VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

1 užduotis (Dirbtinis neuronas)

Praktinio darbo ataskaita

Atliko: Monika Mirbakaitė
VU el. p.: monika.mirbakaite@mif.stud.vu.lt

Vertino: dr. Viktor Medvedev

Vilnius
2025

TURINYS

1. UŽDUOTIES TIKSLAS

Išanalizuoti dirbtinio neurono modelio veikimo principus.

2. DARBO ATLIKIMAS

2.1. Sugeneruoti duomenys

Generavimo pražioje generuojama 10 taškų dvimatėje erdvėje aplink (0;0). Vėliau šie taškai perstumiami pagal pasirinktą intervalą. Šiuo atveju pirmoje klasėje (1 lentelė. Sugeneruoti duomenys pirmoje klasėje.) tai yra (4;5), o antroje (2 lentelė. Sugeneruoti duomenys antroje klasėje.) – (8;10). Gauti rezultatai pateikiami žemiau esančiose lentelėse.

1 lentelė. Sugeneruoti duomenys pirmoje klasėje.

Taško Nr.	x	y
1	3,3924523	4,87386359
2	3,31539364	5,92871475
3	2,15559897	4,53299758
4	6,29249034	5,48881005
5	4,71026699	6,05553444
6	4,0540731	5,25795342
7	4,58828165	5,88524424
8	2,98299298	4,86630697
9	3,5618145	5,49344349
10	3,80099088	3,72501639

2 lentelė. Sugeneruoti duomenys antroje klasėje.

Taško Nr.	x	y
1	8,29349415	10,10895031
2	8,03172679	11,27263986
3	9,0714479	10,41581801
4	9,55067923	9,68862108
5	6,62076009	11,37140879
6	8,02771165	9,67960042
7	7,15382959	9,56657108
8	6,6629655	10,20917217
9	6,5756787	9,44652315
10	8,07479864	9,49438017

2.2. Programos kodas su komentarais

Programos kodas su komentarais bei paaiškinimais pateikiamas 4 skyriuje.

2.3. Svorių ir poslinkio rinkinių paieška

Sugeneruotų taškų informacija išsaugoma bei sujungiama (pirmosios klasės informacija sujungiama su antrosios klasės informacija). Atliekamas pasirinkimas (pagal kokią aktyvacijos funkciją generuojami rinkiniai). Sugeneruojami svoriai bei poslinkis atsitiktinai intervale (-10;10). Sukuriamas dirbtinis neuronas ir atliekama prognozė. Prognozė bando nuspėti taškų klasę. Tikrinama, ar atliktas spėjimas sutampa su tikrais duomenimis. Išsaugomi tik tie rinkiniai (apsiribojama trimis), kurių visi taškai buvo suskirstyti sėkmingai.

2.4. Gauti svorių ir poslinkio reikšmių rinkiniai

Svorių ir poslinkio reikšmių rinkinių algoritmas aprašytas 2.3 poskyryje. Gauti rezultatai pateikiami žemiau esančiose lentelėse (3 lentelė. Gauti svorių ir poslinkio reikšmių rinkiniai naudojant sigmoidinę aktyvacijos funkciją., 4 lentelė. Gauti svorių ir poslinkio reikšmių rinkiniai naudojant slenkstinę aktyvacijos funkciją.).

2.4.1. Naudojant sigmoidinę aktyvacijos funkciją

3 lentelė. Gauti svorių ir poslinkio reikšmių rinkiniai naudojant sigmoidinę aktyvacijos funkciją.

Rinkinio Nr.	Svoris w_1	Svoris w_2	Poslinkis b
1	-1,07	0,28	-9,59
2	0	1,26	-9,49
3	0,44	0,1	-3,58

2.4.1. Naudojant slenkstinę aktyvacijos funkciją

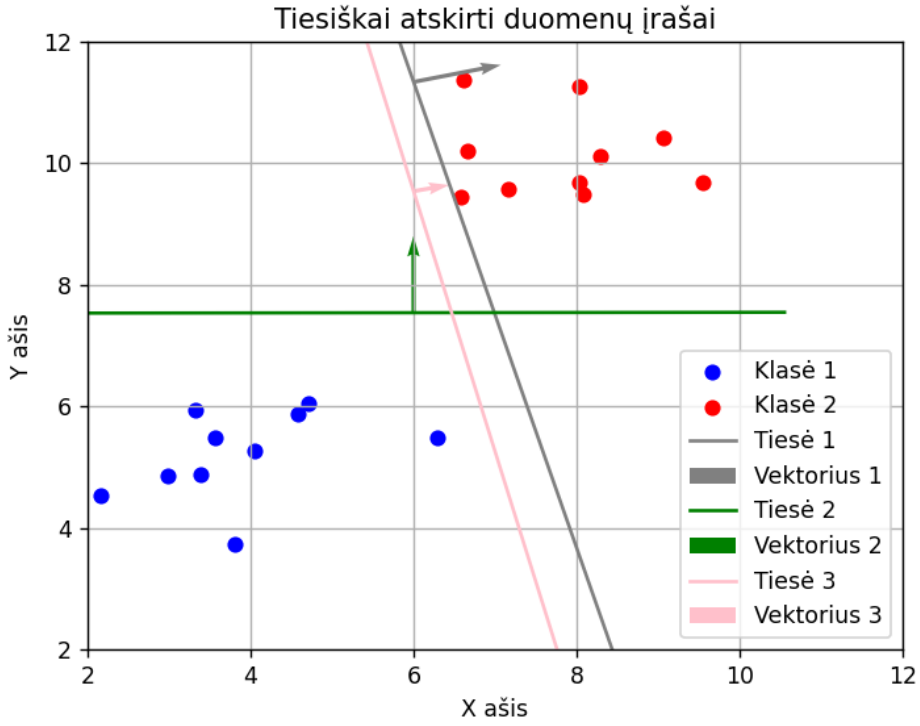
4 lentelė. Gauti svorių ir poslinkio reikšmių rinkiniai naudojant slenkstinę aktyvacijos funkciją.

Rinkinio Nr.	Svoris w_1	Svoris w_2	Poslinkis b
1	1,07	0,28	-9,59
2	0	1,26	-9,49
3	0,44	0,1	-3,58

2.5. Duomenų taškų, klases skiriančių tiesių, neurono svorius atitinkanių vektorių vaizdavimas

1 pav. Duomenų taškai, klases skiriančios tiesės, neurono svorius atitinkantys vektoriai. Taškai yra tiesiškai atskirti. Mėlyna spalva vaizduojami pirmosios klasės taškai, o raudona – antrosios. Pavaiduotos tiesės atskiria klases. Kadangi buvo gauti trys svorių ir poslinkio rinkiniai, tad vaizduojamos trys tiesės. Pirmoji tiesė vaizduojama pilka spalva, antroji – žalia, o trečioji – rožine. Pavaizduoti vektoriai žymi neurono svorius. Kuo ilgesnis vektorius, tuo svoris didesnis. Kryptis rodo, kokia kryptimi atskiriamos klasės. Svorių vektoriai yra statmeni (ortogonalūs)

skiriamajam paviršiui. Pilka spalva žymimas pirmosios tiesės vektorius, žalia – antrosios, o rožine – trečiosios.



1 pav. Duomenų taškai, klases skiriančios tiesės, neurono svorius atitinkantys vektoriai.

3. REZULTATAI

3.1. Dirbtinio intelekto įrankių indėlis

Dirbtinio intelekto įrankis DeepSeek padėjo duomenų vaizdavimui, taškų generavimui.

3.2. Išvados

Darbo metu sugeneruoti duomenų taškai buvo tiesiškai atskiriami, o perceptronas sėkmingai rado ribą tarp dviejų klasių. Taip pat buvo ieškoma tinkamų svorių bei poslinkio atsitiktiniu būdu kol buvo sėkmingai surasti 3 rinkiniai. Be to, buvo rasta keletas skirtingų svorių ir poslinkio rinkinių, kurie visi sėkmingai atskyrė duomenų klases.

4. PRIEDAI

Šiame skyriuje pateikiamas programos kodas su komentarais bei paaiškinimais.

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

def sugeneruoti_duomenis():
    """ Duomenų generavimas. """
    np.random.seed(40)                                    # Kad rezultatai būtų
atkartojami
    klase_1 = np.random.randn(10, 2) + np.array([4, 5])  # Klasė 1: Taškai aplink
tašką (7, 14)
    klase_2 = np.random.randn(10, 2) + np.array([8, 10]) # Klasė 2: Taškai aplink
tašką (15, 20)
    return klase_1, klase_2

def atvaizduoti_taskus(klase_1, klase_2):
    """ Atvaizduojami tiesiškai atskirti taškai. """
    plt.scatter(klase_1[:, 0], klase_1[:, 1], color='blue', label='Klasė 1')
    plt.scatter(klase_2[:, 0], klase_2[:, 1], color='red',  label='Klasė 2')
    plt.xlabel('X ašis')
    plt.ylabel('Y ašis')
    plt.legend()
    plt.title('Tiesiškai atskiriami duomenų įrašai')
    plt.grid(True)
    plt.show()

def zodynas(klase_1, klase_2):
    """ Išsaugomi duomenys pasinaudojant žodynu. """
    zodynas = {
        "klase_1": klase_1,
        "klase_2": klase_2
    }
    return zodynas

def slenkstine_funkcija(a):
    """ Apibrėžiama slenkstinė aktyvacijos funkcija. Atitinkamai pagal sumą grąžinamas
0 arba 1. """
    return 1 if a >= 0 else 0
def sigmoidine_funkcija(a):
    """ Apibrėžiama sigmoidinė aktyvacijos funkcija. Pritaikoma sigmoidinės funkcijos
formulė. """
    sigmoidine_rezultatas = 1 / (1 + np.exp(-a))
    return 1 if sigmoidine_rezultatas >= 0.5 else 0

class Neuronas:
    """. Dirbtinio neurono klasė. """
    def __init__(self, w1, w2, b):
        self.w1 = w1 # Pirmos klasės įėjimo (x1) svoris
        self.w2 = w2 # Antros klasės įėjimo (x2) svoris
        self.b = b   # Poslinkis

    def a_apskaiciavimas(self, x1, x2):
        """ Skaičiuojama įėjimo reikšmių (x1, x2) ir svorių (w1, w2) sandaugų suma,
prie kurios dar pridedamas poslinkis (b). """
        return x1 * self.w1 + x2 * self.w2 + self.b

    def spejimo_funkcija(self, x1, x2, aktyvacijos_funkcija):
        """ Atliekama prognozė: neuronas klasifikuoja įėjimo duomenis (x1 ir x2) į
vieną iš dviejų klasių (0 arba 1). """
        a = self.a_apskaiciavimas(x1, x2)
        # Atliekama prognozė pasirinktu metodu (aktyvacijos funkcija)
        if aktyvacijos_funkcija ==   "slenkstine":
            return slenkstine_funkcija(a)
        elif aktyvacijos_funkcija == "sigmoidine":
```

```
            return sigmoidine_funkcija(a)

def svoriu_tinkamumo_patikra(w1, w2, b, duomenys, klases, aktyvacijos_funkcija):
    """ Tikrinama, ar svoriai bei poslinkis yra tinkami. Palyginami tikri duomenys su
prognoze. """
    neuronas = Neuronas(w1, w2, b)
    for i, (x1, x2) in enumerate(duomenys): # Naudojami 1-ame punkte sugeneruoti
duomenys
        spejamas_y = neuronas.spejimo_funkcija(x1, x2, aktyvacijos_funkcija)
        if spejamas_y != klases[i]:
            return False                    # Jei bent vienas taškas neteisingai
klasifikuotas
    return True                             # Jei visi taškai teisingai klasifikuoti

def tieses_braizymas(w1, w2, b, spalva, label):
    """ Nubraižoma tiesė naudojant svorius ir poslinkį. """
    x_reiksmes = np.array([min(duomenys[:, 0]) - 1, max(duomenys[:, 0]) + 1])
    y_reiksmes = (-w1 * x_reiksmes - b) / w2
    plt.plot(x_reiksmes, y_reiksmes, color=spalva, label=label)


def pasirinkti_aktyvacijos_funkcija():
    """ Realizuojama galimybė pasirinkti aktyvacijos funkciją. """
    pasirinkimas = input("Pasirinkite norimą aktyvacijos funkciją. A - sigmoidinė, B -
slenkstinė: ")
    if pasirinkimas.upper() == "A":
        print("Naudojama sigmoidinė aktyvacijos funkcija.")
        return "sigmoidine"
    elif pasirinkimas.upper() == "B":
        print("Naudojama slenkstinė aktyvacijos funkcija.")
        return "slenkstine"
    else:
        print("Nebuvo pasirinktas nei vienas iš variantų. Šiuo atveju naudojama
slenkstinė funkcija.")
        return "slenkstine"

def rasti_tinkamus_svoriu_rinkinius(duomenys, klases, aktyvacijos_funkcija):
    """ Ieškomi tinkami svorių rinkiniai. """
    print(f"\nIeškoma tinkamų svorių ir poslinkio rinkinių naudojant
{aktyvacijos_funkcija} aktyvacijos funkciją... ")
    tinkami_rinkiniai = []
    bandymu_skaicius = 0
    while len(tinkami_rinkiniai) < 3 and bandymu_skaicius < 10000:
        bandymu_skaicius += 1
        w1, w2, b = np.random.uniform(-10, 10, 3)
        if svoriu_tinkamumo_patikra(w1, w2, b, duomenys, klases,
aktyvacijos_funkcija):
            tinkami_rinkiniai.append((w1, w2, b))
            print(f"Rastas tinkamas rinkinys {len(tinkami_rinkiniai)}: w1 = {w1:.2f},
w2 = {w2:.2f}, b = {b:.2f}")
    return tinkami_rinkiniai

def testuoti_rinkinius(tinkami_rinkiniai, duomenys, klases, aktyvacijos_funkcija):
    """ Testuojami rasti rinkiniai. """
    print(f"\nTestuojami rasti rinkiniai naudojant {aktyvacijos_funkcija} aktyvacijos
funkciją: ")
    for i, (w1, w2, b) in enumerate(tinkami_rinkiniai):
        neuronas = Neuronas(w1, w2, b)
        print(f"\nRinkinys {i + 1}: w1 = {w1:.2f}, w2 = {w2:.2f}, b = {b:.2f}")
        for j, (x1, x2) in enumerate(duomenys):
            spejamas_y = neuronas.spejimo_funkcija(x1, x2, aktyvacijos_funkcija)
            print(f"Taškas ({x1:.2f}, {x2:.2f}) - Prognozė: {spejamas_y}, Tikroji
klasė: {klases[j]}")

def atvaizduoti_tieses_be_vektoriu(tinkami_rinkiniai, klase_1, klase_2):
    """ Atvaizduojami tiesiškai atskirti taškai su tiesėmis. """
```

```
    plt.scatter(klase_1[:, 0], klase_1[:, 1], color='blue', label='Klasė 1')
    plt.scatter(klase_2[:, 0], klase_2[:, 1], color='red', label='Klasė 2')
    plt.xlabel('X ašis')
    plt.ylabel('Y ašis')
    plt.legend()
    plt.title('Tiesiškai atskiriami duomenys su tiesėmis')
    plt.grid(True)
    plt.xlim(-10, 30)
    plt.ylim(-10, 30)
    spalvos = ['green', 'purple', 'orange']
    for i, (w1, w2, b) in enumerate(tinkami_rinkiniai):
        tieses_braizymas(w1, w2, b, spalvos[i], f"Tiesė {i + 1}")
    plt.legend()
    plt.show()
def atvaizduoti_tieses_su_vektoriais(tinkami_rinkiniai, klase_1, klase_2):
    """ Atvaizduojami tiesiškai atskirti taškai su tiesėmis bei vektoriais. """
    plt.scatter(klase_1[:, 0], klase_1[:, 1], color='blue', label='Klasė 1')
    plt.scatter(klase_2[:, 0], klase_2[:, 1], color='red', label='Klasė 2')
    plt.xlabel('X ašis')
    plt.ylabel('Y ašis')
    plt.legend()
    plt.title('Tiesiškai atskiriami duomenys su tiesėmis ir vektoriais')
    plt.grid(True)
    plt.xlim(-10, 30)
    plt.ylim(-10, 30)
    spalvos = ['green', 'purple', 'orange']
    for i, (w1, w2, b) in enumerate(tinkami_rinkiniai):
        tieses_braizymas(w1, w2, b, spalvos[i], f"Tiesė {i + 1}")
        x0 = 3.5
        y0 = (-w1 * x0 - b) / w2
        plt.quiver(x0, y0, w1, w2, angles='xy', scale_units='xy', scale=1,
color=spalvos[i], width=0.005, label=f"Vektorius {i + 1}")
    plt.legend()
    plt.show()

if __name__ == "__main__":
    """ Pagrindinė funkcija (main). """
    klase_1, klase_2 = sugeneruoti_duomenis()
    atvaizduoti_taskus(klase_1, klase_2)

    duomenys = np.vstack((klase_1, klase_2))
    klases = np.array([0] * len(klase_1) + [1] * len(klase_2))

    aktyvacijos_funkcija = pasirinkti_aktyvacijos_funkcija()

    tinkami_rinkiniai = rasti_tinkamus_svoriu_rinkinius(duomenys, klases,
aktyvacijos_funkcija)

    if tinkami_rinkiniai:
        testuoti_rinkinius(tinkami_rinkiniai, duomenys, klases, aktyvacijos_funkcija)
        if aktyvacijos_funkcija == "slenkstine":
            atvaizduoti_tieses_be_vektoriu(tinkami_rinkiniai, klase_1, klase_2)
            atvaizduoti_tieses_su_vektoriais(tinkami_rinkiniai, klase_1, klase_2)
```