VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

STUDIJŲ PROGRAMA DUOMENŲ MOKSLAS 4 KURSAS

Dirbtinis Neuronas

Artificial Neuron

Darbo aprašas

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

Vilnius
2024

Turinys

1 Įvadas

Užduoties tikslas yra išanalizuoti dirbtinio neurono modelio veikimo principus ir realizuoti modelį programiškai naudojant dvi skirtingas aktyvacijos funkcijas. Taip pat, aprašyti nelygybių sistemą, kurią reikia išspręsti, norint teisingai parinkti svorių ir poslinkio reikšmes, kai aktyvacijos funkcija yra slenkstinė. Nelygybių sistemą iliustruoti grafiniu būdu ir patikrinti ar grafiniu būdu gauti sprendiniai yra nelygybių sistemos sprendiniai. Realizuotam modeliui naudojama „Python“ programavimo kalba.

2 Dirbtinio neurono modelio analizė

2.1 Tikslas

Užduoties tikslas - sukurti dirbtinį neuroną, teisingai klasifikuojantį duomenis.

2.2 Užduotys

1. Sukurti dirbtinio neurono modelį, kuris priimtų įėjimo reikšmes (x_1, x_2) ir teisingai priskirtų klasę, naudodamas slenkstinę ir sigmoidinę aktyvacijos funkcijas.
2. Rasti svorius (w_1, w_2) ir poslinkį (b) dviem būdais:
   - pasirinktame intervale tam tikru žingsniu perrinkti svorių ir poslinkio reikšmes.
   - atsitiktinai generuoti svorių ir poslinkių reikšmes.
3. Užrašyti ir išspręsti nelygybių sistemą, gauti teisingų svorių ir poslinkių aibę, kai aktyvacijos funkcija yra slenkstinė.
4. Nubraižyti grafikus, siekiant vizualizuoti nelygybių sistemą ir gauti sprendinius, juos patikrinti įstatant į nelygybes.

2.3 Duomenys klasifikavimui

1 lentelė: Duomenys klasifikavimui

Duomenys		**Klasė**
x_1	x_2	t
-0,2	0,5	0
0,2	-0,7	0
0,8	-0,8	1
0,8	1	1

2.4 Formulės

1. Įėjimo reikšmių ir svorių sandaugų sumos formulė:

$$a = \sum_{k=0}^{n} w_k x_k.$$

2. Slenkstinė aktyvacijos funkcija:

$$f(a) = \begin{cases} 1, & \text{jei } a \geq 0, \\ 0, & \text{jei } a < 0. \end{cases}$$

3. Sigmoidinė aktyvacijos funkcija:

$$f(a) = \begin{cases} 1, & \text{jei } \frac{1}{1+e^{-a}} \geq 0.5, \\ 0, & \text{kitu atveju.} \end{cases}$$

2.5 Svorių ir poslinkio radimo strategijos

1. Svorių ir poslinkio reikšmės w_1, w_2 ir b buvo ieškomos intervale $[-2; 2]$ su žingsniu 0,2, siekiant rasti tinkamą kombinaciją. Surandamos visos tinkamos kombinacijos, o iš jų atsitiktinai parenkami 5 svoriai kaip pavyzdžiai.

2. Reikšmės buvo ieškomos atsitiktinai generuojant svorius w_1, w_2 ir poslinkį b iš intervalo $[-2; 2]$. Funkcijoje buvo galima nurodyti, kiek tinkamų svorių reikia surasti, o pasiekus reikiamą kiekį, funkcija sustabdoma.

2.6 Rezultatų pavyzdžiai

2.6.1 Slenkstinės aktyvacijos funkcijos pritaikymas

Slenkstinė funkcija, klasifikuoja duomenis, patikrinant, ar svorių vektoriaus (įtraukus poslinkį b) ir duomenų taškų sandauga yra didesnė už 0. Jei taip, priskiriama klasė 1, kitu atveju – klasė 0.

```
def slenkstine(duomenys, svoris):
    klase_slen = []

    for row in duomenys:
        if row @ svoris.T >= 0:
            klase_slen.append(1)
        else:
            klase_slen.append(0)

    return klase_slen
```

Svorių ir poslinkių pavyzdžiai:

1) Iš intervalo $[-2; 2]$ su žingsniu 0,2:

```
Naudojant svorius: [-1,  1,8  0,4], gauta klasė: [0, 0, 1, 1].
Naudojant svorius: [-0,6  1,8  0,8], gauta klasė: [0, 0, 1, 1].
Naudojant svorius: [-1,  1,8 -0], gauta klasė: [0, 0, 1, 1].
Naudojant svorius: [-0,4  0,8 -0,2], gauta klasė: [0, 0, 1, 1].
Naudojant svorius: [-0,4  1,2 -0,2], gauta klasė: [0, 0, 1, 1].
```

2) Iš intervalo $[-2; 2]$ generuojant atsitiktinai:

```
Naudojant svorius: [-0,2  0,8  0,4], gauta klasė: [0, 0, 1, 1]
Naudojant svorius: [-0,6  1,8 -0], gauta klasė: [0, 0, 1, 1]
Naudojant svorius: [-0,2  1,6  0,4], gauta klasė: [0, 0, 1, 1]
Naudojant svorius: [-0,4  0,8 -0], gauta klasė: [0, 0, 1, 1]
Naudojant svorius: [-0,8  1,4  0,2], gauta klasė: [0, 0, 1, 1]
```

2.6.2 Sigmoidinės aktyvacijos funkcijos pritaikymas

Antrasis metodas remiasi sigmoidine funkcija, kuri pirmiausia apskaičiuoja a vertę, t.y svorių vektoriaus (įtraukus poslinkį b) ir duomenų taškų sandaugą, o tada taiko sigmoidinę funkciją $f(a)$. Jei gauta $f(a)$ reikšmė ≥ 0.5 tai priskiriama 1 klasė, kitu atveju 0.

```
def sigmoidine(duomenys, svoris):
    klase_sigm = []

    for row in duomenys:
        a = row @ svoris.T
        f = 1 / (1 + np.exp(-a))
        if f >= 0.5:
            klase_sigm.append(1)
        else:
            klase_sigm.append(0)

    return klase_sigm
```

Svorių ir poslinkių pavyzdžiai:

1) Iš intervalo $[-2; 2]$ su žingsniu 0,2:

```
Naudojant svorius: [-0,6  1,2 -0], gauta klasė: [0, 0, 1, 1].
Naudojant svorius: [-0,6  1,6 -0], gauta klasė: [0, 0, 1, 1].
Naudojant svorius: [-0,2  1,8  0,4], gauta klasė: [0, 0, 1, 1].
Naudojant svorius: [-0,8  1,8 -0,4], gauta klasė: [0, 0, 1, 1].
Naudojant svorius: [-0,6  1,8 -0,2], gauta klasė: [0, 0, 1, 1].
```

2) Iš intervalo $[-2; 2]$ generuojant atsitiktinai:

```
Naudojant svorius: [-0,8  1,6 -0,2], gauta klasė: [0, 0, 1, 1].
Naudojant svorius: [-0,2  1,8  0,8], gauta klasė: [0, 0, 1, 1].
Naudojant svorius: [-0,4  1,2  0,2], gauta klasė: [0, 0, 1, 1].
Naudojant svorius: [-0,6  0,8 -0], gauta klasė: [0, 0, 1, 1].
Naudojant svorius: [-0,6  1,8 -0,2], gauta klasė: [0, 0, 1, 1].
```

Matome, kad abiejais būdais buvo rasti tinkami svoriai, teisingai klasifikuojantys turimus duomenis. Tačiau skiriasi radimo būdų efektyvuymas. Iki pirmo tinkamo svorio suradimo pirmu būdu, abiejom aktyvacijos funkcijomis užtruko 5010 bandymų, o atsitiktinai generuojant skaičius prireikė daug mažiau bandymų, pavyzdžiui visiem 5-iems tinkamiems svoriams surasti prireikė tik 295 bandymų.

3 Nelygybių sistema poslinkio ir svorių radimui

3.1 Nelygybių sistemos gavimas

Tarkime, kad reikia gauti teisingų poslinkių ir svorių trejetų (b, w_1, w_2) aibę. Pasirinkus slenkstinę aktyvacijos funkciją, išvestis gaunama taip:

$$y = f(a) = \begin{cases} 1, & \text{jei } a \geq 0 \\ 0, & \text{kitu atveju.} \end{cases}$$

Gauname tokią nelygybių sistemą:

$$\begin{cases} -0.2w_1 + 0.5w_2 + b & < 0, \\ 0.2w_1 - 0.7w_2 + b & < 0, \\ 0.8w_1 - 0.7w_2 + b & \geq 0, \\ 0.8w_1 + w_2 + b & \geq 0. \end{cases}$$

3.2 Nelygybių sistemos sprendimas grafiniu būdu

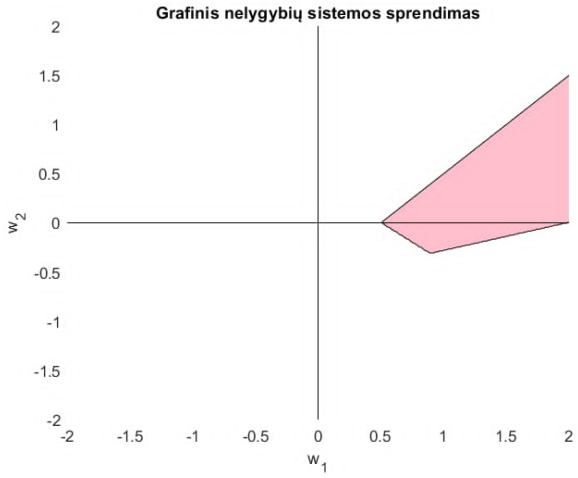
Nelygybių sistemą galima spręsti fiksuojant poslinkį b kaip konstantą ir grafiniu būdu randant svorių w_1 ir w_2 reikšmes. Spręsdami nelygybių sistemą grafiniu būdu gauname užspalvintą figūrą, kuriai priklausantis taškas atitinka tinkamą svorių w_1 ir w_2 porą.

3.2.1 Pavyzdys

Tegu $b = -0.4$, tada turime tokią nelygybių sistemą:

$$\begin{cases} -0.2w_1 + 0.5w_2 - 0.4 & < 0, \\ 0.2w_1 - 0.7w_2 - 0.4 & < 0, \\ 0.8w_1 - 0.7w_2 - 0.4 & \geq 0, \\ 0.8w_1 + w_2 - 0.4 & \geq 0. \end{cases}$$

naudodami „MATLAB“ programą gauname tokį grafiką:



1 pav.: Nelygybių sistemos grafinis sprendimas

Iš grafiko galime parinkti $w_1 = 1$ ir $w_2 = 0.4$. Matome, jog šis taškas priklauso aibei, be to:

$$\begin{cases} -0.2 \cdot 1 + 0.5 \cdot 0.4 - 0.4 & < 0, \\ 0.2 \cdot 1 - 0.7 \cdot 0.4 - 0.4 & < 0, \\ 0.8 \cdot 1 - 0.7 \cdot 0.4 - 0.4 & \geq 0, \\ 0.8 \cdot 1 + 0.4 - 0.4 & \geq 0. \end{cases}$$

Taigi galutinis sprendimas yra:

$$\begin{cases} -0.4 < 0 \\ -0.48 < 0 \\ 0.12 \geq 0 \\ 0.8 \geq 0 \end{cases}$$

Patikrinus gautus sprendinius, galime teigti, kad visos nelygybės yra tenkinamos pasirinkus tašką, esantį užspalvintoje figūroje. Taigi užspalvinta figūra, gauta grafiniu būdu, atspindi sprendinių aibę, tenkinančią nelygybių sistemą.

4 Išvados

Atliekant praktinę užduotį buvo išmoktas dirbtinio neurono modelio veikimo principas ir jo implementacija Pyhton programavimo kalba. Taip pat sužinotos dvi aktyvacijos funkcijos ir duomenų klasifikacijos procesas. Nelygybių sistemos sudarymas leido geriau suprasti, kaip galima rasti svorių ir poslinkių reikšmes. Išbandžius abu metodus, skirtus tinkamų svorių ir poslinkių nustatymui, tapo aišku, kokie yra jų privalumai ir trūkumai.

5 Programos kodas su komentarais

Kodas rašytas Python programine kalba.

5.1 Funkcija svorių radimui iš intervalo su žingsniu

Funkcija `klasifikacija` skirta svorių radimui iš nurodyto intervalo, taikant pasirinktą aktyvacijos funkciją. Ji iteruoja per galimas svorių reikšmes w_1, w_2 ir poslinkį b, klasifikuodama duomenis ir tikrindama, ar gautos klasės atitinka tikras klases. Funkcija grąžina visus svorius ir poslinkį, kurie tinkamai klasifikuoja duomenis.

```
def klasifikacija(intervalas, duomenys, tikra_klase, aktyvacijos_f):
    tinkami_svoriai = []
    bandymai = 0
    bandymai_pirmas_tinkamas = None

    for w1 in intervalas:
        for w2 in intervalas:
            for b in intervalas:
                svoris = np.array([b, w1, w2])

                # Klasifikuojame naudodami pasirinktą aktyvacijos funkciją
                klases = aktyvacijos_f(duomenys, svoris)

                bandymai += 1

                # Tikriname, ar gautos klasės atitinka tikras klases
                if np.array_equal(klases, tikra_klase.flatten()):
                    tinkami_svoriai.append(svoris)
                    print(f"Poslinkis b: {svoris[0]}, Svoriai w1: {svoris[1]}, w2: {svoris[2]}")

                    # Jei tai pirmas tinkamas svoris, fiksuojame bandymų skaičių
                    if bandymai_pirmas_tinkamas is None:
                        bandymai_pirmas_tinkamas = bandymai

    return np.array(tinkami_svoriai), bandymai_pirmas_tinkamas
```

5.2 Funkcija atsitiktiniam svorių radimui iš intervalo

Funkcija `random_svoriai_klasifikacija` yra sukurta atsitiktinių svorių paieškai, kurie tinkamai klasifikuoja duomenis, naudojant pasirinktą aktyvacijos funkciją. Šioje funkcijoje galima nurodyti norimą tinkamų svorių skaičių kaip parametrą, o funkcija tol atsitiktinai generuos svorius, kol ras reikiamą kiekį. Kiekvienoje iteracijoje atsitiktinai parenkami svoriai `w1` ir `w2` bei poslinkis `b`, kurie yra imami iš intervalo $[-2; 2]$.

```
def random_svoriai_klasifikacija(intervalas, duomenys, tikra_klase, aktyvacijos_f, svoriu_sk):
    tinkami_svoriai = []
    bandymai = 0
    while len(tinkami_svoriai) < svoriu_sk:
        # Atsitiktinai pasirenkame svorius iš nustatyto intervalo
```

```
        svoris = np.array([random.choice(intervalas),
                           random.choice(intervalas),
                           random.choice(intervalas)])

        # Klasifikuojame naudodami pasirinktą aktyvacijos funkciją
        klases = aktyvacijos_f(duomenys, svoris)
        bandymai += 1
        # Tikriname, ar gautos klasės atitinka tikras klases
        if np.array_equal(klases, tikra_klase):
            tinkami_svoriai.append(svoris)
            print(f"Poslinkis b: {svoris[0]}, Svoriai w1: {svoris[1]}, w2: {svoris[2]}")

    return np.array(tinkami_svoriai), bandymai
```

Literatūra

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

[2] Overleaf Documentation, https://www.overleaf.com/learn.

[3] https://stackoverflow.com/questions/6127524/python-how-to-know-the-index-when-you-randomly-select-an-element-from-a-sequenc