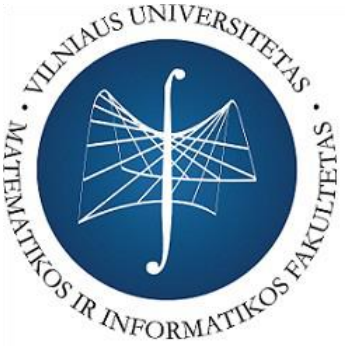


VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

DUOMENŲ MOKSLAS. BAKALAURAS

Dirbtinio neurono mokymas

Praktinė užduotis Nr. 2

Užduotį atliko: Ugnius Vilimas 3 kursas, 2 grupė

VU el. paštas: ugnius.vilimas@mif.stud.vu.lt

2023

Turinys

Tikslas

Užduoties tikslas - apmokyti vieną neuroną spręsti dviejų klasių uždavinį ir atlikti tyrimą su dviem duomenų aibėm.

Studento numerio paskutinis skaitmuo – 7. Todėl bus atliekamas 1 variantas (užduotis).
1) Neurono mokymui naudoti stochastinį gradientinį nusileidimą ir sigmoidinį neuroną.

Uždaviniai

- Parsisiųsti ir susitvarkyti duomenis. Pašalinti trūkstamų elementų turinčias eilutes, įsidėti į „Excel“ failą ir paruošti nuskaitymui.
- Nustatyti duomenų dalį, kuri bus skirta testavimui ir kuri mokymui.
- Aprašyti sigmoidinę aktyvacijos funkciją, pagal kurią bus atliekamas mokymas ir testavimas.
- Atlikti tikslumo ir paklaidos skaičiavimus savo duomenims.
- Pateikti išsamią analizę bei išvadas.

Duomenys

Užduotis buvo atliekama naudojant dvi duomenų aibes, kurios buvo pateiktos sąlygoje. Irisų ir krūties vėžio. Irisų duomenų šaltinis: https://archive.ics.uci.edu/dataset/53/iris .Krūties vėžio duomenų šaltinis: https://archive.ics.uci.edu/dataset/15/breast+cancer+wisconsin+original .

Irisų duomenys sudaryti iš 100 eilučių ir 5 stulpelių, kur vienas stulpelis nurodo klasę: Versicolor arba Virginica. Klasė Setona buvo pašalinta, nes jos analizei nereikėjo. Likę 4 stulpeliai buvo gėlės žiedo požymiai, parametrai. Mano pasirinkimu programoje į mokymo ir testavimo aibes buvo padalinti 70:30, tai yra 70 eilučių mokymui, o 30 eilučių - testavimui.

Krūties vėžio duomenys sudaryti iš 683 eilučių (jau išrinkus nepilnai užpildytas eilutes) ir 10 požymių, tarp kurių vienas yra, kuris rodo ar navikas yra piktybinis ar nepiktybinis. Jie yra užvadinti skaičiais 2 ir 4. Pas mane jie bus 0 – nepiktybinis, 1 – piktybinis. Mano pasirinkimu programoje į mokymo ir testavimo aibes buvo padalinti 70:30, tai yra 478 eilučių mokymui ir 205 eilučių testavimui.

Svarbios formulės

1. Įėjimo reikšmių ir svorių sandaugų sumos formulė: $a = \sum_{k=0}^{n} w_k x_k$.

2. Sigmoidinė aktyvacijos funkcija: $f(a) = \frac{1}{1+e^{-a}}$.

3. Paklaidos funkcija: $E(W) = \frac{1}{2}\sum_{i=1}^{m}(y_i - t_1)^2$.

4. Formulė, svoriams atnaujinti (pagal sidmoidinio neurono taisyklę):
$w_k = w_k - \eta(y_i - t_i)y_i(1 - y_i)(x_{ik})$.

Žymėjimai:

w_k – svoriai (svoris lygybėje yra senasis svoris, o svoris prieš lygybę yra naujasis).

η – mokymo greitis.

t_i – išvesties reikšmė iš mokymo duomenų.

y_i – aktyvacijos funkcijos skaičiuojama išvesties reikšmė.

x_{ik} – įvesties reikšmės.

Užduoties varianto paaiškinimai

Studento numerio paskutinis skaitmuo – 7. Variantas – 1.

Kas yra stochastinis gradientinis nusileidimas? – tai yra vienas iš gradientinio nusileidimo variantų, kuris yra naudojamas mašininio mokymo algoritmų optimizavimui. Stochastinis gradientinis nusileidimas dažniausiai yra naudojamas ir geriausiai veikia su pertekliniais duomenimis. Galime įsivaizduoti duomenis išdėstytus diagramoje, pvz su iris duomenimis. Stochastinio gradientinio nusileidimo principas yra tas, kad kiekvieną kartą atnaujinami parametrai „slope“ ir „intercept“. Pradžioje turime slope – 1, o intercept – 0. Norėdami atnaujinti šiuos parametrus, mes renkamės bet kokį tašką iš duomenų ir pagal slope ir intercept formules mes atnaujiname juos, taip atskirai, random būdu vykdome kiekvienam duomenų taškui. Formulės labai jautrios mokymo greičiui, todėl pradžioje geriau pradėti su didesniu ir mažinti. Toks nusileidimas cost funkcijoje labai varijuoja. Greitis yra geresni nei paketinio gradientinio nusileidimo. Šaltinis: https://youtu.be/vMh0zPT0tLI?si=2NWswDMMmQh22idu

Kas yra sigmoidinis neuronas? – Sigmoidinio neurono mokymo algoritmo tikslas yra nustatyti geriausias parametrų reikšmes taip, kad paklaida būtų kuo mažesnė. Svoriai keičiami yra pagal mokymo taisyklę , kurią gauname per keletą etapų ir tai geriausiai pavaizduoja nurodyti punktai skaidrėse.

- Naudojama **sigmoidinė aktyvacijos funkcija** $f(a_i) = \frac{1}{1+e^{-a_i}}$. Jos išvestinė $f'(a_i) = f(a_i)(1 - f(a_i))$.
- Tuomet funkcijos $E(W)$ **išvestinė** pagal w_k:
$$\frac{\partial E_i(W)}{\partial w_k} = (t_i - y_i) \times \left(-\frac{\partial f(a_i)}{\partial w_k}\right) = (t_i - y_i)(-f(a_i)(1 - f(a_i))$$
$$= (y_i - t_i)y_i(1 - y_i).$$
- Svoriai keičiami (atnaujinami) pagal šią **mokymo taisyklę** (formulę):
$$w_k := w_k - \eta(y_i - t_i)y_i(1 - y_i)(x_{ik})$$

Pagrindiniai žingsniai:

- Duomenų pasirinkimas: Pradžioje turite mokymo duomenų rinkinį, kuris sudarytas iš įvesties ir tikrosios išvesties porų.
- Inicializavimas: Tinklas pradeda su atsitiktinai sugeneruotais pradiniais svoriais ir slenksčiais.
- Epochos mokymas: Mokymo procesas vyksta per epochas, o kiekviena epocha - tai vienas mokymo ciklas, kurio metu naudojami visi mokymo duomenys.
- Iteracijos: Kiekvienoje iteracijoje ištraukiama atsitiktinė mokymo įvestis. Tinklas tada skaičiuoja prognozę ir klaidą.
- Gradiento skaičiavimas: Apskaičiuojamas gradiento vektorius, kuris nurodo, kaip turi keistis tinklo svoriai ir slenksčiai, kad sumažintų klaidą.
- Atnaujinimai: Tinklo svoriai ir slenksčiai yra atnaujinami, naudojant gradiento vektorių ir mokymo greitį. Tai reiškia, kad svoriai yra keičiami taip, kad klaida taptų mažesnė.
- Kartoti: Šis procesas kartojamas daug kartų per epochas, kol pasiekiama norima klaida arba pasibaigia nustatytas mokymo epochų skaičius.

Kas yra epocha? – tai neuronų mokymo proceso dalis, kurios metu apdorojamas visas mokymo duomenų rinkinys vieną kartą. **Stochastinio gradientinio nusileidimo** atveju viena epocha atitinka m iteracijų, čia m yra mokymo duomenų kiekis.

Svorių pasirinkimas

Pradiniai svoriai generuojami taip pat kaip ir pirmajame darbe, atsitiktinai. Pasinaudojus mokymo algoritmu svoriai yra atnaujinami pagal gautas klases ir vartotojo pasirenkamus parametrus funkcijoje „def Pasirinkimas():“. Ten reikia pasirinkti mokymo greitį ir epochų skaičių.

Programinis kodas su komentarais

Programa buvo sukurta naudojant “Python” programavimo kalbą. Žemiau matysime kodo dalis, jos bus su paaiškinimais.

Reikalingų bibliotekų importavimas.

```
#importuoju bibliotekos
import random
import matplotlib.pyplot as plt
import math
import numpy as np
import pandas as pd
import os
from sklearn.model_selection import train_test_split
```

1 pav. Bibliotekos

Parametru pasirinkimo funkcija, kurioje pagal atliekamą užduotį galima keisti: kuriuos duomenis naudosime, mokymo greitį bei epochų skaičių.

```
#cia funkcija pasirinkimui parametru
def pasirinkimas():
    duomenys = 1
    #0 jei iris, 1 jei cancer
    mokymo_greitis = 0.05
    #irasyti mokymo greiti
    epoch = 200
    #irasyti epochu skaiciu
    return duomenys, mokymo_greitis, epoch
```

2 pav. parametrai

Čia, kaip ir pirmame laboratoriniame darbe generuojame atsitiktinius skaičius, tačiau renkamės intervalą žymiai mažesnį ir su daugiau skaičių po kablelio, kaip ir buvo patarta skaidrėse ir paskaitų metu. Sekanti Funkcija generuoja svorius pavadinimu *weights* ir juos išsaugo.

```
#funkcija skaiciams generuoti
def generuoti_atsitiktinius_skaicius():
    return round(random.uniform(1, 1), 5)

#funkcija svoriams generuoti
def generuoja_svorius(number_of_weights):
    weights = []
    for i in range(number_of_weights):
        rand_number = generuoti_atsitiktinius_skaicius()
        weights.append(rand_number)
    return weights
```

3 pav. svorių generavimas

Funkcija gauti a reikšmes. Matome, kad yra paduodamos tokios reikšmės kaip: svoriai, poslinkis ir duomenų eilutės.

```
#funkcija gauti a vertes
def gauti_a_verte(row,weights,bias):
    a_val = np.dot(row, weights) + bias
    return a_val
```

4 pav. a vertės gavimas

Aktyvacijos funkcija. Mano variante reikėjo tik sigmoidinės.

```
#sigmoidine aktyvacijos funkcija
def sigmoid_funkcija(a_val):
    sigmoid_val = 1 / (1 + math.e ** (-a_val))
    return sigmoid_val
```

5 pav. aktyvacijos funkcija

Funkcijos pavaizduoti kaip keičiasi tikslumas ir paklaida pagal epochų skaičių. Į funkciją paduodami visi parametrai, kurių reikės vaizdavimui.

```
#funkcija sukurti 2 plot'us
def plot_duomenys(epoch_list,learning_accuracy_duomenys,testing_accuracy_duomenys,loss,testing_loss):

    plt.figure(figsize=(10, 5))
    #pirmas plot - mokymo tikslumas
    plt.subplot(1,2,1)
    plt.plot(epoch_list,learning_accuracy_duomenys, label="Mokymosi", color = "red")
    plt.plot(epoch_list,testing_accuracy_duomenys, label="Testavimo", color = "green")
    plt.xlabel("Epochos")
    plt.ylabel("Tikslumas (%)")
    plt.title("Gautas mokymo tikslumas")
    plt.legend()

    #antras plot - mokymo paklaida
    plt.subplot(1, 2, 2)
    plt.plot(epoch_list,loss,label="Mokymosi", color = "red")
    plt.plot(epoch_list,testing_loss,label="Testavimo ", color = "green")
    plt.xlabel("Epochos")
    plt.ylabel("Paklaida")
    plt.title("Gautos paklaidos")
    plt.legend()
    plt.show()
```

6 pav. tikslumo ir paklaidos plot'inimas

Čia įdėjau pagrindinę „training" funkcijos dalį, kur aprašytas sigmoidinio neurono mokymo algoritmas (pagal 1-ąjį variantą).Pirmiausia sukamas *for* ciklas, kuris eina per epochas, pagal tai kiek jų nurodė vartotojas pasirinkime. Čia yra sukamas dar vienas ciklas *for,* kur jis eina per kiekvieną duomenų eilutę, pagal tai kuriuos duomenis pasirinko naudoti vartotojas. Ciklas eina tik pagal sidmoidinę aktyvacijos funkciją. Tada prasideda mokymas. Čia delta yra sigmoidinio neurono mokymo formulė, pagal taisykles, kurias aprašiau anksčiau. Tuomet kai pasirinktos funkcijos klasės nesutampa su norima klase, mes vykdome dar kitą *for* ciklą, jau kuris paskaičiuoja svorius, pagal būtent tą formulę.

```
for i in range(epochs):
    for row_id, row in training_x.iterrows():
        a_val = gauti_a_verte(row, weights, bias)
        y = sigmoid_funkcija(a_val)

        if round(y) != training_class[row_id]:
            for value_id, weight in enumerate(weights):
                #sigmoidinio neurono formule
                delta = mokymo_greitis * (y - training_class[row_id]) * y * (1-y) * float(row.iloc[value_id])
                weight -= delta
                weights[value_id] = weight
```

7 pav. neurono mokymas

Čia pateikta paklaidos ir tikslumo funkcija. Mano variantui reikėjo skaičiuoti tik pagal sigmoidine funkciją. Paklaidos skaičavimas labai paprastas. Yra imamas skirtumas tarp neurono gautų ir norimų reikšmių pakeltų kvadratu. Taip pat pastebėjimas toks, kad sigmoidinės funkcijos reikšmės skaičiuojant paklaidą nėra apvalinamos.

```
#paklaidos funkcija
def accuracy(df, training_class,weights):
    bias = 1
    df = df.astype(float)

    got_classes = np.dot(df, weights) + bias
    for index,a in enumerate(got_classes):
        got_classes[index] = sigmoid_funkcija(a)

    equal_classes = 0
    loss = 0
    training_class = training_class.reset_index().iloc[:,-1]
    for e in range(len(training_class)):
        if round(got_classes[e]) == round(training_class.iat[e]):
            equal_classes += 1
        loss += (float(training_class.iat[e])-float(got_classes[e]))**2
    loss = round(loss * 0.5,2)
    accuracy_gotten = round(equal_classes/len(got_classes)*100,2)
    return accuracy_gotten,loss
```

8 pav. paklaidos ir tikslumo skaičiavimas

Sukurta funkcija, kuriai paduodami duomenys ir jau panaudotos reikšmės.

```
#funkcija paleisti programa pagal nurodytus duomenis
def main():
    duomenys, mokymo_greitis, epoch = pasirinkimas()

    if duomenys == 0:
        training_x, testing, class_training, class_testing = iris_duomenys()
        weights = generuoja_svorius(4)
    elif duomenys == 1:
        training_x, testing, class_training, class_testing = cancer_duomenys()
        weights = generuoja_svorius(9)

    weigths = training(training_x,class_training,testing,class_testing,weights,mokymo_greitis,epoch)
```

9 pav. funkcija programos vykdymui

Tyrimo rezultatai

Rezultatai pavaizduoti lentele, kuri turi 5 stulpelius, kurie paeiliui rodo, kokie duomenys buvo naudojami, koks buvo pasirinktas epochų skaičius, taip pat mokymosi greitis, na ir galiausiai atlikus neurono mokymą, kokie buvo tikslumas ir paklaida

1 lentelė. „Iris“ ir „Cancer“ duomenų geriausi tikslumai ir mažiausios paklaidos pagal epochų skaičių ir mokymo greitį

Duomenys	Epochų skaičius	Mokymosi greitis	Geriausias tikslumas (procentais)	Mažiausia paklaida
Iris	10	0,01	80	3,2
		0,05	86,67	2,55
		0,1	86,67	2,28
	50	0,01	83,33	3,57
		0,05	93,33	2,25
		0,1	96,67	1,65
	100	0,01	86,67	3,07
		0,05	93,33	1,4
		0,1	96,67	1,4
Cancer	10	0,01	75,61	19,97
		0,05	84,88	13,37
		0,1	85,37	11,93
	50	0,01	84,88	16,85
		0,05	85,37	13,45
		0,1	88,29	10,25
	100	0,01	82,93	17,4
		0,05	85,85	12,96
		0,1	89,27	9,82

Lentelė vaizduoja „Iris“ ir „Cancer“ duomenų neurono mokymo rezultatus. Pirmi 3 stulpeliai rodo parametrus tokius kaip duomenys, epochų skaičius ir mokymosi greitis, o paskutiniai 2 rodo gautus rezultatus.

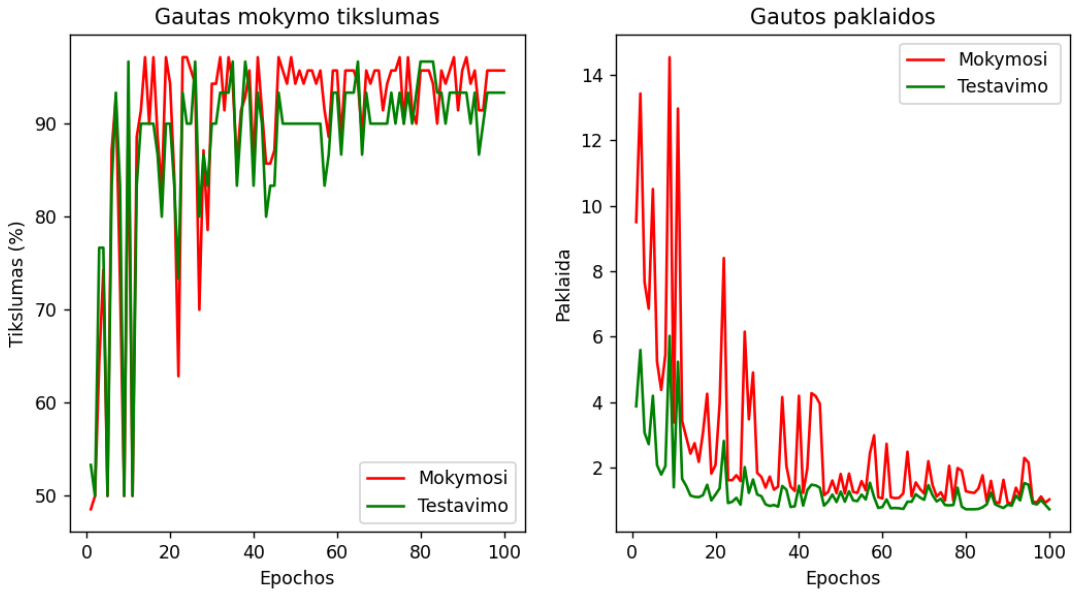
Galime aiškiai pastebėti, kad geresni tikslumai ir mažesnės paklaidos buvo su Iris duomenimis, kurie pirmiausia buvo daug mažesni ir paprastesni. Matome, kad esant 100 epochų ir mokymo greičiui 0,1 tikslumas yra geriausias, o paklaida mažiausia.

Matome, kad padidinus epochų skaičių, ir paklaida ir tikslumas tampa geresni. Tas pats vyksta ir tada, kai naudojame didesnį mokymo greitį.

Mažiausi tikslumas gaunama naudojant 10 epochų ir mokymo greitį 0,01. Tokia pat situacija yra ir su „Cancer“ duomenimis.

Pastebime tendencijas, kad kuo didesnis epochų skaičius, tuo geresnis ir tikslumas ir paklaida. Mokymo greitis taip pat turi tokią pat įtaką gaunamiems rezultatams. Duomenų dydis ir sudėtingumas taip pat buvo svarbus rezultatams. Mano manymu didžiausia įtaka vis tiek yra mokymo greičio, buvo pastebėta,

kad kardinaliai pakeitus mokymo greitį, kardinaliai keitėsi ir visi rezultatai. Tai buvo pastebėta prie geriausių klasifikavimo rezultatų.



10 pav. Mokymo tikslumo ir paklaidų diagrama naudojant „Iris“ duomenis, su mokymo greičiu 0,5 ir epochų skaičiumi 100

10 paveiksle matome dvi diagramas, kurios buvo padarytos su duomenų tikslumu ir paklaidomis, naudojant „Iris“ duomenis, mokymo greitį 0,5 ir epochų skaičių 100. Raudona linija yra mokymosi tikslumas ir paklaida, atitinkamai pagal diagramos pavadinimą, o žalia – testavimo.

Geriausi klasifikavimo rezultatai

Geriausią mokymo tikslumą ir paklaidą gauname pasirinkus Iris duomenis, epochų skaičių 100, mokymo greitį 0,5. Greičiausiai, kad buvo galima gauti ir dar geresniu, padidinus epochų skaičių, tačiau bandymų ir testavimų su daugiau nei 100 epochų bandyti nenorėjau.

```
Kai epocha yra: 1, gautas 50.0% tikslumas ir 10.1 paklaida
Su 1 epochos testavimo duomenimis gautas 50.0% tikslumas ir 4.45 paklaida
Svoriai: [0.13458401054302843, -0.6368849015655047, 0.41099338947034963, -0.7770230041297902]

Kai epocha yra: 2, gautas 50.0% tikslumas ir 17.4 paklaida
Su 2 epochos testavimo duomenimis gautas 50.0% tikslumas ir 7.45 paklaida
Svoriai: [0.43706437077334187, -0.49123864299900194, 1.0352389791986798, -0.5112546478915168]

Kai epocha yra: 3, gautas 50.0% tikslumas ir 16.84 paklaida
Su 3 epochos testavimo duomenimis gautas 50.0% tikslumas ir 7.23 paklaida
Svoriai: [-0.5543197567314353, -1.0101805339214045, 0.5062975453573348, -0.5974694709633355]

Kai epocha yra: 4, gautas 64.29% tikslumas ir 7.52 paklaida
Su 4 epochos testavimo duomenimis gautas 63.33% tikslumas ir 3.26 paklaida
Svoriai: [-0.4179830907663193, -0.9877346185595159, 0.9246613749544332, -0.3243610148076905]

Kai epocha yra: 5, gautas 57.14% tikslumas ir 8.02 paklaida
Su 5 epochos testavimo duomenimis gautas 50.0% tikslumas ir 3.38 paklaida
Svoriai: [-0.7706046580625656, -1.1633002711951679, 1.276901768492281, -0.057785673614562674]

Kai epocha yra: 6, gautas 50.0% tikslumas ir 13.22 paklaida
Su 6 epochos testavimo duomenimis gautas 50.0% tikslumas ir 5.61 paklaida
Svoriai: [-1.098557388949757, -1.2873325907585103, 1.4073769455204557, 0.1486197373866764]

Kai epocha yra: 7, gautas 65.71% tikslumas ir 6.84 paklaida
Su 7 epochos testavimo duomenimis gautas 70.0% tikslumas ir 2.8 paklaida
Svoriai: [-1.0758170249580399, -1.3088774111684405, 1.625172851523746, 0.30971061952645085]

Kai epocha yra: 8, gautas 91.43% tikslumas ir 3.68 paklaida
Su 8 epochos testavimo duomenimis gautas 90.0% tikslumas ir 1.6 paklaida
Svoriai: [-1.3109155375866672, -1.4710500994125204, 2.070212086285463, 0.629068956611415]

Kai epocha yra: 9, gautas 91.43% tikslumas ir 3.4 paklaida
Su 9 epochos testavimo duomenimis gautas 90.0% tikslumas ir 1.49 paklaida
Svoriai: [-1.4403884421699023, -1.563638628604589, 2.2421953344221626, 0.764212499402858]

Kai epocha yra: 10, gautas 92.86% tikslumas ir 3.12 paklaida
Su 10 epochos testavimo duomenimis gautas 90.0% tikslumas ir 1.54 paklaida
Svoriai: [-1.4327718035365822, -1.6068208855345145, 2.3545109259198194, 0.8692367892302023]
```

11 pav. geriausi klasifikavimo rezultatai. Nuo 1 iki 10 epochų.

```
Kai epocha yra: 91, gautas 94.29% tikslumas ir 1.49 paklaida
Su 91 epochos testavimo duomenimis gautas 93.33% tikslumas ir 0.74 paklaida
Svoriai: [-5.502848273758843, -4.8195483845979386, 7.188362052995901, 6.654059275756214]

Kai epocha yra: 92, gautas 95.71% tikslumas ir 1.02 paklaida
Su 92 epochos testavimo duomenimis gautas 93.33% tikslumas ir 0.94 paklaida
Svoriai: [-5.3999335136262365, -4.731087396335479, 7.3539856160356125, 6.749319730348685]

Kai epocha yra: 93, gautas 92.86% tikslumas ir 1.81 paklaida
Su 93 epochos testavimo duomenimis gautas 90.0% tikslumas ir 1.33 paklaida
Svoriai: [-5.325745720160338, -4.7106219462820915, 7.465375248191059, 6.8244390423502415]

Kai epocha yra: 94, gautas 95.71% tikslumas ir 0.91 paklaida
Su 94 epochos testavimo duomenimis gautas 93.33% tikslumas ir 0.86 paklaida
Svoriai: [-5.44386120309089, -4.8345431799835605, 7.375204800764224, 6.8222195165776633]

Kai epocha yra: 95, gautas 95.71% tikslumas ir 1.17 paklaida
Su 95 epochos testavimo duomenimis gautas 90.0% tikslumas ir 1.03 paklaida
Svoriai: [-5.398995463353251, -4.802652218616881, 7.430126038081117, 6.8374310938120995]

Kai epocha yra: 96, gautas 94.29% tikslumas ir 1.39 paklaida
Su 96 epochos testavimo duomenimis gautas 93.33% tikslumas ir 0.73 paklaida
Svoriai: [-5.608903448140398, -4.886411688289467, 7.327462854611452, 6.8136487619145665]

Kai epocha yra: 97, gautas 95.71% tikslumas ir 1.02 paklaida
Su 97 epochos testavimo duomenimis gautas 93.33% tikslumas ir 0.95 paklaida
Svoriai: [-5.482739645855882, -4.791661485147218, 7.459654369238155, 6.8684153965229475]

Kai epocha yra: 98, gautas 97.14% tikslumas ir 0.95 paklaida
Su 98 epochos testavimo duomenimis gautas 93.33% tikslumas ir 0.74 paklaida
Svoriai: [-5.615124368820067, -4.820178806890969, 7.432242807362396, 6.859697396246402]

Kai epocha yra: 99, gautas 97.14% tikslumas ir 0.91 paklaida
Su 99 epochos testavimo duomenimis gautas 93.33% tikslumas ir 0.86 paklaida
Svoriai: [-5.5587616409099185, -4.810665275984273, 7.483997554697471, 6.899874409983723]

Kai epocha yra: 100, gautas 95.71% tikslumas ir 1.13 paklaida
Su 100 epochos testavimo duomenimis gautas 90.0% tikslumas ir 1.02 paklaida
Svoriai: [-5.505549878619252, -4.854430650922287, 7.561653939876287, 6.9156755135059681

Geriausias rezultatas 96.67%
Su svoriais: [-5.505549878619252, -4.854430650922287, 7.561653939876287, 6.915675513505968]
```

12 pav. geriausi klasifikavimo rezultatai. Nuo 91 iki 100 epochos.

```
Kai epocha yra: 67, gautas 95.71% tikslumas ir 1.18 paklaida
Su 67 epochos testavimo duomenimis gautas 96.67% tikslumas ir 0.74 paklaida
Svoriai: [-4.933334912527762, -3.8770169056374213, 6.384364841266879, 5.595481646540004]
```

13 pav. 67 epochoje, užfiksuotas geriausias tikslumas bei paklaida

Čia su „Iris“ duomenimis esant 100 epochų ir 0,5 mokymo greičiu gauname geriausią tikslumą kuris yra 96,67% ir tik 0,74 paklaida. Nors ir galėjome pastebėti 12 paveiksle, kad 96 epochoje yra geresnė paklaida (mažesnė), tačiau ten tikslumas yra ženkliai blogesnis, todėl kaip geriausią stebėjimą, pasirinkau šį. Jame yra nurodyti sugeneruoti svoriai, taip pat tikslumai ir paklaidos testavimo ir mokymo.

Išvados

Iš gautų rezultatų galime pamatyti tam tikras tendencijas. Neurono mokymo tikslumas ir paklaida priklauso nuo keleto dalykų. Pirmiausia yra svarbūs duomenys. Iris duomenų tikslumas buvo ženkliai geresnis, taip pat paklaidos žymiai mažesnės. Kitas svarbus parametras – epochų skaičius. Iš lentelės galėjome pastebėti, kad kuo daugiau epochų, tuo geresnius rezultatus gauname. Na ir manyčiau pats svarbiausias parametras yra mokymosi greitis, kuris atrodo, kad darė didžiausią įtaką tikslumui ir paklaidai. Aišku svarbu suprasti, kad mokymosi greitis gali būti ir per didelis, todėl svarbu surasti aukso viduriuką. Mūsų atveju mokymosi greitis 0,1 buvo pats geriausias. Tačiau patestavus su 0,5 mokymo greičiu, padariau kitą išvadą, kad jis buvo dar geresnis negu 0,1.

Kodas

```
#importuoju bibliotekos

import random

import matplotlib.pyplot as plt

import math

import numpy as np

import pandas as pd

import os

from sklearn.model_selection import train_test_split


#cia funkcija pasirinkimui parametru

def pasirinkimas():

    duomenys = 1

    #0 jei iris, 1 jei cancer

    mokymo_greitis = 0.05

    #irasyti mokymo greiti
```

```
    epoch = 200
    #irasyti epochu skaiciu
    return duomenys, mokymo_greitis, epoch

#funkcija skaiciams generuoti
def generuoti_atsitiktinius_skaicius():
    return round(random.uniform(-1, 1), 5)

#funkcija svoriams generuoti
def generuoja_svorius(number_of_weights):
    weights = []
    for i in range(number_of_weights):
        rand_number = generuoti_atsitiktinius_skaicius()
        weights.append(rand_number)
    return weights

#funkcija gauti a vertes
def gauti_a_verte(row,weights,bias):
    a_val = np.dot(row, weights) + bias
    return a_val

#sigmoidine aktyvacijos funkcija
def sigmoid_funkcija(a_val):
    sigmoid_val = 1 / (1 + math.e ** (-a_val))
    return sigmoid_val

#funkcija sukurti 2 plot'us
def plot_duomenys(epoch_list,learning_accuracy_duomenys,testing_accuracy_duomenys,loss,testing_loss):

    plt.figure(figsize=(10, 5))
    #pirmas plot - mokymo tikslumas
```

```
    plt.subplot(1,2,1)
    plt.plot(epoch_list,learning_accuracy_duomenys, label="Mokymosi", color = "red")
    plt.plot(epoch_list,testing_accuracy_duomenys, label="Testavimo", color = "green")
    plt.xlabel("Epochos")
    plt.ylabel("Tikslumas (%)")
    plt.title("Gautas mokymo tikslumas")
    plt.legend()

    #antras plot - mokymo paklaida
    plt.subplot(1, 2, 2)
    plt.plot(epoch_list,loss,label="Mokymosi", color = "red")
    plt.plot(epoch_list,testing_loss,label="Testavimo ", color = "green")
    plt.xlabel("Epochos")
    plt.ylabel("Paklaida")
    plt.title("Gautos paklaidos")
    plt.legend()
    plt.show()

#neurono mokymas
def training(training_x, training_class, testing, class_testing, weights, mokymo_greitis, epochs):
    training_x = training_x.astype(float)
    bias = 1
    learning_accuracy_duomenys = []
    testing_accuracy_duomenys = []
    epoch_list = []
    loss_duomenys = []
    testing_loss_duomenys = []
    best_training_result = 0
    best_weights = []
    for i in range(epochs):
        for row_id, row in training_x.iterrows():
```

```
        a_val = gauti_a_verte(row, weights, bias)
        y = sigmoid_funkcija(a_val)

        if round(y) != training_class[row_id]:
            for value_id, weight in enumerate(weights):
                #sigmoidinio neurono formule
                delta = mokymo_greitis * (y - training_class[row_id]) * y * (1-y) * float(row.iloc[value_id])
                weight -= delta
                weights[value_id] = weight

    accuracy_gotten,loss = accuracy(training_x,training_class,weights)
    testing_accuracy,testing_loss = accuracy(testing,class_testing,weights)

    if testing_accuracy > best_training_result:
        best_training_result = testing_accuracy
        best_weights = weights

    learning_accuracy_duomenys.append(accuracy_gotten)
    testing_accuracy_duomenys.append(testing_accuracy)
    loss_duomenys.append(loss)
    testing_loss_duomenys.append(testing_loss)
    epoch_list.append(i+1)
    print(f"Kai epocha yra: {i+1}, gautas {accuracy_gotten}% tikslumas ir {loss} paklaida")
    print(f"Su {i+1} epochos testavimo duomenimis gautas {testing_accuracy}% tikslumas ir {testing_loss}
paklaida")
    print(f"Svoriai: {weights}\n")

print(f"Geriausias rezultatas {best_training_result}%\nSu svoriais: {best_weights}")

plot_duomenys(epoch_list,learning_accuracy_duomenys,testing_accuracy_duomenys,loss_duomenys,testi
ng_loss_duomenys)
```

```
    return weights

#tikslumo funkcija
def accuracy(df, training_class,weights):
    bias = 1
    df = df.astype(float)

    got_classes = np.dot(df, weights) + bias
    for index,a in enumerate(got_classes):
        got_classes[index] = sigmoid_funkcija(a)

    equal_classes = 0
    loss = 0
    training_class = training_class.reset_index().iloc[:,-1]
    for e in range(len(training_class)):
        if round(got_classes[e]) == round(training_class.iat[e]):
            equal_classes += 1
        loss += (float(training_class.iat[e])-float(got_classes[e]))**2
    loss = round(loss * 0.5,2)
    accuracy_gotten = round(equal_classes/len(got_classes)*100,2)
    return accuracy_gotten,loss

#funkcija paleisti programa pagal nurodytus duomenis
def main():
    duomenys, mokymo_greitis, epoch = pasirinkimas()

    if duomenys == 0:
        training_x, testing, class_training, class_testing = iris_duomenys()
        weights = generuoja_svorius(4)
    elif duomenys == 1:
```

```
        training_x, testing, class_training, class_testing = cancer_duomenys()
        weights = generuoja_svorius(9)

    weigths = training(training_x,class_training,testing,class_testing,weights,mokymo_greitis,epoch)

#iris duomenu nuskaitymas
def iris_duomenys():
    file_path_iris = 'C:/Users/ugnen/Desktop/iris.xlsx'
    iris = pd.read_excel(file_path_iris, header=None, usecols=[0, 1, 2, 3, 4])
    # iris
    yi = iris.iloc[:, -1]
    xi = iris.iloc[:, 0:4]
    iris_training, iris_testing, iris_class_training, iris_class_testing = train_test_split(xi, yi,
test_size=0.3,random_state=0)

    return iris_training, iris_testing, iris_class_training, iris_class_testing

#cancer duomenu nuskaitymas
def cancer_duomenys():
    file_path_cancer = 'C:/Users/ugnen/Desktop/cancer.xlsx'
    column_names = ["1", "2", "3", "4", "5", "6", "7", "8", "9", "10"]
    cancer = pd.read_excel(file_path_cancer, names=column_names, usecols = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
    cancer["10"] = cancer["10"].replace([2, 4], [0, 1])

    y = cancer.iloc[:, -1]
    x = cancer.iloc[:, 0:9]
    cancer_training, cancer_testing, cancer_class_training, cancer_class_testing = train_test_split(x, y,
test_size=0.3,random_state=0)

    return cancer_training, cancer_testing, cancer_class_training, cancer_class_testing

if __name__ == "__main__":
```

main()