y
        if y != self.data_outputs[i]:
            for j in range(len(self.weights)):
                self.weights[j] = self.weights[j] + \
                    (l_rate * error * self.data[i][j])

# neurono apmokymo funkcija
# funkcija = 0, jei pasirinkta slenkstinė aktyvacijos funkcija
# funkcija = 1, jei sigmoidinė
    def train(self, funkcija, iterations=1000, l_rate=1):
        for _ in range(iterations):
            for i in range(len(self.data)):
                a = self.suma(self.data[i])
                if funkcija == 1:
                    y = self.sigmoidine(a)
                else:
                    y = self.slenkstine(a)

                # apskaičiuojama paklaida
                self.paklaida(y, i)

```

```
46
47 # skaičiuojama išėjimo reikšmė
48     def calculateOutput(self, data, funkcija):
49         a = self.suma(data)
50         if funkcija == 0:
51             return self.slenkstine(a)
52         else:
53             # Kadangi sigmoidinės funkcijos reikšmės yra intervale (0, 1),
54             # nustatoma, kad jei reikšmė mažesnė nei 0,09 - grąžinamas 0,
55             # kitu atveju grąžinamas 1
56             if self.sigmoidine(a) > 0.99:
57                 return 1
58             else:
59                 return 0
60
61
62 if __name__ == '__main__':
63     # nulinis įėjimas, x1, x2
64     data = [[1.0, -0.2, 0.5],
65             [1.0, 0.2, -0.5],
66             [1.0, 0.8, -0.8],
67             [1.0, 0.8, 0.8]]
68
69     data_outputs = np.array([[0, 0, 1, 1]]).T
70
71     weights = [0, 0, 0]
72
73     neuronas = Neuronas(data, weights, data_outputs)
74     neuronas.train(0)
75
76     print("Slenkstine[0] ar sigmoidine[1] aktyvacijos funkcija?")
77     user_input = int(input("Aktyvacijos funkcija: "))
78
79     print(neuronas.calculateOutput(data[3], user_input))
```

1.1. Dirbtinis neuronas

Dirbtinio neurono modelyje (2 pav.) pažymėtos įėjimo reikšmės, slenkstis, pradiniai svoriai, bei norimos išėjimo reikšmės įvedamos programos pagrindinėje dalyje. Pradiniai svoriai pasirinkti nuliniai, o nulinis įėjimas– 1. Vartotojas įveda norimą aktyvacijos funkciją: 0 – slenkstinei funkcijai, 1 – sigmoidinei.

Išėjimo reikšmės paskaičiuojamos ”calculateOutput” dalyje: suskaičiuojama a = įėjimo reikšmių ir svorių sandaugų suma, o tuomet reikšmė a įstatoma į pasirinktą aktyvacijos funkciją.



2 pav. Dirbtinio neurono modelis

Kodas reikalingas dirbtinio neurono modelio sukūrimui:

```
import numpy as np


class Neuronas:
    def __init__(self, data, weights, data_outputs):
        self.data = data
        self.data_outputs = data_outputs
        self.weights = weights

# slenkstinė funkcija
    def slenkstine(self, a):
        return 1 if a > 0 else 0

# sigmoidinė funkcija
    def sigmoidine(self, a):
        return 1 / (1 + np.exp(-a))

# įėjimo reikšmių ir svorių sandaugų suma
    def suma(self, data):
        a = np.dot(data, self.weights)
        return a
# skaičiuojama išėjimo reikšmė
    def calculateOutput(self, data, funkcija):
        a = self.suma(data)
        if funkcija == 0:
            return self.slenkstine(a)
        else:
            return self.sigmoidine(a)

if __name__ == '__main__':
    # nulinis įėjimas, x1, x2
    data = [[1.0, -0.2, 0.5],
```

```
            [1.0, 0.2, -0.5],
            [1.0, 0.8, -0.8],
            [1.0, 0.8, 0.8]]

    data_outputs = np.array([[0, 0, 1, 1]]).T

    weights = [0, 0, 0]

    neuronas = Neuronas(data, weights, data_outputs)

    print("Slenkstine[0] ar sigmoidine[1] aktyvacijos funkcija?")
    user_input = int(input("Aktyvacijos funkcija: "))

    print(neuronas.calculateOutput(data[3], user_input))
```

1.2. Perceptrono mokymas

Tinkamų svorių radimui (perceptrono mokymui) nustatytas mokymo greitis (angl. *learning rate*) lygus 1, o iteracijų skaičius lygus 1000.

Mokymo procese tinklo išėjimo reikšmė y_i, gauta į įėjimą pateikus vektorių X_i, būtų kiek galima artimesnė norimai reikšmei, todėl pirmiausia yra skaičiuojama vektoriaus išėjimo reikšmė, ji įdedama į nustatytą aktyvacijos funkciją, o tuomet skaičiuojama veikimo paklaida ir nustatomos naujos svorių reikšmės.

Naujasis svoris apskaičiuojamas prie senojo pridedant mokymo greičio, paklaidos, bei įėjimo reikšmės sandaugą.

Kodo dalis dirbtinio neurono apmokymui:

```
# mokymo paklaida ir naujų svorių gavimas
    def paklaida(self, y, i, l_rate=1):
        error = self.data_outputs[i] - y
        if y != self.data_outputs[i]:
            for j in range(len(self.weights)):
                self.weights[j] = self.weights[j] + \
                    (l_rate * error * self.data[i][j])

# neurono apmokymo funkcija
# funkcija = 0, jei pasirinkta slenkstinė aktyvacijos funkcija
# funkcija = 1, jei sigmoidinė
    def train(self, funkcija, iterations=1000, l_rate=1):
        for _ in range(iterations):
            for i in range(len(self.data)):
                a = self.suma(self.data[i])
                if funkcija == 1:
                    y = self.sigmoidine(a)
                else:
                    y = self.slenkstine(a)

```

```
                # apskaičiuojama paklaida
                self.paklaida(y, i)
```

Kadangi sigmoidinės funkcijos reikšmės yra intervale (0, 1), taigi išėjimo reikšmės skaičiavimo funkcijoje nustatomos sąlygos, kada bus klasė = 0, kada klasė = 1. Pasirinkta, jog klasė = 0 bus pritaikyta, jei išėjimo reikšmė bus itin maža, šiuo atveju pasirinkta riba yra 0.99, taigi mažesnės reikšmės, nei 0.99 grąžins – 0, didesnės – 1.

```
# skaičiuojama išėjimo reikšmė
    def calculateOutput(self, data, funkcija):
        a = self.suma(data)
        if funkcija == 0:
            return self.slenkstine(a)
        else:
            if self.sigmoidine(a) > 0.99:
                return 1
            else:
                return 0
```

2. Svorių ir slenksčio reikšmės

Galutinės slenksčio ir svorių reikšmės, gautos naudojant neurono apmokymą, su kuriomis gaunami norimi rezultatai:

1. Taikant slenkstinę funkciją:

   w_0 = -1, w_1 = 2, w_2 = 0.2

   w_0 - slenkstis

2. Taikant sigmoidinę funkciją (reikšmės suapvalintos 0,01 tikslumu):

   w_0 = -7.75, w_1 = 17.54, w_2 = 1.29

Galima teigti, kad $w_0 < 0$, todėl sudaroma lygčių sistema, kurią išsprendus gaunamos svorių apibrėžimo sritys:

Sprendžiama lygčių sistema:

$$\begin{cases} -0.2w_1 + 0.5w_2 + w_0 < 0 \\ 0.2w_1 - 0.5w_2 + w_0 < 0 \\ 0,8w_1 - 0.8w_2 + w_0 \geq 0 \\ 0,8w_1 + 0.8w_2 + w_0 \geq 0 \\ w_0 < 0 \end{cases}$$

Gautos 4 sprendinių aibės:

$$w_1 > 0, w_2 = \frac{2w_1}{5}, -\frac{12w_1}{25} \leq w_0 < 0$$

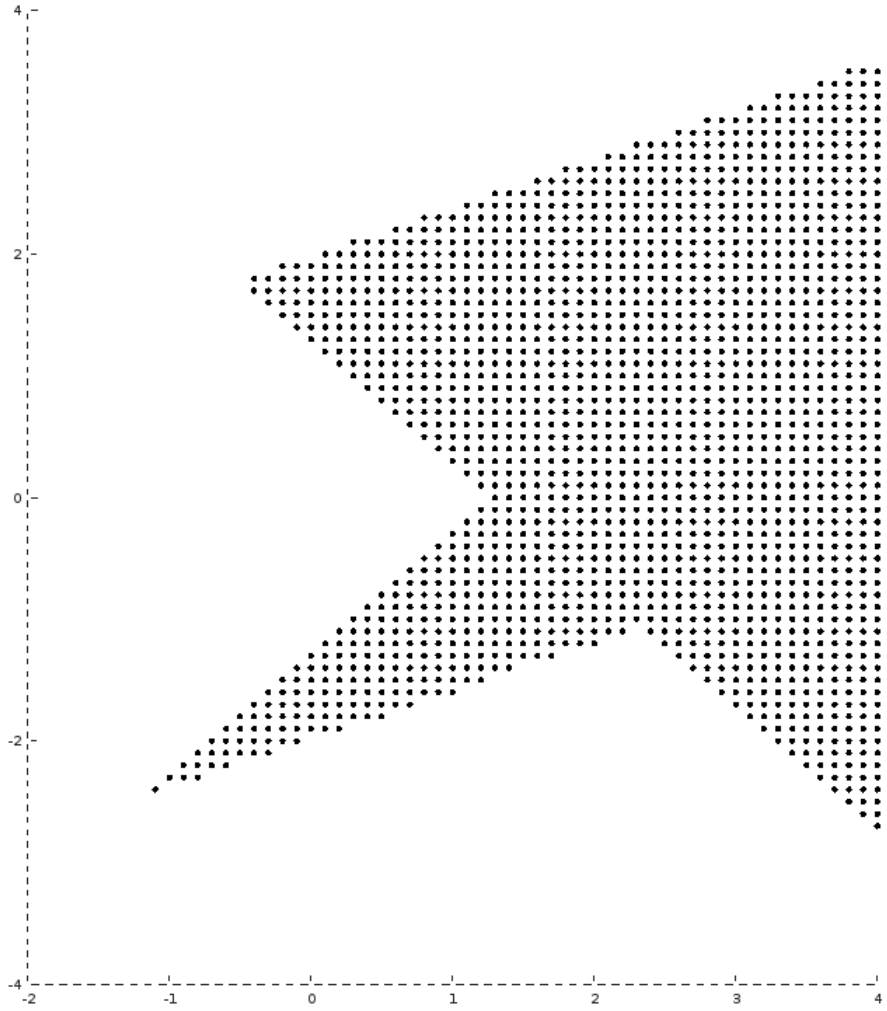
$$w_1 > 0, 0 < w_2 < \frac{2w_1}{5}, -\frac{4}{5}(w_1 - w_2) \leq w_0 < \frac{1}{10}(5w_2 - 2w_1)$$

$$w_1 > 0, -\frac{6w_1}{13} < w_2 \leq 0, -\frac{4}{5}(w_1 - w_2) \leq w_0 < \frac{1}{10}(5w_2 - 2w_1)$$

$$w_1 > 0, \frac{2w_1}{5} < w_2 \leq \frac{10w_1}{13}, -\frac{4}{5}(w_1 - w_2) \leq w_0 < \frac{1}{10}(2w_1 - 5w_2)$$

Taikant slenkstinę funkciją, buvo naudota antroji sprendinių aibė, o taikant sigmoidinę - pirmoji .

Lygčių sistemos grafinis sprendinys: (3 pav).



3 pav. Grafinis lygčių sistemos sprendinys