VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

Dirbtinis neuronas

Praktinio darbo ataskaita

Atliko: Justinas Bliujus
VU el. p.: justinas.bliujus@mif.stud.vu.lt

Vertino:

Vilnius
2025

1. SKYRIUS

Šiame darbe aprašomas dirbtinio neurono kūrimas ir duotų užduočių vykdymo ataskaita.

1.1. Tikslas

Išanalizuoti dirbtinio neurono veikimo principus vykdant duotas užduotis.

1.2. Taškų generavimas

Sukuriama dvidešimt taškų. Dešimt, kai x koordinatės reikšmė iki 4. Dešimt, kai x reikšmė nuo 6. Y reikšmės atsitiktinės.

1 lentelė. Taškai

Klasė 0		**Klasė 1**	
x	***y***	***x***	***y***
1	2	6	7
2	3	7	6
1,5	2,5	6,5	6,5
3	3	8	7
2,5	4	7,5	8
3,5	2,5	8,5	6,5
4	4,5	9	9,5
2	2	7	7
3	2	8	6
4	3	9	8

1.3. Neurono aprašymas

Neuronas šiuo atveju yra funkcija, priimanti taškų x_1 ir x_2 koordinates, svorius w_1,w_2 bei funkcijos tipą step arba sigmoid. Iš pradžių neuronas apskaičiuoja a reikšmę pagal formulę

$$a = w_1 \times x_1 + w_2 \times x_2 + b$$

Kai funkcijos tipas step, neuronas grąžina 1 jeigu $a \geq 0$, 0 jeigu $a < 0$.

Kai funkcijos tipas sigmoid, neuronas grąžina suapvalintą sigmoidinės funkcijos $y = \frac{1}{1+e^{-a}}$ reikšmę (0 arba 1).

1.4. Svorių ir poslinkio ieškojimas

Pirmiausia sugeneruojamos atsitiktinės svorių ir poslinkio reikšmės, šiuo atveju iš intervalo (-5;5). Tuomet su visais nulinės klasės taškai tikrinama, ar neuronas grąžina reikšmę 0. Tikrinama ar su visais pirmosios klasės taškais neuronas grąžina reikšmę 1. Jeigu taip, yra rasti tinkami svoriai ir poslinkis. Tai kartojama kol randami trys komplektai. Jeigu reikšmės netinka, generuojamos naujos atsitiktinės svorių ir poslinkio reikšmės ir viskas kartojama.

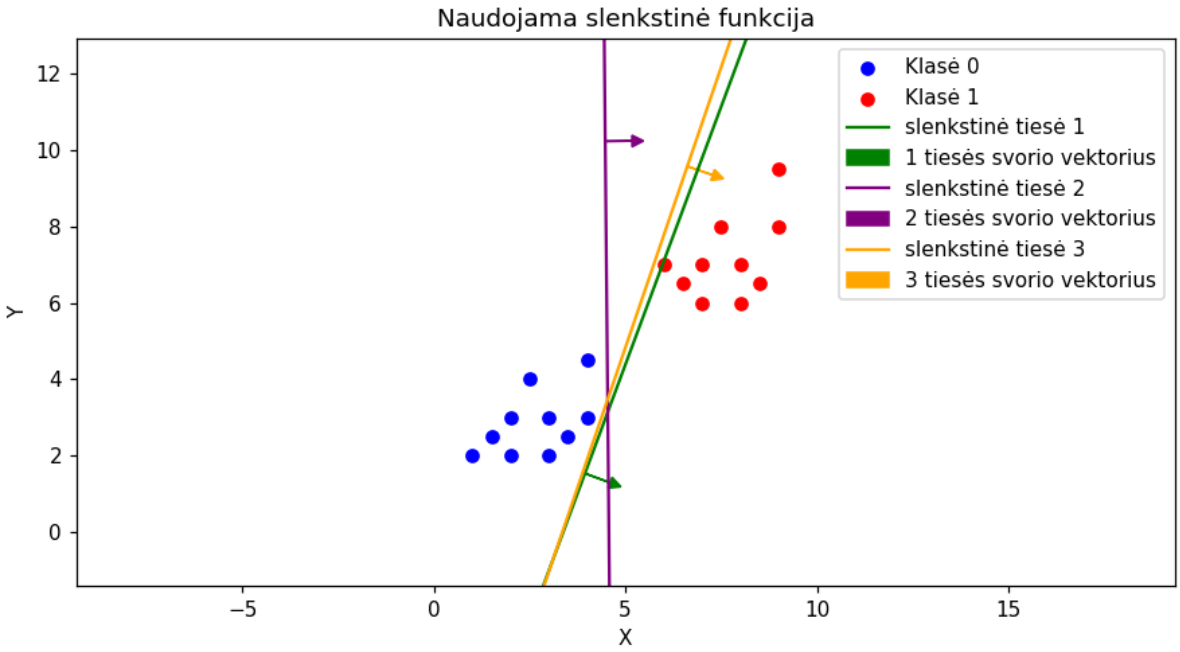
1.5. Tiesių ir vektorių formavimas

Tiesės nubrėžiamos naudojant 100 taškų and x ašies, nuo mažiausios x reikšmės taškų aibėje iki didžiausios *x* reikšmės taškų aibėje. Y reikšmėms surasti naudojama formulė $x_1 = -\frac{w_1 \times x_0 + b}{w_2}$, išreikšta iš $a = w_1 \times x_1 + w_2 \times x_2 + b$, kai a = 0 (nes a yra klasifikavimo riba).

Vektoriams nubrėžti reikia rasti pradžios taškus. X koordinatė parenkama atsitiktinai iš galimos aibės. Y koordinatė apskaičiuojama pagal $x_1 = -\frac{w_1 \times x_0 + b}{w_2}$. Vektoriaus kryptis yra svoriai w_1,w_2.

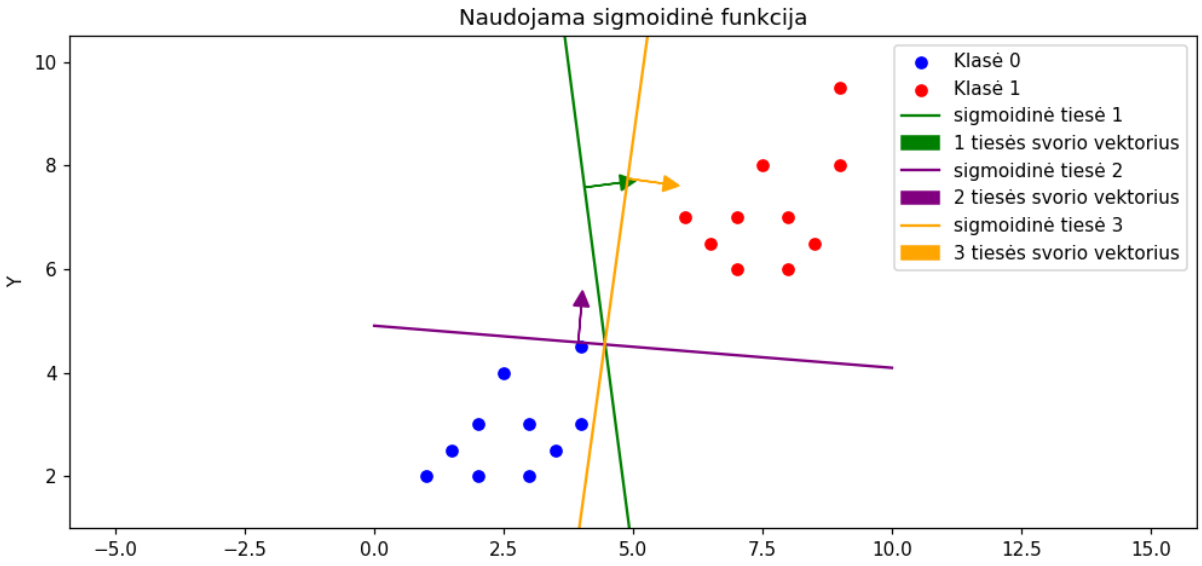
1.6. Rezultatai

Paveikslėlyje *1 pav.* Rezultatai naudojant slenkstinę funkciją matomi sugeneruoti dvejų klasių taškai, mėlyna spalva – kai x reikšmė iki 4, raudoni – kai x reikšmė nuo 6. Matomos trys juos skiriančios tiesės bei statmeni tiesėms svorių vektoriai. Nors x ir y ašys nesutampa, visi taškai yra intervale (0;10) atsižvelgiant į abi ašis.



1 pav. Rezultatai naudojant slenkstinę funkciją

Paveikslėlyje *2 pav.* Rezultatai naudojant sigmoidinę funkciją matomi tie patys sugeneruoti taškai, rastos trys skiriamosios tiesės ir joms statmeni svorių vektoriai.



2 pav. Rezultatai naudojant sigmoidinę funkciją

Paveikslėlyje *3 pav.* Rasti svoriai ir poslinkiai matomi trys komplektai svorių ir poslinkių naudojant slenkstinę funkciją ir trys komplektai rasti naudojant sigmoidinę funkciją. Kadangi reikšmės ieškotos aibėje (-5;5), rezultatai yra atitinkamai iš šio intervalo.

```
Using step function:
Found weights (step activation): w1=0.44, w2=-0.16, b=-1.49
Found weights (step activation): w1=1.03, w2=0.01, b=-4.70
Found weights (step activation): w1=0.57, w2=-0.20, b=-1.91
Using sigmoid function:
Found weights (sigmoid activation): w1=0.88, w2=0.12, b=-4.44
Found weights (sigmoid activation): w1=0.05, w2=0.56, b=-2.75
Found weights (sigmoid activation): w1=1.07, w2=-0.15, b=-4.07
```

3 pav. Rasti svoriai ir poslinkiai

1.7. Išvados

Darbo metu buvo išanalizuotas dirbtinio neurono veikimo principas, suprantant, kad neuronas atlieka matematinį skaičiavimą su įvestimi ir duotais svoriais. Mokymas nebuvo atliktas, todėl pasitelkus atsitiktinį svorių ir poslinkių ieškojimą buvo rastos reikiamos reikšmės. Nubrėžtos tiesės parodo, kad taškai buvo klasifikuojami teisingai. Reikšmingo skirtumo naudojant slenkstinę aktyvacijos funkciją ir naudojant sigmoidinę aktyvacijos funkciją nėra, nes sigmoidinės funkcijos rezultatai yra apvalinami.

1 priedas. Programinis kodas.

```
import numpy as np
import matplotlib.pyplot as plt

def plot_data(found_sets, func):
    class_0 = np.array([(1, 2), (2, 3), (1.5, 2.5), (3, 3), (2.5, 4), #aprašomi taškai
                        (3.5, 2.5), (4, 4.5), (2, 2), (3, 2), (4, 3)])
    class_1 = np.array([(6, 7), (7, 6), (6.5, 6.5), (8, 7), (7.5, 8),
                        (8.5, 6.5), (9, 9.5), (7, 7), (8, 6), (9, 8)])

    # Nupiešiami taškai
    plt.scatter(class_0[:, 0], class_0[:, 1], color='blue', label='Klasė 0')
    plt.scatter(class_1[:, 0], class_1[:, 1], color='red', label='Klasė 1')

    # pagal aprašytus taškus randamos ribos (vaizdavimo patogumui)
    all_points = np.concatenate((class_0, class_1), axis=0)
    x_min, x_max = all_points[:, 0].min() - 1, all_points[:, 0].max() + 1
    y_min, y_max = all_points[:, 1].min() - 1, all_points[:, 1].max() + 1

    colors = ['green', 'purple', 'orange'] #spalvos tiesėms ir jų svorių vektoriams
    for i, (w1, w2, b) in enumerate(found_sets):
        # tiesės piešiamos pasitelkiant 100 taškų ant x ašies tarp ribų
        x_vals = np.linspace(x_min, x_max, 100)
        # y reikšmės randamos pagal formulę ir sugeneruotus x taškus
        y_vals = -(w1 * x_vals + b) / w2

        # nupiešiama tiesė
        plt.plot(x_vals, y_vals, color=colors[i], label=f'{func} tiesė {i+1}')

        # nupiešti svorio vektoriui pasirenkama x reikšmė tarp ribų
        x0 = np.random.uniform(x_min, x_max)
        y0 = -(w1 * x0 + b) / w2 # randama atitinkama y reikšmė

        # patikrinama ar x ir y bus matomos grafike. Jeigu ne, ieškomos kitos
reikšmės.
        if y0 < y_min or y0 > y_max:
            y0 = np.random.uniform(y_min, y_max)
            x0 = (w2 * y0 + b) / (-w1)

        # Svorio vektorius normalizuojamas dėl estetikos
        norm = np.sqrt(w1**2 + w2**2)
        w1_norm, w2_norm = w1 / norm, w2 / norm

        # nupiešiamas vektorius
        plt.arrow(x0, y0, w1_norm, w2_norm, color=colors[i], head_width=0.3,
head_length=0.3, length_includes_head=True,
                  label=f'{i+1} tiesės svorio vektorius')

    plt.title(f'Naudojama {func} funkcija')
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.legend()
    plt.axis('equal')
    plt.xlim(x_min, x_max)
    plt.ylim(y_min, y_max)
    plt.show()

# aktyvacijos funkcija, priema funkcijos tipą ir a reikšmę, išveda 1 arba 0
def activation(value, func='step'):
    if func == 'step':
        return 1 if value >= 0 else 0
    elif func == 'sigmoid':
        return round(1 / (1 + np.exp(-value)))
# neuronas, priimantis reikšmes reikalingas a apskaičiuoti ir funkcijos tipą
aktyvacijai.
def neuron(x, y, w1, w2, b, func='step'):
```

```
    a = x * w1 + y * w2 + b
    return activation(a, func)
# funkcija apskaičiuoti svoriams
def find_weights(func='step'):
    class_0 = [(1, 2), (2, 3), (1.5, 2.5), (3, 3), (2.5, 4),# taškai
               (3.5, 2.5), (4, 4.5), (2, 2), (3, 2), (4, 3)]
    class_1 = [(6, 7), (7, 6), (6.5, 6.5), (8, 7), (7.5, 8),
               (8.5, 6.5), (9, 9.5), (7, 7), (8, 6), (9, 8)]

    found_sets = [] # rasti skaičių komplektai
    while len(found_sets) < 3:# ieškoma kol randami trys komplektai
        w1, w2, b = np.random.uniform(-5, 5, 3)# reikšmės ieškomos šiame intervale

        correct = True
        # Jeigu svoriai ir poslinkis netinka visiems taškams klasifikuoti, ieškomos
kitos reikšmės.
        for x, y in class_0:
            if neuron(x, y, w1, w2, b, func) != 0:
                correct = False
                break
        for x, y in class_1:
            if neuron(x, y, w1, w2, b, func) != 1:
                correct = False
                break

        if correct:# jeigu reikšmės tinka, informacija išvedama
            found_sets.append((w1, w2, b))
            print(f'Found weights ({func} activation): w1={w1:.2f}, w2={w2:.2f},
b={b:.2f}')

    return found_sets

if __name__ == "__main__":
    print("Using step function:")
    step_weights = find_weights('step')
    plot_data(step_weights, 'slenkstinė')
    print("Using sigmoid function:")
    sigmoid_weights = find_weights('sigmoid')
    plot_data(sigmoid_weights, 'sigmoidinė')
```