VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

STUDIJŲ PROGRAMA DUOMENŲ MOKSLAS 4 KURSAS

Vieno neurono mokymas sprendžiant klasifikavimo uždavinį

Darbo aprašas

Autorius: Austėja Ona Plančiūnaitė
VU el. p.: ona.plančiūnaitė@mif.stud.vu.lt

Vilnius
2024

Turinys

1 Įvadas

Praktinės užduoties tikslas yra apmokyti vieną neuroną spręsti dviejų klasių uždavinį ir atlikti tyrimą su dviem duomenų aibėmis. Darbe sukuriama programa, kuri įgyvendina vieno neurono mokymo ir testavimo procesą, sprendžiant klasifikavimo uždavinį. Atliekamas tyrimas, norint sužinoti kaip paklaidos reikšmės ir klasifikavimo tikslumas priklauso nuo epochų skaičiaus, tikrinama kaip rezultatai skiriasi nuo mokymosi greičio.

Realizuoti modeliui naudojama „Python“ programavimo kalba su „math“ ir „random“ paketais. Pagal studento numerį: 2110911, atliekamas 1 variantas – naudojamas stochastinis gradientinis nusileidimas ir sigmoidinis neuronas.

2 Duomenys

Praktiniam darbui atlikti naudojami „Irisų“ ir „Krūties vėžio“ duomenų rinkiniai. Abu atsiųsti iš „UC Irvine Machine Learning Repository“ duomenų saugyklos.

2.1 Irisų duomenų aibė

Šiame duomenų rinkinyje išviso yra 150 įrašų, 4 požymiai ir 3 galimos klasės reikšmės: Setosa, Versicolor ir Virginica. Tačiau darbe naudojami tik įrašai priklausantys Versicolor arba Virginica klasėms. Programoje šios klasės reikšmės pakeičiamos atitinkamai į 0 ir 1. Pašalinus Setosa reikšmes, duomenų rinkinyje lieka 100 eilučių. Be to, atlikdami tyrimą šią duomenų aibę padalinsime į mokymo ir testavimo dalis santykiu 80:20.

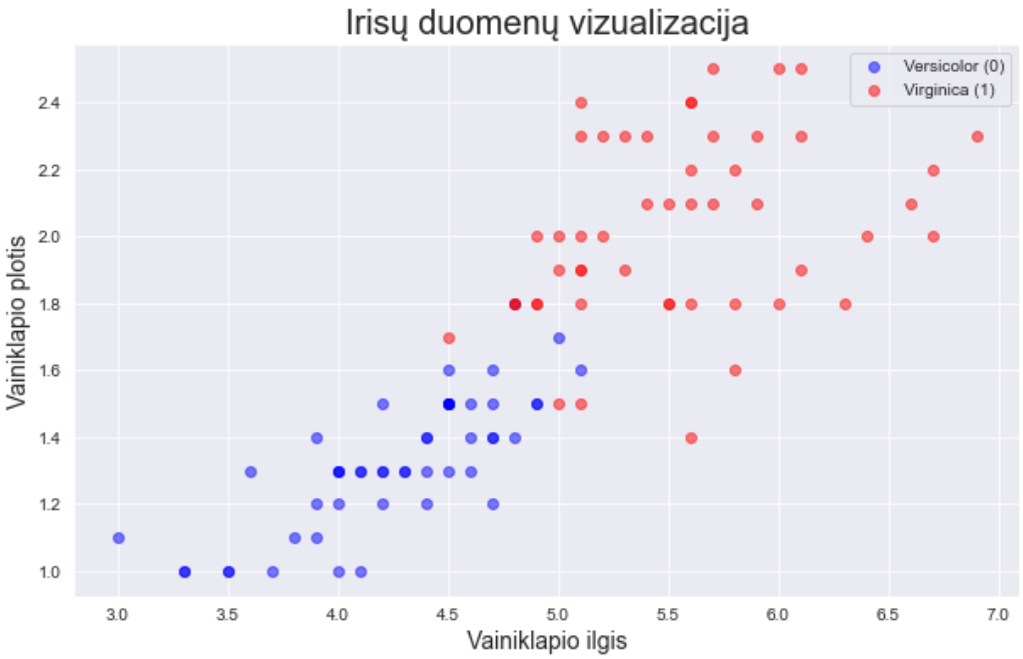
2.2 Krūties vėžio duomenų aibė

Duomenų aibė turi 699 įrašus, 9 požymius ir dvejas klasės reikšmes: 2 ir 4. Jas atitinkamai pakeičiame į 0 ir 1. Užduočiai atlikti, pašalinami įrašai turintys trūkstamų reikšmių, todėl lieka 683 eilutės. Padalijus įrašus mokymo ir testavimo aibėms santykiu 80:20, gauname 546 eilutes mokymui ir 137 – testavimui.

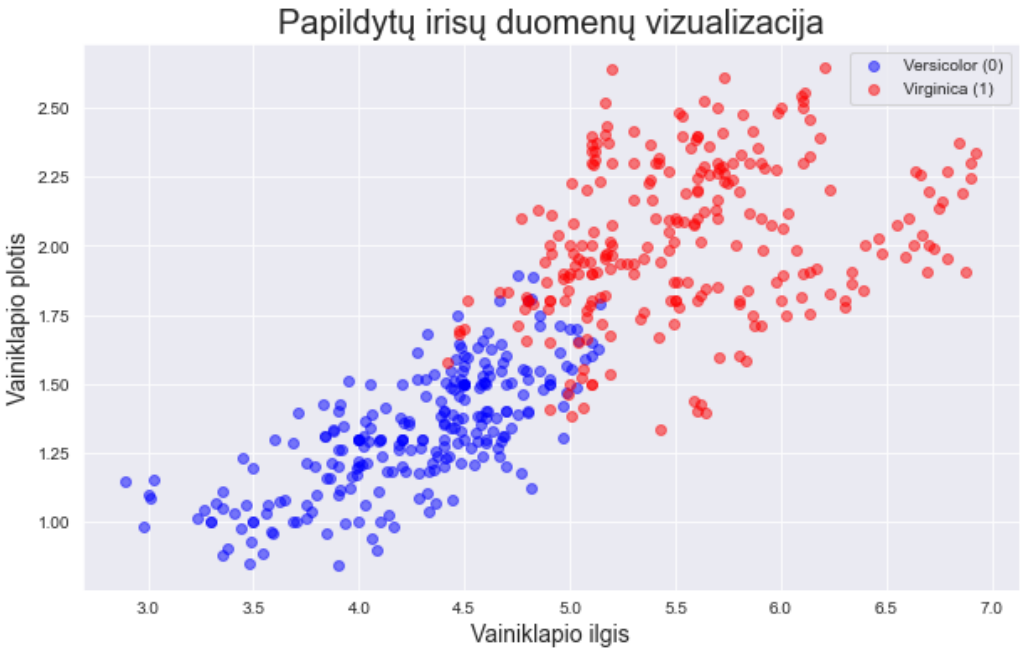
2.3 Irisų duomenų aibės didinimas

Kadangi irisų duomenų buvo nedaug (po 50 kiekvienos klasės įrašų), reikėjo padidinti kiekvienos klasės įrašų kiekį taip, kad duomenų pasiskirstymas klasėse išliktų toks pat. Duomenų kiekį didinau, pridėdama triukšmo (noise). Atsitiktinai generavau reikšmes iš normalaus skirstinio su vidurkiu 0 ir dispersija 0,08, o gautas reikšmes pridėjau prie esamų irisų duomenų. Šis procesas buvo pakartotas keturis kartus, po kurio kiekvienos klasės įrašų skaičius išaugo iki 250.

Patikrinti, ar duomenų pasiskirstymas klasėse išliko nepakitęs, pasirinkau vizualizuoti „Vainiklapio ilgį“ ir „Vainiklapio plotį“ požymius. Iš žemiau pateiktų sklaidos diagramų matome, kad duomenų pasiskirstymas klasėse išliko nepasikeitęs.



1 pav.: Pradinių duomenų vainiklapio ilgio ir pločio sklaidos diagrama



2 pav.: Papildytų duomenų vainiklapio ilgio ir pločio sklaidos diagrama

3 Praktinis darbas

3.1 Tikslas

Praktinės užduoties tikslas yra apmokyti vieną neuroną spręsti dviejų klasių uždavinį ir atlikti tyrimą su dviem duomenų aibėmis.

3.2 Uždaviniai

- Parsisiųsti ir paruošti duomenis.
- Sukurti programą, kuri įgyvendintų vieno neurono mokymo ir testavimo procesą, sprendžiant klasifikavimo uždavinį.
- Atlikti tyrimus ir vienai, ir kitai duomenų aibei („Irisų“ ir „Krūties vėžio“).
- Pateikti lenteles ir grafikus, kurie buvo gauti iš atlikto tyrimo.
- Pateikti atlikto tyrimo išvadas.

4 Perceptrono mokymas

4.1 Mokymo strategija

Duomenų aibės i-tosios eilutės požymių reikšmių vektorius žymimas: $X_i = (x_{i0}, x_{i1}, x_{i2}, \ldots, x_{in})$, kur n yra požymių skaičius. Tariama, kad $x_{i0} = 1$, $\forall i = 1, \ldots, n$, nes poslinkis b laikomas nuliniu svoriu: $b = w_0$. Taip pat turime kiekvienam duomenų įrašui žymes t_i, kurios sprendžiant klasifikavimo uždavinį atitinka klases.

Darbe naudojama sigmoidinė aktyvacijos funkcija:

$$f(a_i) = \frac{1}{1 + e^{-a_i}},$$

čia a_i yra suma, apskaičiuojama taip:

$$a_i = \sum_{k=0}^{n} w_k x_{ik}, \quad \text{kur } n \text{ yra duomenų rinkinio įrašų skaičius.}$$

Kadangi naudojama sigmoidinė aktyvacijos funkcija, neuronas vadinamas sigmoidiniu. Jį mokome taikant stochastinio gradientinio nusileidimo algoritmą. Neurono mokymo procese svoriai $W = (w_0, w_1, \ldots, w_n)$ keičiami taip, kad tinklo išėjimo reikšmė y_i, gauta įėjimą pateikus vektorių $X_i = (x_{i0}, x_{i1}, \ldots, x_{in})$, būtų kiek galima artimesnė norimai reikšmei t_i. Siekiama, kad neurono veikimo paklaida būtų kiek galima mažesnė. Ši paklaida $E(W)$ apibrėžiama taip:

$$E(W) = \sum_{k=1}^{m} (t_i - y_i)^2, \quad \text{kur } m \text{ yra mokymo duomenų rinkinio įrašų skaičius.}$$

Taikant stochastinį gradientinį nusileidimą, siekiama minimizuoti paklaidą kiekvienam duomenų įrašui atskirai. Sigmoidinio neurono mokymas taikant stochastinį gradientinį nusileidimą vykdomas taip:

- Tarkime, kad turime m duomenų įrašų: $X = (X_1, X_2, \ldots, X_m)$, čia $X_i = (x_{i0}, x_{i1}, \ldots, x_{in})$ - požymių vektorius.
- Parenkami pradiniai svoriai $W = (w_0, w_1, \ldots, w_n)$, mokymo greitis η, epochų skaičius ir paklaidos siekiamo tikslumo reikšmė E_{min}.
- Tuomet vykdomas iteracinis procesas, kuriame svoriai atnaujinami po kiekvieno duomenų įrašo apdorojimo.
  - Kiekvienam duomenų įrašui apskaičiuojamos y_i reikšmės pagal sigmoidinę funkciją:

    $$y_i = f(a_i), \quad \text{čia } f\text{- sigmoidinė funkcija.}$$

  - Apskaičiuojame kiekvieno įrašo paklaidos funkcijos gradientą:

    $$\nabla E_i(W) = (y_i - t_i) y_i (1 - y_i) x_{ik}, \quad \text{čia } k = 0, \ldots, n.$$

– Svoriai keičiami pagal šią mokymo taisyklę:

$$w_k := w_k - \eta \nabla E_i(W),$$

– Šis procesas vykdomas kiekvienam duomenų įrašui, o ne visam duomenų rinkiniui iš karto.

- Mokymo procesas tęsiamas tol, kol praėjome visas epochas arba paklaida tampa mažesnė nei E_{min}.

4.2 Pradinių svorių parinkimas

Pradinis poslinkis w_0 ir svoriai $w_1, w_2, \ldots, w_n$, kur n yra duomenų aibės požymių skaičius, buvo pasirinkti atsitiktinai generuoti $(w_0, w_1, w_2, \ldots, w_n)$ rinkinyje, kai $w_0, w_1, w_2, \ldots, w_n \sim U(-1, 1)$.

4.3 Tikslumo skaičiavimas

Klasifikavimo tikslumas (accuracy) – tai santykis tarp teisingai klasifikuotų ir visų duomenų. Naudojant sigmoidinę aktyvacijos funkciją, neurono išėjimo reikšmės yra apvalinamos iki artimiausio sveikojo skaičiaus (0 arba 1). Tada gauta reikšmė yra lyginama su duomenų įrašo klasės reikšme.

5 Rezultatai

5.1 Tikslumo ir paklaidos priklausomybė nuo epochų skaičiaus ir mokymo greičio

Tyrimas buvo atliktas analizuojant paklaidų ir tikslumo priklausomybę nuo tokių hiperparametrų kaip epochų skaičius ir mokymo greitis. Tikslumo ir paklaidos vertės pateiktos pagal gautus svorius paskutinėje mokymo epochoje.

1 lentelė: Mokymo ir testavimo tikslumas bei paklaida pagal epochas ir mokymo greitį Irisų aibei

Epochos	Mok. greitis	Mok. tikslumas	Test. tikslumas	Mok. paklaida	Test. paklaida
10	0,10	0,9400	0,9000	0,0552	0,0646
10	0,50	0,9200	0,9300	0,0571	0,0543
10	0,99	0,5125	0,4500	0,4875	0,5500
100	0,10	0,9575	0,9500	0,0379	0,0328
100	0,50	0,9550	0,9600	0,0425	0,0254
100	0,99	0,9475	0,9700	0,0487	0,0238
1000	0,10	0,9575	0,9700	0,0283	0,0320
1000	0,50	0,9475	0,9800	0,0444	0,0161
1000	0,99	0,9525	0,9500	0,0391	0,0427

2 lentelė: Mokymo ir testavimo tikslumas bei paklaida pagal epochas ir mokymo greitį Krūtų vėžio aibei

Epochos	Mok. greitis	Mok. tikslumas	Test. tikslumas	Mok. paklaida	Test. paklaida
10	0,10	0,6465	0,6642	0,3535	0,3358
10	0,50	0,6703	0,5693	0,3297	0,4307
10	0,99	0,6703	0,5693	0,3297	0,4307
100	0,10	0,9689	0,9781	0,0244	0,0259
100	0,50	0,6538	0,6350	0,3462	0,3650
100	0,99	0,6392	0,6934	0,3608	0,3066
1000	0,10	0,9744	0,9854	0,0229	0,0134
1000	0,50	0,6502	0,6496	0,3498	0,3504
1000	0,99	0,6355	0,7080	0,3645	0,2920

Pagal gautus rezultatus, irisų aibėje didesnis epochų skaičius padidino tiek mokymo, tiek testavimo tikslumą. Pavyzdžiui, kai epochų buvo 1000 ir mokymo greitis (alpha) buvo 0,10, mokymo tikslumas siekė 97,44%, o testavimo tikslumas – 98,54%. Tačiau didėjant mokymo greičiui, tikslumas mažėjo, ypač kai alpha siekė 0,99.

Krūtų vėžio aibėje didesnis epochų skaičius taip pat pagerino rezultatus. Su 1000 epochų ir mažu mokymo greičiu (alpha = 0,10), mokymo tikslumas pasiekė 97,44%, o testavimo tikslumas buvo 96,35%. Tačiau esant didelei alpha reikšmei (0,99), net su 1000 epochų mokymo tikslumas nukrito iki 64,47%.

Apibendrinant, didesnis epochų skaičius gerina tikslumą, o per didelis mokymo greitis mažina modelio efektyvumą.

5.2 Apmokytas neuronas

5.2.1 Irisų aibė

Tolimesnei analizei pasirinkome mokymo greitį $\eta = 0,5$ ir 1000 epochų skaičių. Iš 1 lentelės matome, kad klasifikavimo tikslumas tarp testavimo ir mokymo duomenų aibių skiriasi 3,25%. Irisų duomenų aibėje apmokyto neurono **svoris**:

$$w_0 = -37,535, \quad w_1 = -10,145, \quad w_2 = -12,457, \quad w_3 = 19,920, \quad w_4 = 23,521.$$

Naudojant šiuos svorius, paklaida, gauta paskutinėje epochoje mokymo duomenims yra 0,0415, o klasifikavimo tikslumas = 94,75%. Paklaida testavimo duomenų aibei = 0,0178, o klasifikavimo tikslumas = 98%.

Klasifikuojant visą duomenų rinkinį su išsirinktu svoriu pasiektas 97% tikslumas.

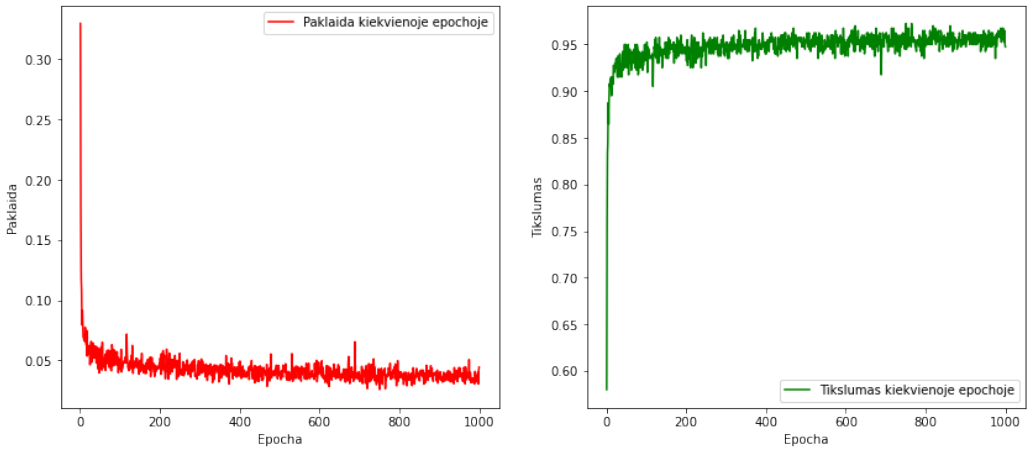
Naudojant jau išmokytą neuroną, palyginamos klasės, nustatytos neurono, ir tos, kurios turėjo būti testavimo duomenų aibėje. Rezultatai pateikti 3 lentelėje. Kaip matome, neuronas neteisingai suklasifikavo tik 2 įrašus.

3 lentelė: Priskirtos ir tikros klasės

(y, t)	**(y, t)**	**(y, t)**	**(y, t)**	**(y, t)**	**(y, t)**	**(y, t)**	**(y, t)**	**(y, t)**	**(y, t)**
(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(0, 0)	(1, 1)	(0, 0)	(1, 1)	(0, 0)
(1, 1)	(1, 1)	(0, 0)	(1, 1)	(1, 1)	(1, 1)	(0, 0)	(0, 0)	(0, 0)	(0, 0)
(1, 1)	(1, 1)	(1, 1)	(0, 0)	(0, 0)	(0, 0)	(1, 1)	(1, 1)	(0, 0)	(0, 0)
(0, 0)	(0, 0)	(1, 1)	(0, 0)	(1, 1)	(0, 0)	(1, 1)	(1, 1)	(1, 1)	(0, 0)
(1, 1)	(1, 1)	(1, 1)	(0, 0)	(1, 1)	(0, 0)	(1, 1)	(1, 1)	(0, 0)	(1, 1)
(1, 1)	(1, 1)	(1, 1)	(0, 0)	(0, 0)	(1, 1)	(1, 1)	(1, 1)	**(1, 0)**	(0, 0)
(0, 0)	(0, 0)	(0, 0)	(1, 1)	(1, 1)	(0, 0)	(0, 0)	(0, 0)	(1, 1)	(1, 1)
(0, 0)	(1, 1)	(0, 0)	(1, 1)	(0, 0)	(1, 1)	(1, 1)	(0, 0)	(1, 1)	(1, 1)
(1, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(1, 1)	(1, 1)	(1, 1)	(0, 0)
(0, 0)	(1, 1)	(1, 1)	(1, 1)	(0, 0)	**(0, 1)**	(0, 0)	(0, 0)	(1, 1)	(1, 1)

5.2.2 Paklaidos pokytis ir klasifikavimo tikslumas pagal epochą

Analizuojant paklaidos pokytį pagal epochos numerį, 3 paveiksle matome, kad paklaida nuolat mažėja, tačiau pasiekus apie 250-tąją epochą, pokyčiai tarp sekančių epochų tapo mažesni ir paklaida artėja prie tam tikros reikšmės, nors svyravimai išlieka. Šie svyravimai gali rodyti modelio persimokymą. Klasifikavimo tikslumas taip pat didėja, artėdamas link 1, kartu su epochų skaičiaus didėjimu.



3 pav.: Mokymo aibės tikslumo ir paklaidos pokytis per 1000 epochų su 0,5 mokymo greičiu

5.2.3 Krūtų vėžio aibė

Tolimesnei analizei pasirinkau mokymo greitį $\eta = 0,1$ ir 1000 epochų skaičių. Klasifikavimo tikslumas tarp testavimo ir mokymo duomenų aibių skiriasi beveik 3 procentais. Irisų duomenų aibėje apmokyto neurono **svoris**:

$$w_0 = -15,878, \quad w_1 = 0,678, \quad w_2 = 0,530, \quad w_3 = 0,940, \quad w_4 = 0,384,$$

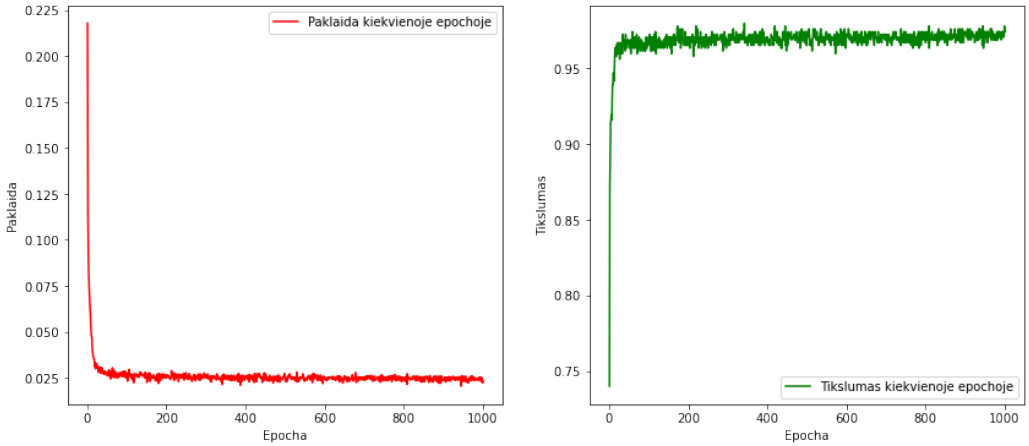
Naudojant šiuos svorius, paklaida, gauta paskutinėje epochoje mokymo duomenims yra 0,0229, o klasifikavimo tikslumas = 97,44%. Paklaidą testavimo duomenų aibei = 0,0134, o klasifikavimo tikslumas = 98,54%.

Naudojant jau išmokytą neuroną, palyginamos klasės, nustatytos neurono, ir tos, kurios turėjo būti testavimo duomenų aibėje. Klasifikavimo tikslumas siekia 98,54%, tai rodo, kad neuronas neteisingai suklasifikavo 2 įrašus.

Klasifikuojant visą duomenų rinkinį su išsirinktu svoriu pasiektas 97,95% tikslumas.

5.2.4 Paklaidos pokytis ir klasifikavimo tikslumas pagal epochą

Analizuojant paklaidos pokytį pagal epochos numerį, 4 paveiksle matome, kad paklaida staigiai nukrenta ir per maždaug 100 epochų konverguoja į konkrečią reikšmę. Modelio tikslumas irgi staigiai pagerėja ir išlieka aukštas, nors ir su svyravimais ir išsišokimais.



4 pav.: Mokymo aibės tikslumo ir paklaidos pokytis per 1000 epochų su 0,1 mokymo greičiu

6 Išvados

Užduoties metu buvo sėkmingai sukurtas sigmoidinis dirbtinis neuronas su stochastiniu gradientu. Neuronas buvo apmokytas ir validuotas naudojant irisų ir krūtų vėžio duomenų rinkinius. Eksperimentai parodė, kad didesnis epochų skaičius pagerina mokymo ir testavimo tikslumą, tačiau per didelis mokymosi greitis gali pabloginti rezultatus, ypač esant mažam epochų skaičiui.

Literatūra

[1] Iris dataset, UCI Machine Learning Repository, https://archive.ics.uci.edu/dataset/53/iris.

[2] Breast Cancer Wisconsin (Original) dataset, UCI Machine Learning Repository, https://archive.ics.uci.edu/dataset/15/breast+cancer+wisconsin+original.

[3] Prof. Dr. Olga Kurasova *"Dirbtinis neuronas - perceptronas"* paskaitų skaidrės.

[4] https://datagy.io/pandas-replace-values/

[5] https://machinelearningmastery.com/train-neural-networks-with-noise-to-reduce-overfitting/

[6] https:https://stackoverflow.com/questions/46093073/adding-gaussian-noise-to-a-dataset-of-floating-points-and-save-it-python

7 Programos kodas su komentarais

Kodas rašytas Python programine kalba.

7.1 Duomenų aibės didinimas

```
# Atsiskiriame požymius ir tikslus
X = iris.iloc[:, :4]
y = iris.iloc[:, -1]

mod_iris = pd.DataFrame(X)
mod_label = pd.DataFrame(y)

# 4 kartus generuojame modifikuotus (su pridėtu triukšmu) požymius
for _ in range (4):
    mod = np.random.normal(0, 0.08, [100, 4])
    mod_noise = X.iloc[:, :4] + mod
    mod_iris = pd.concat([mod_iris, mod_noise], axis=0)

# Tas pačias klases perrašom 4 kartus
mod_label = pd.concat([mod_label] * 5)

# Sudedam padidintus duomenis
iris_new = pd.concat([mod_iris.reset_index(drop=True), mod_label.reset_index(drop=True)], axis=1)

iris_new.to_csv('iris.csv', sep=',', index=False, header=True)
iris_new
```

7.2 Pagalbinės funkcijos

```
# Funkcija duomenų nuskaitymui ir paruošimui

def nuskaityti_duomenis(failas):
    # Nuskaitome duomenų rinkinį iš failo ir sumaišome
    data = pd.read_csv(failas).sample(frac=1).reset_index(drop=True)

    # Atskiriame požymius (X) ir žymes (t)
    X = data.iloc[:, :-1]
    t = data.iloc[:, -1]

    # Pridedame poslinkio (bias) stulpelį

    bias = pd.DataFrame(np.ones(X.shape[0]), columns=['Bias'])
    X.reset_index(drop=True, inplace=True)
    bias.reset_index(drop=True, inplace=True)
    t.reset_index(drop=True, inplace=True)
```

```
    # Sujungiame požymius su poslinkio stulpeliu
    X = pd.concat([bias, X], axis=1)

    return X, t


# Sigmoidinės funkcijos apibrėžimas
def sigmoidine(z):
    return 1 / (1 + np.exp(-z))

# Funkcija klasifikavimui
def klasifikavimas(f):
    return 1 if f >= 0.5 else 0

# Funkcija svoriams atnaujinti
def svorio_atnaujinimas(w, alpha, xi, ti, f):
    return w - alpha * (f - ti) * f * (1 - f) * xi

# Funkcija klaidai skaičiuoti
def skaiciuoti_klaida(predicted, ti):
    return (predicted - ti) ** 2
```

7.3 Neurono mokymas

```
def mokymas(failas, alpha, epochos, min_error, test_size):
    X, t = nuskaityti_duomenis(failas)

    # Padalijame duomenis į mokymo ir testavimo aibes
    X_train, X_test, t_train, t_test = train_test_split(X, t, test_size=test_size, random_state=42)

    # Inicijuojame svorius
    w = np.random.uniform(-1, 1, X_train.shape[1])

    epocha = 0
    mok_klaidos = []
    mok_tikslumai = []
    test_klaidos = []
    test_tikslumai = []

    while epocha < epochos:
        mok_klaida = 0
        klasifikacijos_mok = []

        # Atsitiktinai sumaišome duomenis
        data_shuffled = pd.concat([X_train, t_train], axis=1).sample(frac=1).reset_index(drop=True)

        # Atliekame mokymą naudojant SGD
```

```
    for i in range(X_train.shape[0]):
        xi = data_shuffled.iloc[i, :-1].values
        ti = data_shuffled.iloc[i, -1]

        # Skaičiuojame sandaugą ir pritaikome sigmoidinę aktyvacijos funkciją
        z = np.dot(xi, w.T)
        f = sigmoidine(z)

        # Klasifikuojame
        predicted = klasifikavimas(f)
        klasifikacijos_mok.append(predicted)

        # Atnaujiname svorius
        w = svorio_atnaujinimas(w, alpha, xi, ti, f)

        # Skaičiuojame kvadratinę klaidą
        mok_klaida += skaiciuoti_klaida(f, ti)

    epocha += 1

    # Skaičiuojame ir išsaugome mokymo klasifikavimo tikslumą
    mok_accuracy = accuracy_score(data_shuffled.iloc[:, -1], klasifikacijos_mok)
    mok_tikslumai.append(mok_accuracy)
    mok_klaidos.append(mok_klaida / X_train.shape[0])

    # Baigiame mokymą, jei klaida tampa mažesnė už nustatytą ribą
    if 1 - mok_accuracy <= min_error:
        break

# Testavimo fazė vykdoma po visų epochų arba pasiekus minimalią klaidą
klasifikacijos_test = []
test_klaida = 0
results = []

for i in range(X_test.shape[0]):
    xi_test = X_test.iloc[i, :].values

    # Skaičiuojame sandaugą ir pritaikome sigmoidinę aktyvacijos funkciją
    z_test = np.dot(xi_test, w.T)
    f_test = sigmoidine(z_test)

    # Klasifikuojame
    predicted_test = klasifikavimas(f_test)
    klasifikacijos_test.append(predicted_test)

    # Išsaugome priskirtas ir tikrąsias klases
    results.append((predicted_test, t_test.iloc[i]))
```

```
        # Skaičiuojame kvadratinę klaidą
        test_klaida += skaiciuoti_klaida(t_test.iloc[i], f_test)

    test_accuracy = accuracy_score(t_test, klasifikacijos_test)
    test_tikslumai.append(test_accuracy)
    test_klaidos.append(test_klaida / X_test.shape[0])

    # Grąžinti svorius, klaidų ir tikslumo rezultatus
    return w, mok_klaidos, mok_tikslumai, test_klaidos, test_tikslumai, results
```

7.4 Neurono mokymas ant irisų aibės

```
# Paleidžiame mokymą ir testavimą su Irisų duomenimis

file = 'iris.csv'

#Hiperparametrai
alphos = [0.1, 0.5, 0.99]
epochos = [10, 100, 1000]
min_error = 0.01
test_size = 0.2

w_irisu_alpha = []
mok_klaidos_alpha = []
mok_tikslumai_alpha = []
test_klaidos_alpha = []
test_tikslumai_alpha = []
test_results = []

for alpha in alphos:
    for epocha in epochos:
        w_irisu, mok_klaidos, mok_tikslumai, test_klaidos, test_tikslumai, test_result = mokymas(file

        w_irisu_alpha.append(w_irisu)
        mok_klaidos_alpha.append(mok_klaidos)
        mok_tikslumai_alpha.append(mok_tikslumai)
        test_klaidos_alpha.append(test_klaidos)
        test_tikslumai_alpha.append(test_tikslumai)
        test_results.append(test_result)
```

7.5 Neurono mokymas ant krūtų vėžio aibės

```
# Paleidžiame mokymą ir testavimą su krutų vėžio duomenimis

file = 'breast_cancer.csv'

#Hiperparametrai
```

```
alphos = [0.1, 0.5, 0.99]
epochos = [10, 100, 1000]
min_error = 0.01
test_size = 0.2

w_cancer_alpha = []
c_mok_klaidos_alpha = []
c_mok_tikslumai_alpha = []
c_test_klaidos_alpha = []
c_test_tikslumai_alpha = []
c_test_results = []

for alpha in alphos:
    for epocha in epochos:
        w_cancer, mok_klaidos, mok_tikslumai, test_klaidos, test_tikslumai, c_test_result = mokymas(f

        w_cancer_alpha.append(w_cancer)
        c_mok_klaidos_alpha.append(mok_klaidos)
        c_mok_tikslumai_alpha.append(mok_tikslumai)
        c_test_klaidos_alpha.append(test_klaidos)
        c_test_tikslumai_alpha.append(test_tikslumai)
        c_test_results.append(c_test_result)
```

7.6 Tikrinimo funkcija

```
def tikrinimas(file, w):
    X, y = nuskaityti_duomenis(file)
    teisingi = 0
    for i in range(X.shape[0]):
        z = np.dot(X.iloc[i, :].values, w.T)
        f = sigmoidine(z)
        predicted = klasifikavimas(f)
        if predicted == y[i]:
            teisingi += 1
    accuracy = teisingi/X.shape[0]*100
    return teisingi, accuracy
```