



VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Skaitmeninis intelektas ir sprendimų priėmimas

I užduotis: Dirbtinis neuronas

Tomas Giedraitis
VU MIF Informatika
3 kursas 3 grupė

Vilnius 2021

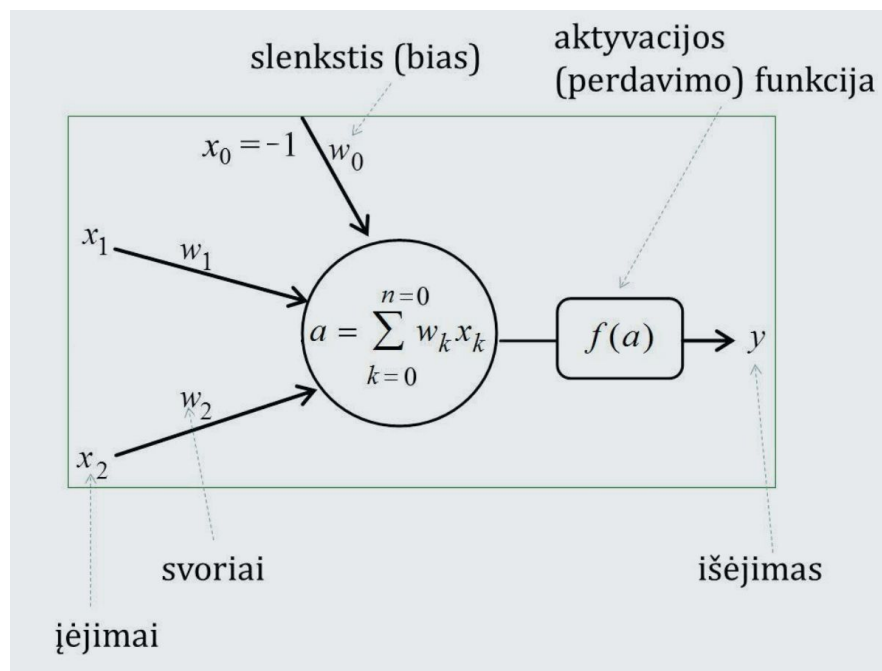
1. Darbo tikslas

Dirbtinio neurono modelio analizė. Išanalizuoti jo veikimo principus, randant neurono svorių ir neurono slenksčio reikšmes (realizuojant slenkstinę ir sigmoidinę aktyvacijos funkcijas), kad būtų patenkintas duoto klasifikatoriaus rezultatas:

1 lentelė. Pateikto klasifikatoriaus rezultatas.

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

Dirbtinio neurono modelis (su dviem svoriais w_1, w_2 , ir slenksčiu w_0) pavaizduotas paveikslėlyje apačioje. Slenksčio įėjimo reikšmę x_0 fiksuojame, priskiriama $x_0 = -1$.



1 pav. Dirbtinio neurono modelis

2. Darbo eiga

2.1. Dirbtinio neurono programa

Parašyta paprasta programa Python kalba (žr. A.1 priedą), kurioje keisdami svorių (w_1, w_2) ir slenksčio (w_0) reikšmes, interpretuojant slenkstį kaip vieną iš svorių, nustatomos atitinkamos reikšmės, su kuriomis patenkinamas pateikto klasifikatoriaus rezultatas. Pradinės svorių reikšmės yra lygios 1, ir toliau jos keičiamos perrinkimo būdu kol randami tinkami svoriai. Kai jie randami, programa nutraukiama, toliau nebeieškome dar labiau tinkančių svorių. Programa veikia su slenkstine ir sigmoidine aktyvacijos funkcijomis.

2.2. Nelygybių sistema

Pagal užduoties reikalavimus, su slenkstine funkcija galima sudaryti nelygybių sistemą iš keturių nelygybių (1)–(4), kurią išsprendę gausime svorių apibrėžimo sritis:

$$-0,2w_1 + 0,5w_2 - w_0 \leq 0 \quad (1)$$

$$0,2w_1 - 0,5w_2 - w_0 \leq 0 \quad (2)$$

$$0,8w_1 - 0,8w_2 - w_0 > 0 \quad (3)$$

$$0,8w_1 + 0,8w_2 - w_0 > 0 \quad (4)$$

Ši sistema buvo spręsta grafiškai, pasinaudojant Desmos¹ ir GNU Octave programomis.

3. Rezultatai

3.1. Dirbtinio neurono programa

Rezultatas naudojant slenkstinę aktyvacijos funkciją:

2 lentelė. Klasifikatoriaus rezultatas su slenkstine funkcija

w_1	w_2	w_0	Funkcijos reikšmė y	Norima reikšmė t (klasė)
2,4	1,0	1,0	0	0
			0	0
			1	1
			1	1

¹ <https://www.desmos.com/>

Slenkstinės funkcijos reikšmių sritis yra $\{0,1\}$, taigi jos reikšmės sutampa su norimomis klasės reikšmėmis t .

Rezultatas naudojant sigmoidinę aktyvacijos funkciją:









3 lentelė. Klasifikatoriaus rezultatas su sigmoidine funkcija

$w1$	$w2$	$w0$	Funkcijos reikšmė y	Interpretuojama klasės reikšmė pagal f-jos reikšmę	Norima reikšmė t (klasė)
4,4	1,0	1,6	0,12131884	0	0
			0,22793645	0	0
			0,75398872	1	1
			0,93819653	1	1

Sigmoidinės funkcijos reikšmių sritis – realiųjų skaičių aibė intervale $(0;1)$. Interpretuojant šią reikšmę, buvo pasirinkta, kad reikšmės, mažesnės už 0,25, atitiks $t = 0$ klasės reikšmę, o reikšmės, didesnės už 0,75: $t = 1$. Laikoma, kad visos kitos reikšmės neatitinka nei vienos iš klasių.

3.1. Nelygybių sistema

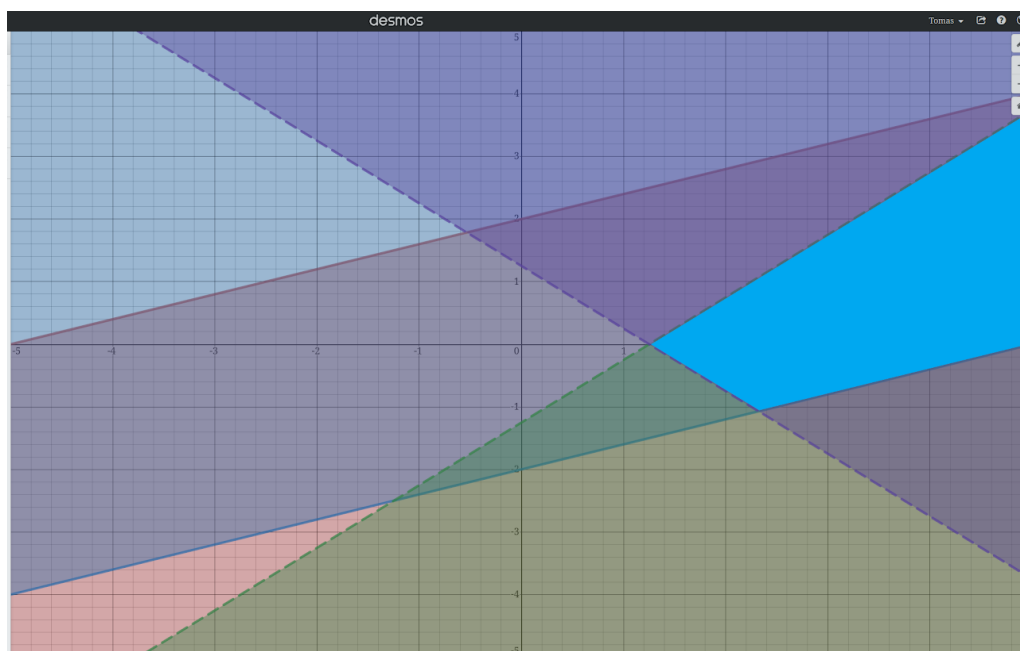
Spręsdami šią sistemą grafiškai, su programa Desmos, įvedame nelygybes, ir kiekvienos jų sprendiniams priskiriama atskira spalva kairėje:

1		$-1 - 0.2x + 0.5y \leq 0$	
2		$-1 + 0.2x - 0.5y \leq 0$	
3		$-1 + 0.8x - 0.8y > 0$	
4		$-1 + 0.8x + 0.8y > 0$	

2 pav. Įvestos nelygybės Desmos programoje

Čia x atitinka svorį w_1 , y – svorį w_2 , o w_0 prilyginamas vienetui, siekiant suprojektuoti sprendinių grafiką plokštumoje.

Gauname tokį brėžinį:



3 pav. Nelygybės sprendinių brėžinys

Jame atitinkamos spalvos žymi atskirų nelygybių sprendinius, o ryškiai mėlyna spalva pažymėta sritis – šios nelygybių sistemos sprendinius – tinkamas svorių w_1 ir w_2 reikšmės, esant fiksuotam w_0 svoriui.

Ši nelygybių sistema buvo papildomai išspręsta ir su GNU Octave programa (žr. programos kodą A.2 priede).

4. Išvados

Parašius programinį kodą, pavyko susipažinti iš arti su dirbtinio neurono veikimo principais. Taip pat, susipažinta su skirtingais jo mokymo parametrais, šiuo atveju tai buvo tik dviejų skirtingų aktyvacijos funkcijų parinkimas.

Reikšmių parinkimas atsitiktiniu (perrinkimo) būdu nėra našus dirbtinio neurono mokymo metodas, tačiau ir tokiu būdu galima jį sėkmingai apmokyti.

Taip pat, galima traktuoti svorių ir norimų reikšmių santykį kaip nelygybių sistemą, ir, jei ji nėra sudėtinga, išspręsti ją grafiniu, o taip pat net ir analitiniu būdu.

A Priedai

A.1 Python programos kodas

```
#!/usr/bin/env python3

import numpy as np

def threshold_fn(x):
    output = []
    for el in x:
        if el[0] > 0:
            output.append(1)
        else:
            output.append(0)
    return output

def sigmoid_fn(x):
    sig = 1 / (1 + np.exp(-x))    # Define sigmoid function
    sig = np.minimum(sig, 0.9999)  # Set upper bound
    sig = np.maximum(sig, 0.0001)  # Set lower bound
    return sig

training_inputs = np.array([[-0.2, 0.5, -1],
                             [0.2, -0.5, -1],
                             [0.8, -0.8, -1],
                             [0.8, 0.8, -1]])

training_outputs = [0, 0, 1, 1]

synaptic_weights = np.array([[1.0], [1.0], [1.0]])

# USAGE: change one to True, and another to False
threshold = True
sigmoid = False

for i in range(1, 1000001):
    outputs = np.dot(training_inputs, synaptic_weights)

    if threshold:
        outputs = threshold_fn(outputs.tolist())

        if outputs == training_outputs:
            break
    elif sigmoid:
        outputs = sigmoid_fn(outputs)
        out = outputs.flatten().tolist()
```

```

        for j in range(2):
            if out[j] < 0.25:
                out[j] = 0

        for j in range(2,4):
            if out[j] > 0.75:
                out[j] = 1

        if out == training_outputs:
            break
    else:
        print('ERROR: unspecified activation function')
        exit(1)

    # update weights
    if i % 10000 == 0:
        synaptic_weights[0][0] += 0.2
        synaptic_weights[1][0] = 1.0
        synaptic_weights[2][0] = 1.0
    elif i % 100 == 0:
        synaptic_weights[1][0] += 0.2
        synaptic_weights[2][0] = 1.0
    else:
        synaptic_weights[2][0] += 0.2

print('Suitable synaptic weights:')
print(synaptic_weights)

print('Corresponding outputs:')
print(outputs)

```

A.2 Octave programos kodas nelygybių sistemai spręsti

```
#!/usr/bin/env octave

% make a grid
[w1, w2] = meshgrid(- 5:0.1:5, - 5:0.1:5);

% fix the weight of the bias
w0 = 1;

% system of inequalities
ineq1 = - w0 - 0.2 * w1 + 0.5 * w2 <= 0;
ineq2 = - w0 + 0.2 * w1 - 0.5 * w2 <= 0;
ineq3 = - w0 + 0.8 * w1 - 0.8 * w2 > 0;
ineq4 = - w0 + 0.8 * w1 + 0.8 * w2 > 0;

% color palette (black color (=0 0 0) at the last
% position indicates where the solutions reside)
mymap = [1 1 1; 1 1 1; 1 1 1; 0 0 0];
colormap(mymap);
colors = zeros(size(w0)) + ineq1 + ineq2 + ineq3 + ineq4;

% draw the system of inequalities solution graph
plt = scatter(w1(:), w2(:), 3, colors(:), 'filled');

waitfor(plt)
disp('Exiting...')
```