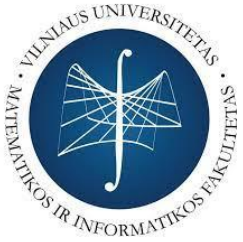


VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

DUOMENŲ MOKSLO BAKALAURO STUDIJOS

Dirbtinio neurono modelis

Artificial Neuron Model

tiriamasis darbas

Atliko: Vainius Gataveckas

VU el.p.: vainius.gataveckas@mif.stud.vu.lt

Vertintojas: Dr. Viktor Medvedev

Vilnius

2021

Darbo tikslas – išanalizuoti dirbtinio neurono modelį ir jo veikimo principus.

Dirbtinio neurono modelio tikslas klasifikuoti duomenis. Tai pasiekiama panaudojus mokomąją aibę tokių svorių ir poslinkių parinkimui, kad visiems stebėjimams svorių $w_i \in W$ ir $x_i \in X$ įėjimų sandaugos bei poslinkio w_0 suma a būtų tokia, kad aktyvacijos funkcija f(a) = y, kur y yra duoto stebėjimo klasė.

Darbo uždaviniai:

1. Suprogramuoti dirbtinio neurono modelį su slenkstine ir sigmoidinę aktyvacijos funkcija f(a).
2. Pasinaudojus duomenų aibe (1 lentelė) su keturiais stebėjimais taikant dvi svorių ir poslinkio parinkimo strategijas – iš nustatyto intervalo generuoti atsitiktines w_i reikšmes, pereiti intervalą su tam tikru žingsniu.
3. Grafiniu būdu išspręsti svorių ir poslinkio radimo lygtį duomenų aibei.
4. Patikrinti ar grafiniu būdu rasti sprendiniai teisingai klasifikuoja duotus duomenis

1 lentelė. Duomenų aibė

Duomenys		**Klasė**
x_1	**x_2**	**t**
-0,3	**0,6**	**0**
0,3	**-0,6**	**0**
1,2	**-1,2**	**1**
1,2	**1,2**	**1**

Darbo tikslų įgyvendinimui bus pasitelkiama programavimo kalba „Python” ir grafinio lygčių sprendimo sistema suteikiama svetainės „WolframAlpha”.

1. Neurono modelio realizacija.

Duomenys perdaromi į sąrašo struktūrą, taip, kad indeksas i, atitiktų vieną duomenų stebėjimą(1 kodo fragmentas). Sukuriami trys sąrašai x1, x2 ir t. x1, x2 yra stebėjimų požymiai, o t klasės reikšmė.

1 kodo fragmentas. Duomenų struktūra.

```
x1 = [-.3,0.3,1.2,1.2]
x2 = [.6,-.6,-1.2,1.2]
t = [0,0,1,1]
```

Neuronas realizuojamas kaip klasinis komponentas(2 kodo fragmentas). Jo iniciavimo metu suteikiamas įėjimų skaičius. Į šį skaičių neįeina poslinkis. Klasės funkcija „grazinti_a“ suskaičiuoja a reikšmę pateikus svorių ir duomenų aibę. Svorių aibė pateikiama sąrašu, kurio primas elementas yra poslinkis, toliau iš eilės svoriai pagal indeksą i atitinkantį požymį i. Slenkstinės aktivacijos funkcija „slenkstine_aktivacija“ priima a reikšmę ir gražina klasės reikšmę. Funkcija „sigmoidinė_aktivacija“ gražina sigmoidinės funkcijos reikšmę.

2 kodo fragmentas. Neurono klasė.

```
class Neuronas:
  def __init__(self, iejimai):
    self.iejimai = iejimai

  def grazinti_a(self, svoriai, duomenys):
    a = 0 + svoriai[0]
    for i in range(self.iejimai):
      a += svoriai[i+1] * duomenys[i]
    return a
  def slenkstine_aktivacija(self,a):
    if(a<0):
      return 0
    elif(a>=0):
      return 1
  def sigmoidine_aktivacija(self,a):
    return 1/(1+math.exp(-a))
```

2. Svorių parinkimas

Svoriams parinkti panaudotos dvi strategijos. Svorių paieška intervalo reikšmes pereinant nustatytu žingsniu (a) ir atsitiktinių svorių parinkimas iš intervalo (b). Pirmiausia ekspertiškai nuspręsta naudoti intervalą [-10; 10].

a) Intervalo perėjimas žingsniu.

Intervalą [-10; 10] pereinama žingsniu vienas, t.y. tiek svoriams tiek poslinkiui išbandomos visos įmanomos reikšmių kombinacijos (-10, -10, -10), (-10,-10,-9) ir t.t.

3 kodo fragmentas. Svorių parinkimas su slenkstine aktyvacijos funkcija.

```
def rasti_svorius(neuronas, duomenu_matrica, intervalas):
  for w0 in intervalas:
    for w1 in intervalas:
      for w2 in intervalas:
        tikslumas = 0
        duom_kiekis = len(duomenu_matrica[0])
        for x in range(duom_kiekis):
          a = neuronas.grazinti_a([w0,w1,w2],[duomenu_matrica[0][x],
                duomenu_matrica[1][x]])
          if(neuronas.slenkstine_aktivacija(a)==duomenu_matrica[2][x]):
            tikslumas+=1
        if(tikslumas == duom_kiekis):
          print(f"w1:{w1}, w2:{w2}, poslinkis:{w0}")
```

Funkcija „rasti_svorius“ (3 kodo fragmentas) su slenkstine aktyvacija trimis ciklais pereina per visas įmanomas kombinacijas. Kintamasis „tikslumas“ seka kiek neuronas su atitinkama svorių kombinacija teisingai klasifikuoja stebėjimų. Tos kombinacijos, kurios „tikslumas“ lygus stebėjimų skaičiui yra atspausdinamos.

Funkcija inicijuojama pateikiant „Neuronas“ klasės objektą, duomenų matricą(požymių ir klasių sąrašus) ir intervalą (4 kodo fragmentas).

4 kodo fragmentas. Svorių parinkimo funkcijos argumentai.

```
neuronas = Neuronas(2)
duomenu_matrica = [x1,x2,t]
intervalas = [x for x in range(-10, 10, 1)]
```

b) atsitiktinės reikšmės iš intervalo.

Generuoti atsitiktines reikšmes pasitelkta paketas „random“. Šio paketo funkcija „random.uniform(x1,x2)“ generuoja pseudo atsitiktinius skaičius tolygiajame skirstinyje nuo x1 iki x2 (5 kodo fragmentas). Neurono svorių paieškos funkcijoje x1 ir x2 atitinka argumentai „pirm“ ir „pask“ atitinkamai.

5 kodo fragmentas. Atsitiktinių svorių parinkimo funkcija.

```
def rasti_svorius(neuronas, duomenu_matrica, pirm, pask):
  for x in range(9261):
    w0 = random.uniform(pirm, pask)
    w1 = random.uniform(pirm, pask)
    w2 = random.uniform(pirm, pask)
    tikslumas = 0
    duom_kiekis = len(duomenu_matrica[0])
    for x in range(duom_kiekis):
      a = neuronas.grazinti_a([w0,w1,w2],[duomenu_matrica[0][x],
                duomenu_matrica[1][x]])
      if(neuronas.slenkstine_aktivacija(a)==duomenu_matrica[2][x]):
        tikslumas+=1
    if(tikslumas == duom_kiekis):
      print(f"w1:{w1}, w2:{w2}, poslinkis:{w0}")
```

2.1. Slenkstinė ir Sigmoidinė aktyvacijos funkcijos.

Abiejų Aktyvacijos funkcijų argumentai a yra iš visų realiųjų skaičių aibės. Tačiau slenkstinė gražina 0 kai argumentas a yra mažesnis už slenkstinę reikšmę 0, kitu atveju gražina 1. Sigmoidinė aktyvacijos funkcija gražina reikšmes intervale (0; 1). Tai naudinga norint interpretuoti neurono rezultatą kaip klasės 1 tikimybę.

Taigi, naudojant sigmoidinę aktyvaciją reikia suapvalinti neurono rezultatą ir tada gaunama stebėjimo klasė(6 kodo fragmentas). Tai padaroma naudojant „round(x,0)“ funkciją.

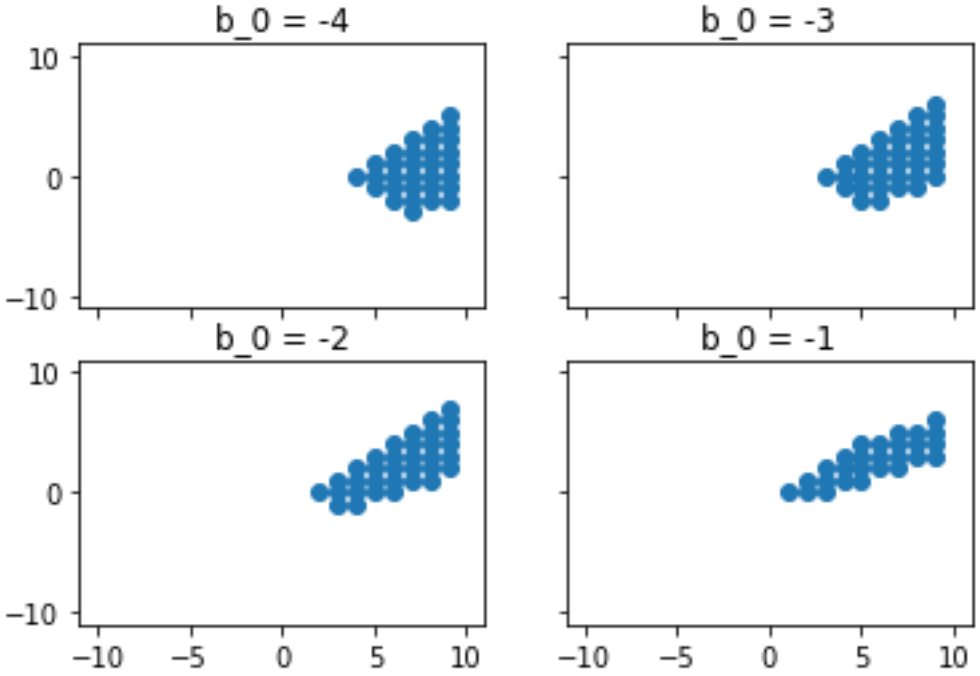
6 kodo fragmentas. Neurono rezultato interpretavimas skirtingoms aktyvacijos funkcijoms.

```
# slenkstine_aktivacija
if(neuronas.slenkstine_aktivacija(a)==duomenu_matrica[2][x]):
        tikslumas+=1

# sigmoidine_aktivacija
klase = round(neuronas.sigmoidine_aktivacija(a),0)
if(klase == duomenu_matrica[2][x]):
        tikslumas+=1
```

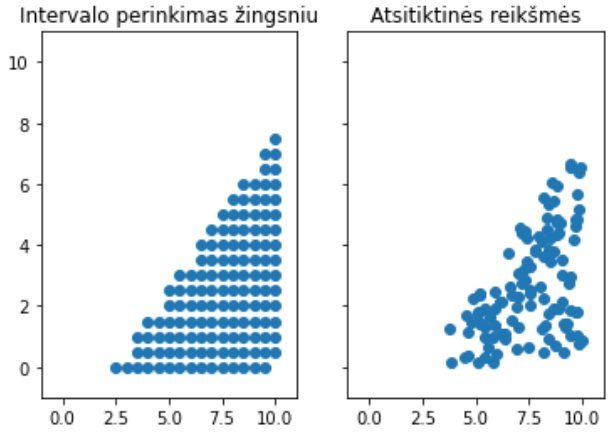
2.2. Rezultatai.

Svorius ir poslinkį galima vaizduoti kaip aibę koordinačių sistemoje. Kadangi turima tris demencijas, du svoriai ir poslinkis, fiksuojamas svoris (pažymėtas „b_0“), o koordinačių plokštumoje atidedami taškai (w_1, w_2). 1 paveiksle pateikiamas slenkstinės aktyvacijos funkcijos neurono svorių taškai (w_1, w_2), kurie teisingai klasifikuoja duomenis.



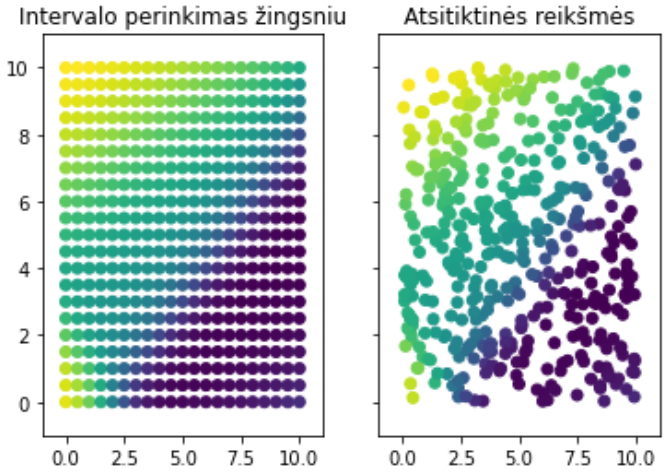
1 pav. Dalis tinkamų neurono svorių imčiai.

Abiejų neurono svorių parinkimo strategijų rezultatų vizualizavimui fiksuotas poslinkis -3. Iš pirmojo eksperimento matyti, kad intervale [0; 10] gaunama sritis sprendinių. Pirmiausia panaudojus slenkstinę aktyvaciją pereinamas intervalas žingsniu 0,5 (2 paveikslas).



2 pav. Slenkstinės aktyvacija. Intervalas [0; 10].

Sigmoidinės aktyvacijos atveju, turima funkcijos reikšmė. Vizualizacijai padaroma prielaida, kad funkcijos reikšmė yra spėjama klasė, t.y. neapvalinama neurono aktyvacijos funkcijos reikšmė. Tada skaičiuojama vidutinė kvadratinė paklaida nuo tikrosios klasės visai duomenų aibei. Gauta paklaida vizualizacijoje pateikiamas spalviniu indikatoriumi. (2 paveikslas).



2 pav. Sigmoidinė aktyvacija. Intervalas [0; 10].

Remiantis eksperimentais (1 priedas) galima teigti, kad rastos sprendinių aibės yra tolydžios. Šiems duomenims imant, bet kuriuos taškus aprėžtus kraštiniais tinkamų svorių taškais gausime tinkamą neurono svorių sprendinį.

3. Svorių radimas grafiniu būdu.

Naudojant slenkstinę aktyvacijos funkciją reikia išspręsti nelygybių sistemą

$$-0{,}3w_1 + 0{,}6w_2 + w_0 < 0$$

$$0{,}3w_1 - 0{,}6w_2 + w_0 < 0$$

$$1{,}2w_1 - 1{,}2w_2 + w_0 \geq 0$$

$$1{,}2w_1 + 1{,}2w_2 + w_0 \geq 0$$

Iš pirmųjų dviejų lygčių gaunama, kad $w_0 < 0$. Fiksuojamas poslinkis -2. Įsistačius reikšmę gaunamos nelygybės:

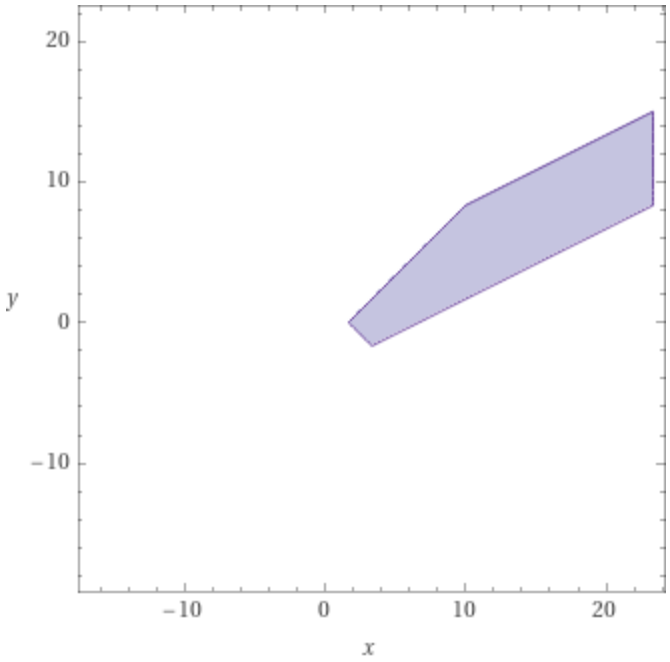
$$w_2 < \frac{2 + 0{,}3w_1}{0{,}6}$$

$$w_2 > \frac{0{,}3w_1 - 2}{0{,}6}$$

$$w_2 \leq \frac{1{,}2w_1 - 2}{1{,}2}$$

$$w_2 \geq \frac{2 - 1{,}2w_1}{1{,}2}$$

Grafiškai sprendinių aibė (3 paveikslas) sutampa su pirmojo eksperimento svorių aibe(1 paveikslas).



3 pav. Sprendinių aibė fiksuojant poslinkį -2.

4. Patikrinimui pasirinkus tašką (20;10) su poslinkiu -2 gaunama:

$$-0{,}3 * 20 + 0{,}6 * 10 - 2 < 0$$

$$0{,}3 * 20 - 0{,}6 * 10 - 2 < 0$$

$$1{,}2 * 20 - 1{,}2 * 10 - 2 \geq 0$$

$$1{,}2 * 20 + 1{,}2 * 10 - 2 \geq 0$$

Iš tiesų tai yra lygties sprendinys:

$$-2 < 0$$

$$-2 < 0$$

$$10 \geq 0$$

$$34 \geq 0$$

Išvados.

Neurono modelio svorių parinkimo uždavinį galima laikyti nelygybių sistemos uždaviniu. Sprendinys yra n demencijų turinti aibė, kur n yra svorių, įskaičiuojant ir poslinkį, skaičius. Šią aibę galima atrasti nagrinėjant tam tikrą intervalą, tačiau nėra garantijos, kad intervale aibė egzistuoja. Naudojant sigmoidinę aktyvacijos funkciją intervalo nagrinėjimui, pastebima, kad paklaidos nuo tikrosios klasės reikšmės turi tendenciją mažėti, svorių reikšmėms artėjant prie teisingų sprendinių aibės.

Šaltiniai.

McCulloch, Warren S., and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." *The bulletin of mathematical biophysics* 5.4 (1943): 115-133.

Naudoti „Python“ paketai.

math – matematinių funkcijų skaičiavimo paketas.

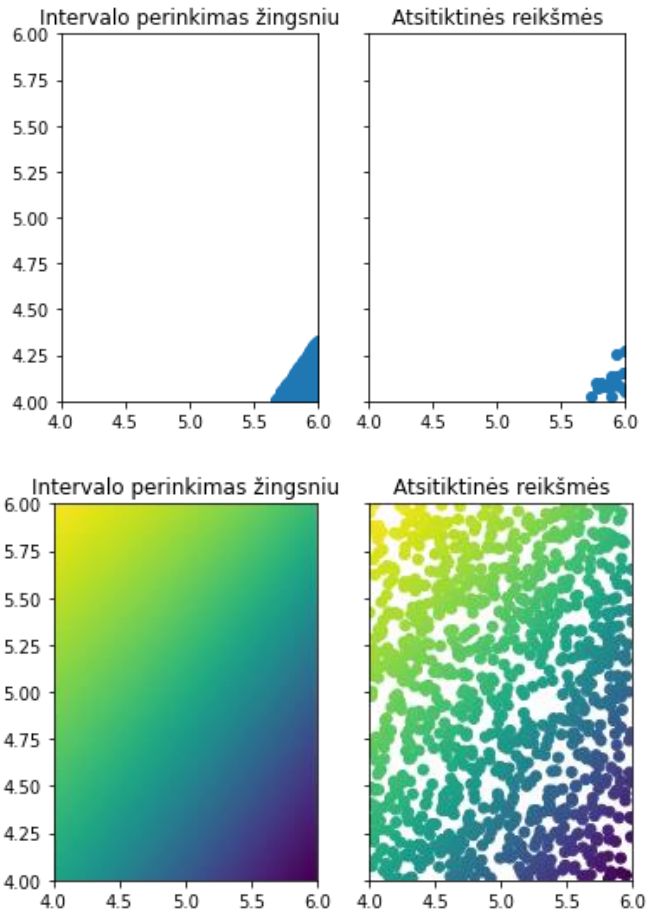
random – pseudo atsitiktinių skaičių generavimo paketas.

numpy – matricinių duomenų struktūrų modeliavimo paketas.

matplotlib – duomenų vizualizavimo paketas.

1 priedas.

Fiksuojamas poslinkis -2. Intervalas [4;6]. Žingsnis 0,01. Žingsninė strategija vizualizuojama kaip tolydi aibė.



2 priedas. Pilnas programos kodas.

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


x1 = [-.3,0.3,1.2,1.2]
x2 = [.6,-.6,-1.2,1.2]
t = [0,0,1,1]


class Neuronas:
  def __init__(self, iejimai):
    self.iejimai = iejimai

  def grazinti_a(self, svoriai, duomenys):
    a = 0 + svoriai[0]
    for i in range(self.iejimai):
      a += svoriai[i+1] * duomenys[i]
    return a
  def slenkstine_aktivacija(self,a):
    if(a<0):
      return 0
    elif(a>=0):
      return 1
  def sigmoidine_aktivacija(self,a):
    return 1/(1+math.exp(-a))


neuronas = Neuronas(2)
duomenu_matrica = [x1,x2,t]
intervalas = [x for x in range(-10, 11, 1)]

def rasti_svorius(neuronas, duomenu_matrica, intervalas):
  for w0 in intervalas:
    for w1 in intervalas:
      for w2 in intervalas:
        i+=1
        tikslumas = 0
        duom_kiekis = len(duomenu_matrica[0])
        for x in range(duom_kiekis):
          a = neuronas.grazinti_a([w0,w1,w2],[duomenu_matrica[0][x],duomen
u_matrica[1][x]])
          if(neuronas.slenkstine_aktivacija(a)==duomenu_matrica[2][x]):
            tikslumas+=1
```

```
        # print(f"tikslumas:{tikslumas}/{duom_kiekis}")ss
        if(tikslumas == duom_kiekis):
          print(f"w1:{w1}, w2:{w2}, poslinkis:{w0}")

rasti_svorius(neuronas, duomenu_matrica, intervalas)


neuronas = Neuronas (2)
duomenu_matrica = [x1,x2,t]
pirm = -10
pask = 10
def rasti_svorius(neuronas, duomenu_matrica, pirm, pask):
  for x in range(9261):
    w0 = random.uniform(pirm, pask)
    w1 = random.uniform(pirm, pask)
    w2 = random.uniform(pirm, pask)
    tikslumas = 0
    duom_kiekis = len(duomenu_matrica[0])
    for x in range(duom_kiekis):
      a = neuronas.grazinti_a([w0,w1,w2],[duomenu_matrica[0][x],duomenu_ma
trica[1][x]])
      if(neuronas.slenkstine_aktivacija(a)==duomenu_matrica[2][x]):
        tikslumas+=1
    if(tikslumas == duom_kiekis):
      print(f"w1:{w1}, w2:{w2}, poslinkis:{w0}")


rasti_svorius(neuronas, duomenu_matrica, pirm, pask)

neuronas = Neuronas (2)
duomenu_matrica = [x1,x2,t]
pirm = 4
pask = 6
intervalas = [x/100 for x in range(400, 601, 1)]

duomenys1 = []
duomenys2 = []

def rasti_svorius_rand(neuronas, duomenu_matrica, pirm, pask):
  for x in range(1000):
    w0 = -2
    w1 = random.uniform(pirm, pask)
    w2 = random.uniform(pirm, pask)
    tikslumas = 0
    duom_kiekis = len(duomenu_matrica[0])
```

```
    for x in range(duom_kiekis):
      a = neuronas.grazinti_a([w0,w1,w2],[duomenu_matrica[0][x],duomenu_ma
trica[1][x]])
      if(neuronas.slenkstine_aktivacija(a)==duomenu_matrica[2][x]):
        tikslumas+=1
    if(tikslumas == duom_kiekis):
      duomenys2.append([w1,w2,w0])

def rasti_svorius_int(neuronas, duomenu_matrica, intervalas):
  w0 = -2
  for w1 in intervalas:
    for w2 in intervalas:
      tikslumas = 0
      duom_kiekis = len(duomenu_matrica[0])
      for x in range(duom_kiekis):
        a = neuronas.grazinti_a([w0,w1,w2],[duomenu_matrica[0][x],duomenu_
matrica[1][x]])
        if(neuronas.slenkstine_aktivacija(a)==duomenu_matrica[2][x]):
          tikslumas+=1
      if(tikslumas == duom_kiekis):
        duomenys1.append([w1,w2,w0])
rasti_svorius_rand(neuronas, duomenu_matrica, pirm, pask)
rasti_svorius_int(neuronas, duomenu_matrica, intervalas)

duomenys_int = np.array(duomenys1)
duomenys_rand = np.array(duomenys2)

fig, axs = plt.subplots(1, 2, sharex=True, sharey=True)
axs[0].scatter(duomenys_int[:,0], duomenys_int[:,1])
axs[0].set_title(f'Intervalo perinkimas žingsniu')
axs[0].set(xlim=(4,6), ylim=(4, 6))
axs[1].scatter(duomenys_rand[:,0], duomenys_rand[:,1])
axs[1].set_title(f'Atsitiktinės reikšmės')


neuronas = Neuronas (2)
duomenu_matrica = [x1,x2,t]
pirm = 4
pask = 6
intervalas = [x/100 for x in range(400, 601, 1)]

duomenys1 = []
duomenys2 = []

def rasti_svorius_rand(neuronas, duomenu_matrica, pirm, pask):
```

```
  for x in range(1000):
    w0 = -2
    w1 = random.uniform(pirm, pask)
    w2 = random.uniform(pirm, pask)
    tikslumas = 0
    duom_kiekis = len(duomenu_matrica[0])
    for x in range(duom_kiekis):
      a = neuronas.grazinti_a([w0,w1,w2],[duomenu_matrica[0][x],duomenu_ma
trica[1][x]])
      tikslumas += (neuronas.sigmoidine_aktivacija(a)-
duomenu_matrica[2][x])**2
    duomenys2.append([w1,w2,tikslumas/duom_kiekis])

def rasti_svorius_int(neuronas, duomenu_matrica, intervalas):
  w0 = -2
  for w1 in intervalas:
    for w2 in intervalas:
      tikslumas = 0
      duom_kiekis = len(duomenu_matrica[0])
      for x in range(duom_kiekis):
        a = neuronas.grazinti_a([w0,w1,w2],[duomenu_matrica[0][x],duomenu_
matrica[1][x]])
        tikslumas += (neuronas.sigmoidine_aktivacija(a)-
duomenu_matrica[2][x])**2
      duomenys1.append([w1,w2,tikslumas/duom_kiekis])
rasti_svorius_rand(neuronas, duomenu_matrica, pirm, pask)
rasti_svorius_int(neuronas, duomenu_matrica, intervalas)

duomenys_int = np.array(duomenys1)
duomenys_rand = np.array(duomenys2)
fig, axs = plt.subplots(1, 2, sharex=True, sharey=True)
axs[0].scatter(duomenys_int[:,0], duomenys_int[:,1], c=duomenys_int[:,2],
cmap='viridis')
axs[0].set_title(f'Intervalo perinkimas žingsniu')
axs[0].set(xlim=(4,6), ylim=(4, 6))
axs[1].scatter(duomenys_rand[:,0], duomenys_rand[:,1], c=duomenys_rand[:,2
], cmap='viridis')
axs[1].set_title(f'Atsitiktinės reikšmės')


def rasti_svorius(neuronas, duomenu_matrica, intervalas):
  for w0 in intervalas:
    for w1 in intervalas:
      for w2 in intervalas:
        tikslumas = 0
```

```
        duom_kiekis = len(duomenu_matrica[0])
        for x in range(duom_kiekis):
          a = neuronas.grazinti_a([w0,w1,w2],[duomenu_matrica[0][x],duomen
u_matrica[1][x]])
          if(neuronas.slenkstine_aktivacija(a)==duomenu_matrica[2][x]):
            tikslumas+=1
        # print(f"tikslumas:{tikslumas}/{duom_kiekis}")
        if(tikslumas == duom_kiekis):
          print(f"w1:{w1}, w2:{w2}, poslinkis:{w0}")
          duomenys.append([w1,w2,w0])


unique, counts = np.unique(duomenys[ :,2], return_counts=True)
unique = unique[2:]
print(dict(zip(unique, counts)))
i = len(unique)
while i !=0:
  if(i-4>=0):

    fig, axs = plt.subplots(2, 2, sharex=True, sharey=True)
    axs[0, 0].scatter(duomenys[duomenys[:, 2] == unique[0]][:,0], duomenys
[duomenys[:, 2] == unique[0]][:,1])
    axs[0, 0].set_title(f'b_0 = {unique[0]}')
    axs[0, 0].set(xlim=(-11, 11), ylim=(-11, 11))
    unique = unique[1:]
    axs[0, 1].scatter(duomenys[duomenys[:, 2] == unique[0]][:,0], duomenys
[duomenys[:, 2] == unique[0]][:,1])
    axs[0, 1].set_title(f'b_0 = {unique[0]}')
    unique = unique[1:]
    axs[1, 0].scatter(duomenys[duomenys[:, 2] == unique[0]][:,0], duomenys
[duomenys[:, 2] == unique[0]][:,1])
    axs[1, 0].set_title(f'b_0 = {unique[0]}')
    unique = unique[1:]
    axs[1, 1].scatter(duomenys[duomenys[:, 2] == unique[0]][:,0], duomenys
[duomenys[:, 2] == unique[0]][:,1])
    axs[1, 1].set_title(f'b_0 = {unique[0]}')
    unique = unique[1:]
    i -=4
  else:
    i = 0
```