Vilniaus Universitetas

Perceptrono mokymas

Dirbtinio intelekto pagrindai

2 užduotis

Darbą atliko:

Dovydas Martinkus

Duomenų Mokslas 4 kursas 2 gr.

Vilnius, 2022

Turinys

1 Tikslas ir uždaviniai

Tikslas: apmokyti vieną dirbtinį neuroną spręsti nesudėtingą dviejų klasių uždavinį, atlikti rezultatų tyrimą su dviem duomenų aibėmis.

Uždaviniai:

Naudojamų duomenų aibių paruošimas.

Duomenų aibių padalijimas į mokymo ir testavimo aibes, dirbtinio neurono mokymo ir testavimo įgyvendinimas naudojant pasirinktą programavimo kalbą.

Klasifikavimo rezultatų priklausomybės nuo epochų skaičiaus, mokymo greičio parametro reikšmės, naudojamos aktyvacijos funkcijos tyrimas.

Visų tyrimų gautų rezultatų pateikimas lentelėse arba grafikuose.

2 Duomenys

Užduotyje naudoti 2 duomenų rinkiniai:

Irisų duomenų aibę sudaro 150 stebėjimų, iš kurių kiekvienas turi po 4 skaitinius požymius ir klasę, kuriai priklauso. Šiuose duomenyse yra trys klasės Setosa, Versicolor ir Virginica. Analizei naudotos tik dvi: Versicolor ir Virginica. Prieiga per internetą: https://archive.ics.uci.edu/ml/datasets/iris.

Krūties vėžio duomenų aibę sudaro 6873 stebėjimai, turintys po 10 skaitinių požymių ir klasės kintamąjį, kurį sudaro dvi klasės: 2 – nepiktybinis navikas , 4 – piktybinis navikas. Prieiga per internetą: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).

Abi duomenų aibes pasirinkta dalinti į mokymo ir testavimo aibes naudojant santykį 80-20.

Kai kurie ataskaitoje naudojami terminai:

Epocha – vienas algoritmo pilnas perėjimas pro visus turimus mokymo duomenis.

Iteracija – tai mokymo proceso dalis, kai į perceptroną (arba į dirbtinį neuroninį tinklą) pateikiamas vienas mokymo aibės duomuo.

3 Užduoties ataskaita

3.1 Programinis kodas

Užduotis atlikta naudojant programavimo kalbą „Python“. Žemiau pateiktas programinis kodas su tarpiniais paaiškinimais:

```
import pandas as pd
import numpy as np
import seaborn as sns
import math
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt



# pirmojo duomenų rinkinio nuskaitymas, pertvarkymas ir padalijimas į mokymo ir
testavimo aibes
x1 = pd.read_csv('iris.data', sep=",", header=None).values

x1 = x1[x1[:,4] != "Iris-setosa"]
x1 = np.column_stack([np.ones(x1.shape[0]),x1])
x1[:,-1] = np.where(x1[:,-1]=='Iris-virginica',0,1)

X1_train, X1_test, y1_train, y1_test = train_test_split(x1[:,0:-1],x1[:,-1],

train_size=0.8,random_state=123,stratify=x1[:,-1])


# antrojo duomenų rinkinio nuskaitymas, pertvarkymas ir padalijimas į mokymo ir
testavimo aibes
x2 = pd.read_csv('breast-cancer-wisconsin.data', sep=",", header=None).values

x2 = x2[x2[:,6] != "?"]
x2[:,-1] = np.where(x2[:,-1]==2,0,1)
x2 = x2[:,1:]
x2 = np.column_stack([np.ones(x2.shape[0]),x2])
x2 = x2.astype(float)

X2_train, X2_test, y2_train, y2_test = train_test_split(x2[:,0:-1],x2[:,-1],

train_size=0.8,random_state=123,stratify=x2[:,-1])
```

Atlikus reikalingus pertvarkymus, pirmojo duomenų rinkinio (irisų) mokymo aibėje turima 80 stebėjimų, testavimo aibėje – 20 stebėjimų. Antrojo duomenų rinkinio (krūties vėžio) mokymo ir testavimo aibėse turimi atitinkamai 546 ir 137 stebėjimai.

Užduotyje naudotos slenkstinė ir sigmoidinė aktyvacijos funkcijos:

```
def sigmoid(row,weights):
    # Sigmoidine aktyvacijos f-ja
    a = np.dot(row, weights)
    return 1/(1+math.exp(-a))


def threshold(row,weights):
    # Slenkstinė aktyvacijos f-ja
    a = np.dot(row, weights)
    return int(a >= 0)

# abi aktyvacijos funkcijos skirtos paduoti kaip tolimesnių funkcijų argumentas
```

Svoriai atnaujinti naudojantis formule:

$$w_{(k)}(t+1) = w_k(t) + \eta(t_i - y_i)x_{ik}$$

```
def update_weights(weights, row, y_true, y_pred, lrate):
    # Svorius atnaujinanti funkcija
    updated_weights = []
    for i in range(len(weights)):
        updated_weights.append(weights[i] + lrate*(y_true - y_pred)*row[i])
    return updated_weights
```

```
from matplotlib.ticker import MaxNLocator

def lineplot(y, xlabel,ylabel, title,c="b"):
    # funkcija, skirta nubraižyti linijinį grafiką
    fig, ax = plt.subplots(figsize=(8, 4))
    ax.plot(range(1,len(y)+1),y, alpha = 0.7,c=c)
    ax.set_xlabel(xlabel)
    ax.set_ylabel(ylabel)
    ax.set_title(title)
    ax.xaxis.set_major_locator(MaxNLocator(integer=True))
```

Kaip pradines parametrų reikšmes pasirinkta naudoti 20 epochų ir mokymosi greitį (angl. learning rate) lygų 0,5.

Klasifikavimo tikslumas naudojant mokymo ir testavimo aibes skaičiuotas padalijant teisingai priskirtų klasių skaičių iš bendro stebėjimų atitinkamoje aibėje skaičiaus.

```
def train_perceptron(X, y, activation, lrate, epochs, plot = True):
    weights = np.full(X.shape[1], 1) # pradiniai svoriai
    errors = []  # paklaidų sąrašas
    accuracies = [] # klasifikavimo tikslumų sąrašas
```

```
    for e in range(epochs):
        for i in range(len(X)):
            y_true = y[i]
            row = X[i]
            y_pred = round(activation(row, weights),0)
            weights = update_weights(weights, row, y_true, y_pred , lrate)


        accuracy, error = test_accuracy(X,y,weights,activation)
        errors.append(error) # sąrašas papildomas paklaida, gauta po kiekvienos
epochos
        accuracies.append(accuracy)

    if plot:
        names = {"sigmoid": "sigmoidinę", "threshold": "slenkstinę"}
        lineplot(errors, "Epocha", "Paklaida", "Epochų paklaida, naudojant " +
                 names[activation.__name__] +
                     " aktyvacijos f-ją","b")
        lineplot(accuracies, "Epocha", "Tikslumas", "Epochų tikslumas mokymo
duomenims, naudojant " +
                 names[activation.__name__] +
                     " aktyvacijos f-ją","orange")
    return errors, accuracies, weights




def test_accuracy(X_test, y_test, weights, activation):
    # Apskaičiuoja tikslumą testavimo duomenims, naudojant pasirinktą aktyvacijos
funkciją
    correct = []
    error = 0
    for i in range(len(X_test)):
        y_pred = round(activation(X_test[i],weights),0)
        correct.append(y_pred == y_test[i])
        error += (y_test[i] - y_pred)**2
    return round(np.mean(correct),2), round(error,2)
```

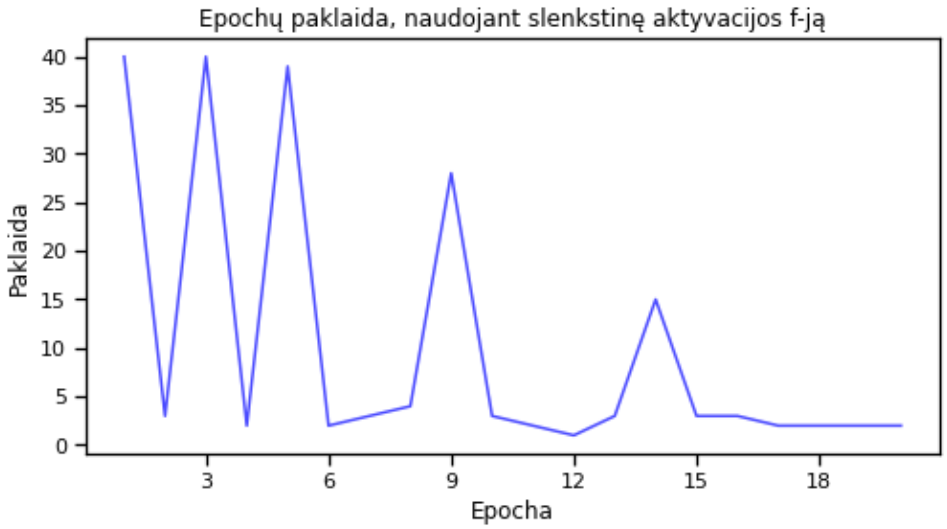
3.2 Tyrimo rezultatai

Pirmiausia naudotas pirmas (irisų) duomenų rinkinys.

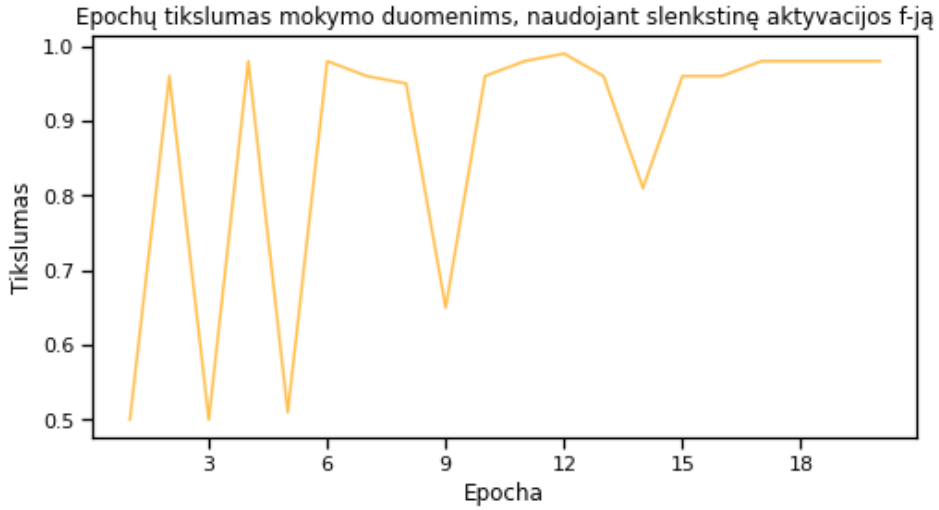
```
sns.set_context("notebook")
# kviečiama perceptrono mokymo f-ja su X1 duomenimis, slenkstine aktyvacijos f-ja
# 0.5 mokymo greičiu ir naudojant 20 epochų
errors, accuracies , weights = train_perceptron(X1_train, y1_train, threshold, 0.5,
20)
# perceptrono mokymas, naudojantis mokymo duomenimis ir slenstine aktyvacijos
funkcija

print("Galutiniai svoriai: ", [round(i,2) for i in weights],
      "\nGalutinė paklaida: ",  errors[-1],
      "\nGalutinis tikslumas: ",  accuracies[-1],
      "\nTikslumas testavimo duomenims: " ,  test_accuracy(X1_test, y1_test, weights,
threshold)[0],
      "\nPaklaida testavimo duomenims: " ,  test_accuracy(X1_test, y1_test, weights,
threshold)[1])
```

```
Galutiniai svoriai:  [11.5, 12.4, 20.1, -22.65, -23.0]
Galutinė paklaida:  2
Galutinis tikslumas:  0.98
Tikslumas testavimo duomenims:  0.9
Paklaida testavimo duomenims:  2
```

Paklaidos ir tikslumo priklausomybė nuo epochos pavaizduota grafiškai naudojant linijinę diagramą (atitinkamai 1 ir 2 pav.). Pastebimas paklaidos mažėjimas didėjant atliktų iteracijų skaičiui, tačiau su 0,5 mokymosi greičiu jis yra stipriai nepastovus: neretai sekančios epochos metu gauta paklaida daug didesnė už praėjusios. Tikėtina, kad ši pri ežastis atsirado dėl per didelio parinkto mokymosi greičio. Taip pat pastebimas stiprus ryšys tarp paklaidos ir tiksl umo: epochose su mažesnėmis paklaidomis gaunami ir didesni tikslumai.



1 pav. Paklaida pagal epochą, naudojant slenkstinę aktyvacijos funkciją (pirmas duomenų rinkinys)



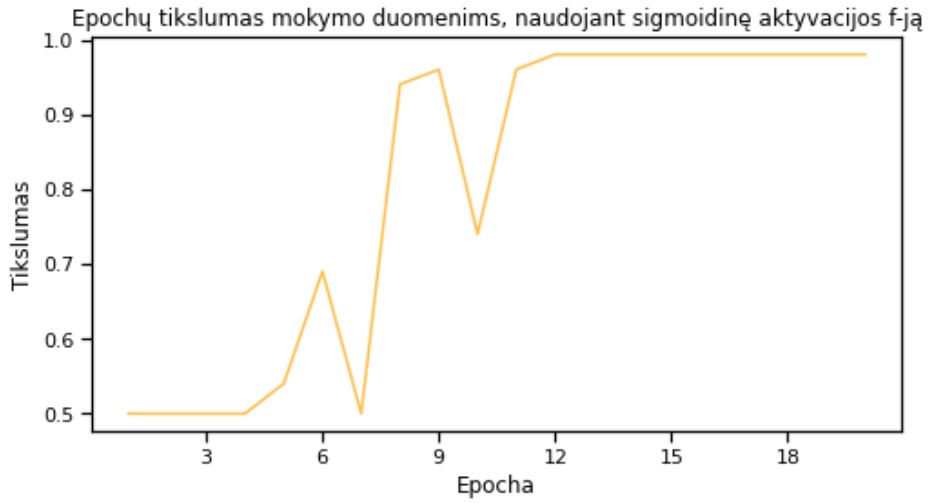
2 pav. Tikslumas pagal epochą, naudojant slenkstinę aktyvacijos funkciją (pirmas duomenų rinkinys)

```
errors, accuracies, weights = train_perceptron(X1_train, y1_train, sigmoid, 0.5, 20)
# sigmoidinė aktyvacijos fuhnkcija

print("Galutiniai svoriai: ", [round(i,2) for i in weights],
      "\nGalutinė paklaida: ",  errors[-1],
      "\nGalutinis tikslumas: ",  accuracies[-1],
      "\nTikslumas testavimo duomenims: " ,  test_accuracy(X1_test, y1_test, weights,
sigmoid)[0],
      "\nPaklaida testavimo duomenims: " ,  test_accuracy(X1_test, y1_test, weights,
sigmoid)[1])
```

```
Galutiniai svoriai:  [11.5, 12.4, 20.1, -22.65, -23.0]
Galutinė paklaida:  1.58
Galutinis tikslumas:  0.98
Tikslumas testavimo duomenims:  0.9
Paklaida testavimo duomenims:  1.96
```

Tokie pat grafikai nubraižyti vietoje slenkstinės aktyvacijos funkcijos naudojant sigmoidinę (atitinkamai 3 ir 4 pav.) . Tiek paklaidos, tiek tikslumo tendencijos išlieka gana panašios.

3 pav. Paklaida pagal epochą, naudojant sigmoidinę aktyvacijos funkciją (pirmas duomenų rinkinys)



4 pav. Tikslumas pagal epochą, naudojant sigmoidinę aktyvacijos funkciją (pirmas duomenų rinkinys)

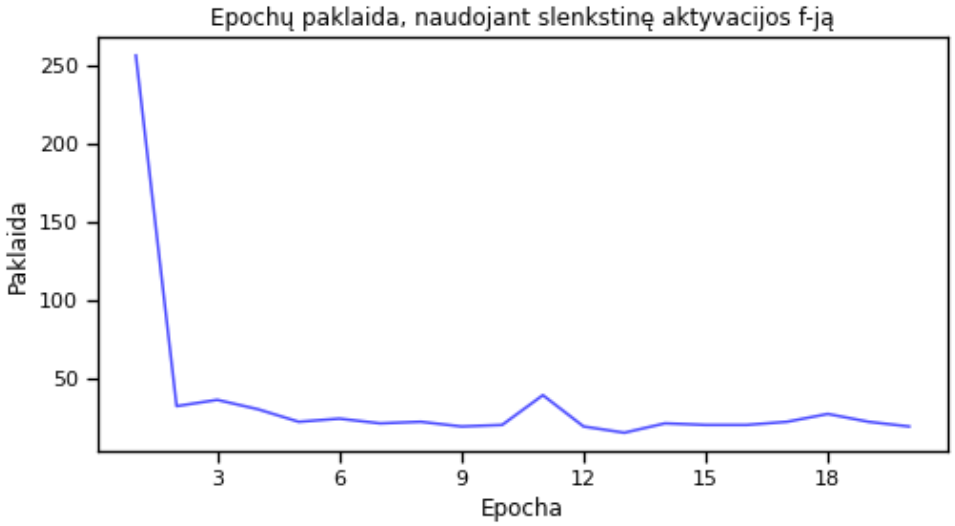
Analogiška procedūra kartota ir naudojant antrąjį duomenų rinkinį:

```
errors, accuracies, weights = train_perceptron(X2_train, y2_train, threshold, 0.5,
20)
```

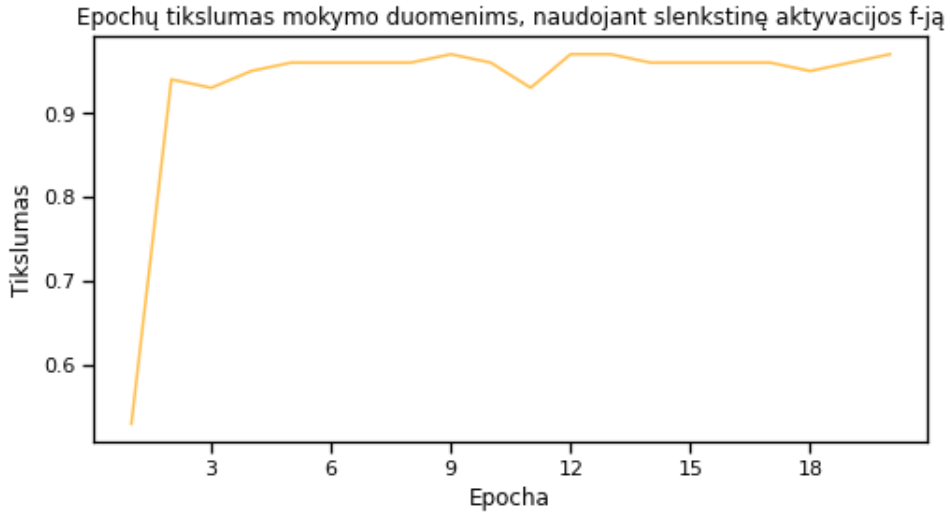
```
print("Galutiniai svoriai: ", [round(i,2) for i in weights],
      "\nGalutinė paklaida: ",  errors[-1],
      "\nGalutinis tikslumas: ",  accuracies[-1],
      "\nTikslumas testavimo duomenims: " ,  test_accuracy(X2_test, y2_test, weights,
threshold)[0],
      "\nPaklaida testavimo duomenims: " ,  test_accuracy(X2_test, y2_test, weights,
threshold)[1])
```

```
Galutiniai svoriai:  [-99.0, 1.0, 3.5, 6.5, 9.0, 2.5, 5.0, 1.5, 2.5, 9.0]
Galutinė paklaida:  19.0
Galutinis tikslumas:  0.97
Tikslumas testavimo duomenims:  0.97
Paklaida testavimo duomenims:  4.0
```

Naudojant slenkstinę aktyvacijos funkciją tikslumo didėjimas daug pastovesnis didėjant epochoms, lyginant su pir muoju duomenų rinkiniu (6 pav.). Taip pat galima matyti, kad paklaida stipriai mažėja (tikslumas didėja) iki 2 epoc hos, bet tolimesnėse epochose beveik išvis nustoja keistis.



5 pav. Paklaida pagal epochą, naudojant slenkstinę aktyvacijos funkciją (antras duomenų rinkinys)

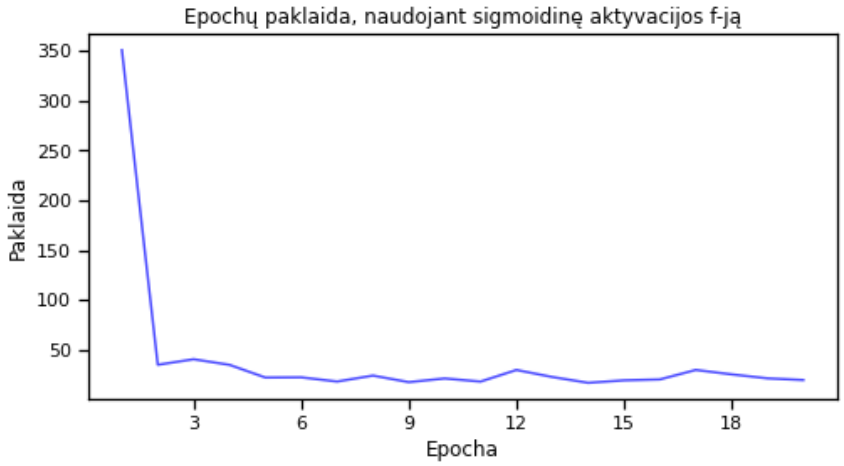


6 pav. Tikslumas pagal epochą, naudojant slenkstinę aktyvacijos funkciją (antras duomenų rinkinys)

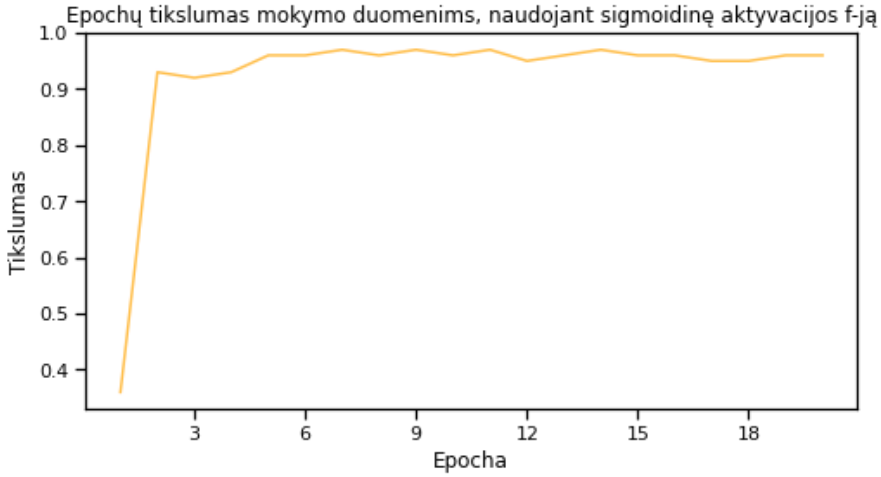
Lyginant slenkstinę ir sigmoidinę aktyvacijos funkcijas šįkart gauname beveik identiškus rezultatus (8 pav.).

```
errors, accuracies, weights = train_perceptron(X2_train, y2_train, sigmoid, 0.5, 20)


print("Galutiniai svoriai: ", [round(i,2) for i in weights],
      "\nGalutinė paklaida: ",  errors[-1],
      "\nGalutinis tikslumas: ",  accuracies[-1],
      "\nTikslumas testavimo duomenims: " ,  test_accuracy(X2_test, y2_test, weights,
sigmoid)[0],
      "\nPaklaida testavimo duomenims: " ,  test_accuracy(X2_test, y2_test, weights,
sigmoid)[1])
```

```
Galutiniai svoriai:  [-97.5, 0.0, 7.0, 4.5, 7.0, 4.0, 5.0, 2.5, -0.5, 10.5]
Galutinė paklaida:  18.02
Galutinis tikslumas:  0.96
Tikslumas testavimo duomenims:  0.96
Paklaida testavimo duomenims:  5.97
```



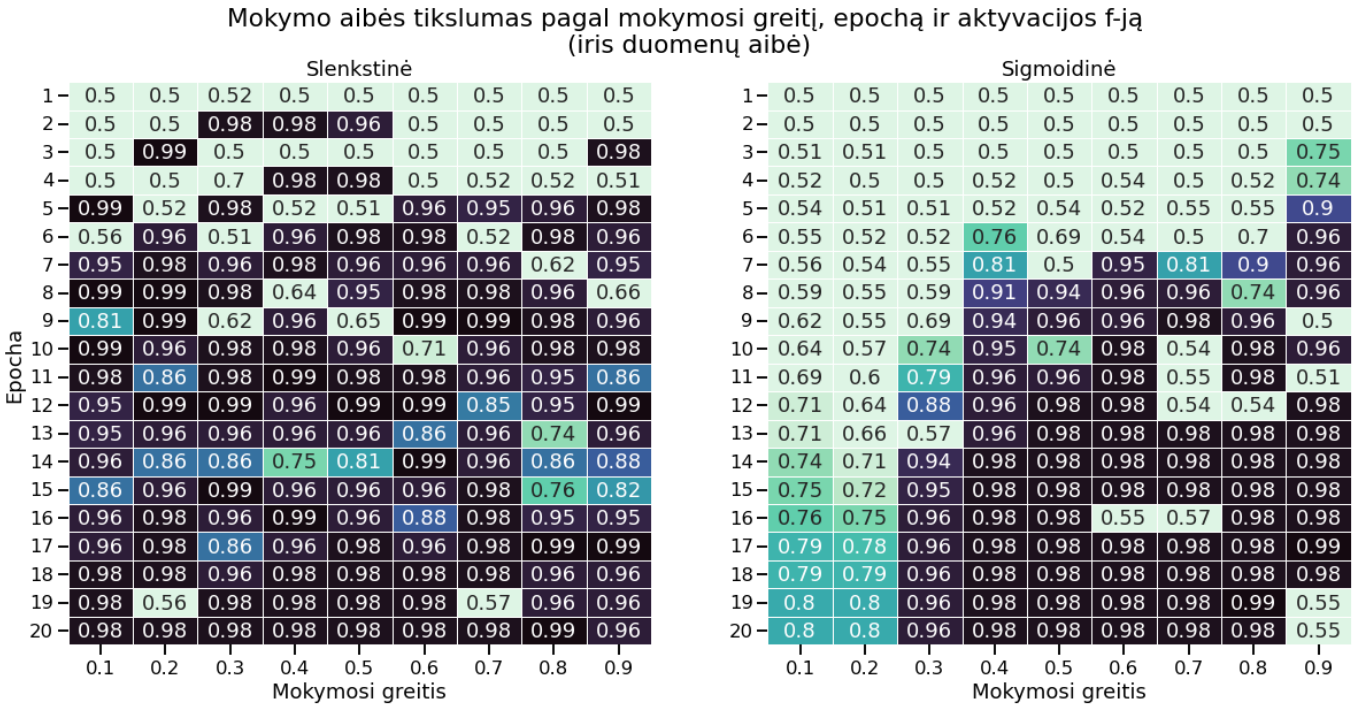
7 pav. Paklaida pagal epochą, naudojant sigmoidinę aktyvacijos funkciją (antras duomenų rinkinys)



8 pav. Tikslumas pagal epochą, naudojant sigmoidinę aktyvacijos funkciją (antras duomenų rinkinys)

```
# grafikas, parodantys klasifikavimo tikslumo priklausomybę nuo mokymo epochos ir
mokymosi greičio parametro reikšmių
def plot_heatmap(data, ax, title, left=True):
    cmap = sns.cm.mako_r
    plot=sns.heatmap(data, vmax=1,vmin=0.7,center=0.85,cbar=False,cmap=cmap,
            linewidth=1,annot=True, ax=ax)
    plot.set_title(title)
    plot.set_xlabel("Mokymosi greitis")
    if left:
        plot.set_ylabel("Epocha")
    plot.set_xticklabels(lrates)
    plot.set_yticklabels(range(1,results1.shape[0]+1))
    plot.tick_params(axis='x', rotation=0)
    plot.tick_params(axis='y', rotation=360)



lrates = np.round(np.arange(0.1, 1, 0.1),1)
results1 = np.zeros((20,len(lrates)))
results2 = np.zeros((20,len(lrates)))
for i,j in enumerate(lrates):
    _, accuracies, __ = train_perceptron(X1_train, y1_train, threshold, j, 20,
plot=False)
    results1[:,i] = accuracies

    _, accuracies, __ = train_perceptron(X1_train, y1_train, sigmoid, j, 20,
plot=False)
    results2[:,i] = accuracies



sns.set_context("talk")
fig, ax = plt.subplots(1,2,figsize=(22, 9))
fig.suptitle("Tikslumas pagal mokymosi greitį ir aktyvacijos f-ją " + "(iris)")
plot_heatmap(results1,ax[0], "Slenkstinė")
plot_heatmap(results2,ax[1], "Sigmoidinė", left=False)
```

Rezultatų priklausomybė nuo epochos, mokymosi greičio ir naudojamos aktyvacijos funkcijos pavaizduota grafiškai (9 pav.). Kaip galime matyti, didžiausias tikslumas mokymo duomenims gaunamas, pavyzdžiui, naudojant slenkstinę aktyvacijos funkciją, 0,2 mokymosi greitį ir 7 epochas. Taip pat matoma, kad kai kuriais atvejais po sekančios epochos gaunamas daug mažesnis tikslumas testavimo duomenims negu po praėjusios epochos. Nepaisant to, apskritai matoma tendencija gauti didesnį tikslumą didėjant epochų skaičiui. Lyginant aktyvacijos funkcijas bendrai galime teigti, kad mokant neuroną su sigmoidine aktyvacijos funkcija gaunami prastesni rezultatai.



9 pav. Mokymo aibės tikslumas pagal mokymosi greitį, epochą ir aktyvacijos f-ją (pirma duomenų aibė)

Nustačius geriausią variantą, išvedami galutiniai svoriai, tikslumas ir paklaida naudojant tiek mokymo duomenys, tiek testavimo duomenis:

```
errors, accuracy, weights = train_perceptron(X1_train, y1_train, threshold, 0.2, 7,
plot=False)

print("Galutiniai svoriai: ", [round(i,2) for i in weights],
      "\nTikslumas: ",  accuracy[-1],
      "\nPaklaida: ",  errors[-1],
      "\nTikslumas testavimo duomenims: " ,  test_accuracy(X1_test, y1_test, weights,
threshold)[0],
      "\nPaklaida testavimo duomenims: " ,  test_accuracy(X1_test, y1_test, weights,
threshold)[1])
```

```
Galutiniai svoriai:  [2.8, 3.6, 4.76, -6.18, -5.56]
Tikslumas:  0.98
Paklaida:  2
Tikslumas testavimo duomenims:  0.9
Paklaida testavimo duomenims:  2
```

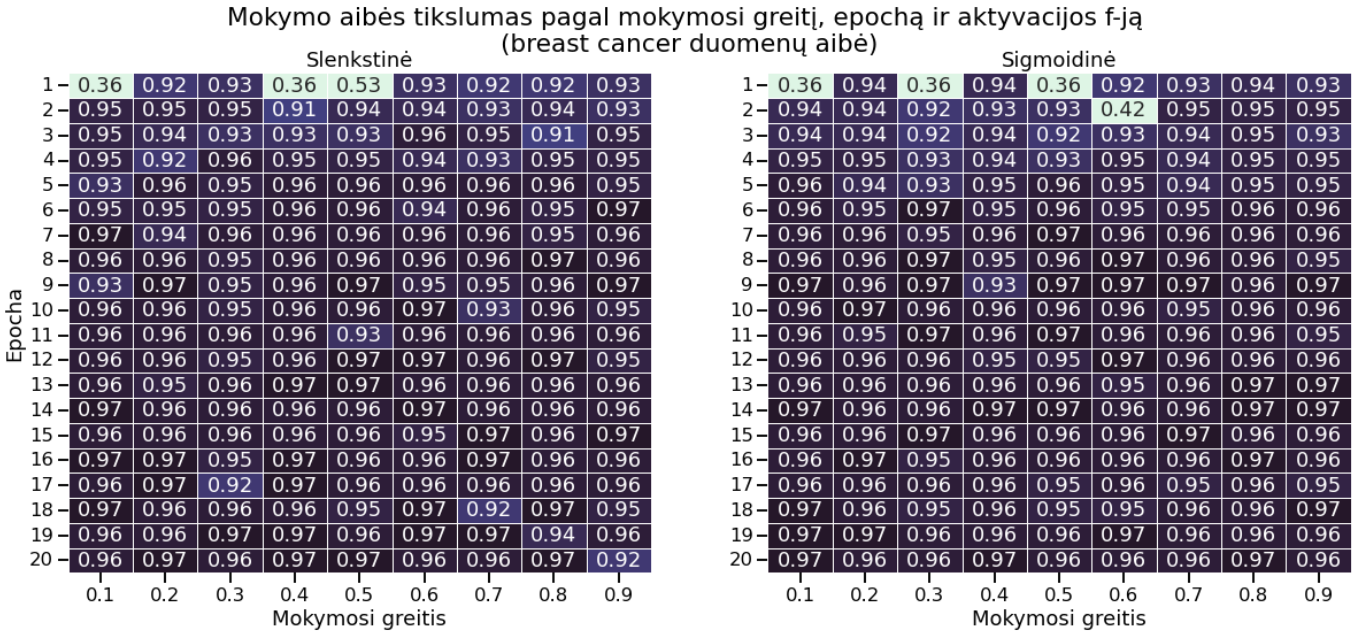
Taip pat pateikiama lentelė kokias klases kiekvienam testavimo aibės įrašui nustatė neuronas ir kokia klasė turėjo būti (1 priedas).

Tokia pati procedūra kartota ir naudojant antrąjį duomenų rinkinį:

```
lrates = np.round(np.arange(0.1, 1, 0.1),1)
results1 = np.zeros((20,len(lrates)))
results2 = np.zeros((20,len(lrates)))
for i,j in enumerate(lrates):
    _, accuracies, __ = train_perceptron(X2_train, y2_train, threshold, j, 20,
plot=False)
    results1[:,i] = accuracies

    _, accuracies, __ = train_perceptron(X2_train, y2_train, sigmoid, j, 20,
plot=False)
    results2[:,i] = accuracies
```

```
sns.set_context("talk")
fig, ax = plt.subplots(1,2,figsize=(22, 9))
plot_heatmap(results1,ax[0], "Slenkstinė")
plot_heatmap(results2,ax[1], "Sigmoidinė", left=False)
```

Tokio pat tipo grafikas nubraižytas ir naudojant antrą duomenų rinkinį (10 pav.). Šį kartą matome, kad abiem aktyvacijos funkcijoms su beveik visomis mokymosi greičio reikšmėmis iš karto (po 1 epochos) pasiekiamas tikslumas, kuris vėliau tik minimaliai didėja. Iš rezultatų taip pat galime teigti, kad naudojant antrąjį duomenų rinkinį gaunami geresni rezultatai, lyginant su rezultatais, gautais apmokant perceptroną pirmajam duomenų rinkiniui.



10 pav. Mokymo aibės tikslumas pagal mokymosi greitį, epochą ir aktyvacijos f-ją (antra duomenų aibė)

```
errors, accuracy, weights = train_perceptron(X2_train, y2_train, sigmoid, 0.2, 9,
plot=False)
```

```
print("Galutiniai svoriai: ", [round(i,2) for i in weights],
      "\nTikslumas: ",  accuracy[-1],
      "\nPaklaida: ",  errors[-1],
      "\nTikslumas testavimo duomenims: " ,  test_accuracy(X2_test, y2_test, weights,
sigmoid)[0],
      "\nPaklaida testavimo duomenims: " ,  test_accuracy(X2_test, y2_test, weights,
sigmoid)[1])
```

```
Galutiniai svoriai:  [-29.15, -1.21, 3.28, 1.5, 0.99, -0.61, 2.51, -0.04, 1.14, 2.0]
Tikslumas:  0.96
Paklaida:  21.35
Tikslumas testavimo duomenims:  0.96
Paklaida testavimo duomenims:  5.31
```

Antrajam duomenų rinkiniui taip pat pateiktas tikrų ir perceptrono pateiktų reikšmių palyginimas naudojant testavimo aibę (2 priedas).

4 Išvados

Remiantis tyrimo metu gautais rezultatais galima teigti, kad daugeliu atveju didėjant epochoms gautas didesnis klasifikavimo tikslumas (mažesnė paklaida). Nepaisant to, tyrimo metu rasta, kad perceptronas per mažą epochų skaičių (6-7 pirmajam duomenų rinkinui, 1-2 antrajam rinkinui), pasiekia tikslumą, kuris (beveik) nebegerėja tolimesnių epochų metu.

Taip pat pastebėta problema galinti kilti dėl to, jei parenkamas per didelis mokymosi greitis: svoriai atnaujinami per daug greitai ir tolimesnėje epochoje gauti rezultatai gali būti prastesni negu epochose prieš tai.

Rasta ir naudojamo duomenų rinkinio įtaka. Su beveik visomis epochų skaičiaus ir mokymosi greičio parametrų reikšmes naudojant antrąjį duomenų rinkinį gautas didesnis klasifikavimo tikslumas mokymo aibėje negu su pirmuoju duomenų rinkiniu. Galima priežastis, kodėl gaunamas šis rezultatas yra tai, kad antrojo duomenų rinkinio mokymo aibėje turima daugiau stebėjimų (vienos epochos metu svoriai atnaujinami daugiau kartų).

Priedas

Tikra reikšmė	Prognozuota reikšmė
0	0
0	0
0	0
0	0
0	0
1	1
1	1
0	0
0	0
1	1
1	0
1	1
0	0
1	1
1	1
1	1
0	0
1	0
0	0
1	1

1 priedas Tikros ir prognozuotos klasių reikšmės testavimo duomenims (pirma duomenų aibė)

Tikra reikšmė	Prognozuota reikšmė
0	0
0	0
1	1
0	1
0	0
1	1
1	1
0	0
1	1
1	1
0	0
0	0
0	0
1	0
0	0
0	0
0	0
1	1

0	0
0	1
1	1
1	1
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
1	1
0	0
0	0
0	0
0	0
1	1
1	1
0	0
0	0
0	0
1	1
0	0
0	0
0	0
0	0
0	0
0	0
0	0
1	1
0	0
0	0
0	0
1	1
0	0
0	0
1	1
1	0
1	1
1	1
1	0
0	0
0	0
0	0
1	1
0	0
0	0
0	0
0	0

0	0
1	1
0	0
1	1
1	1
0	0
0	0
1	1
0	0
1	1
1	1
1	1
1	1
0	0
0	0
0	0
0	0
0	0
0	0
1	1
0	0
1	1
0	0
0	0
1	1
0	0
1	1
0	0
0	0
1	1
1	1
1	1
0	0
0	0
0	0
0	0
0	0
0	0
1	1
1	1
1	0
0	0
0	0
1	1
0	0
1	1
0	0
0	0
0	0
0	0

0	0
0	0
0	0
1	1
1	1
1	1
1	1
0	0
0	0
0	0
0	0
1	1
0	0
0	0
1	1
0	0
1	1
0	0
0	0

2 priedas Tikros ir prognozuotos klasių reikšmės testavimo duomenims (antra duomenų aibė)