VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

Saviorganizuojantys neuroniniai tinklai

Laboratorinis darbas

Atliko: 4 kurso 1 grupės studentas
Paulius Minajevas (SN: 2110599)

Vilnius – 2024

Turinys

Įvadas

Užduoties tikslas – suprogramuoti saviroganizuojančio neuroninio tinklo (žemėlapio, SOM) mokymo algoritmą, apmokyti jį naudojant pasirinktus duomenis.

1. Klasifikavimo duomenys ir metodika

1.1. Klasifikavimo duomenys

Šio tyrimo metu bus naudojamas Irisų duomenų rinkinys, kuris skirtas atpažinti gėlės rūšį pagal gėlės atributus. Visas rinkinys sudarytas iš 150 eilučių. Kiekvienai klasei priskirtas indeksas, kuris žemiau parašytas prie klasės pavadinimo.

1.1.1. Irisų duomenų rinkinys

Irisų duomenų rinkinys susidaro iš atributų:

1. „sepal length" [taurėlapio ilgis]
2. „sepal width" [taurėlapio plotis]
3. „petal length" [žiedlapio ilgis]
4. „petal width" [žiedlapio plotis]
5. „class" [klasė]

, iš kurių, šiame duomenų rinkinyje, galima atpažinti tris gėlių rūšis:

1. Iris Setosa [0]
2. Iris Versicolour [1]
3. Iris Virginica [2]

1.2. Metodika

Savarankiško organizavimo žemėlapiai (angl. Self-Organizing Maps, SOM) yra neuroninių tinklų modelis, skirtas duomenų vizualizavimui ir klasterizavimui. SOM mokymo procesas prasideda tinklo inicializacija, kurio metu sukuriamas dvimatis neuronų tinklas, kurio kiekvienas neuronas turi svorių vektorių, atsitiktinai inicializuotą arba pritaikytą pagal duomenų sritį. Svarbu užtikrinti, kad įvesties duomenys būtų normalizuoti, kad visi bruožai turėtų vienodą įtaką. Apmokymo metu iteracijų cikle vykdomi keli žingsniai. Pirmiausia, atsitiktinai parenkamas įvesties vektorius, tuomet randamas geriausiai atitinkantis vienetas (BMU), kurio svorių vektorius yra mažiausio atstumo nuo įvesties vektoriaus. BMU ir jo kaimynystėje esantys neuronai atnaujinami, kad artėtų prie įvesties vektoriaus, naudojant formulę, kur mokymosi greitis ir kaimynystės spindulys palaipsniui mažinami, siekiant užtikrinti stabilų tinklo konvergavimą. Procesas tęsiasi tol, kol tinklas stabilizuojasi arba pasiekiamas maksimalus iteracijų skaičius. Apmokytas SOM tinklas naudojamas duomenų klasterizavimui ir vizualizavimui, leidžiant efektyviai analizuoti kompleksines duomenų struktūras. Žemiau poskyriuose pateikiamos realizacijos detalės su komentarais.

1.2.1. Mokymo ciklas

```
def get_neighboor_hierarchy(wx1,wy1,x2,y2):
    distX = wx1 - x2 if wx1 > x2 else x2 - wx1
    distY = wy1 - y2 if wy1 > y2 else y2 - wy1
    return distX if distX > distY else distY

def SOM_training(data, neurons, epochs, row_num, col_num):
    for epoch in range(epochs):
      for entry in data:
        distances = np.zeros((row_num, col_num))
        for row in range(row_num):
          for col in range(col_num):
            distances[row,col] = np.sqrt(np.sum((neurons[row,col]
                - entry) ** 2))

        neuron_winner = np.unravel_index(np.argmin(distances),
            distances.shape)

        for row in range(row_num):
          for col in range(col_num):
            distWin = np.sqrt((row - neuron_winner[0]) ** 2 + (
                col - neuron_winner[1]) ** 2)
            hier = get_neighboor_hierarchy(neuron_winner[0],
                neuron_winner[1], row, col)
            if hier != 0:
              gausH = (1 - epoch/epochs) * np.exp(-distWin ** 2 /
                  (2 * get_neighboor_hierarchy(neuron_winner[0],
                  neuron_winner[1], row, col) ** 2))
              neurons[row, col] = neurons[row, col] + gausH * (
                  entry - neurons[row, col])

    return neurons
```

Šis kodas realizuoja savarankiško organizavimo žemėlapių (SOM) mokymo algoritmą. Jį sudaro du pagrindiniai komponentai:

1. get_neighboor_hierarchy funkcija: Ji apskaičiuoja duotojo neurono hierarchinį atstumą nuo nugalėtojo neurono (winner neuron). Atstumas yra lygus didžiajam skirtumui tarp eilučių (X) ir stulpelių (Y) koordinačių, naudojant Manhattan tipo logiką.
2. SOM_training funkcija: Tai pagrindinė mokymo funkcija. Ji iteruoja per duomenų rinkinį nurodytą skaičių epochų. Kiekvienam duomenų įrašui apskaičiuojami euklidiniai

atstumai tarp įrašo ir kiekvieno tinklo neurono svorių. Tada nustatomas nugalėtojas – tai neuronas, kurio svoriai yra arčiausiai duomenų įrašo. Visi tinklo neuronai atnaujinami atsižvelgiant į jų atstumą nuo nugalėtojo. Tam naudojama kaimynystės funkcija (Gaussian neighborhood), kuri mažėja su didėjančiu atstumu ir epochų skaičiumi. Kuo neuronai yra arčiau nugalėtojo, tuo labiau jie prisitaiko prie įvesties įrašo. Grąžinami galutiniai neuronų svoriai, atnaujinti pagal SOM mokymo procesą.

1.2.2. Modelio įvertinimas

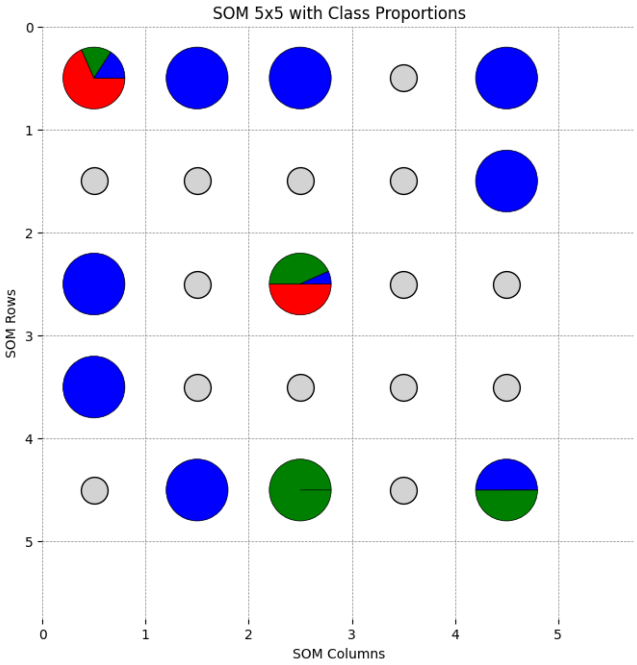
```
def quantization_error(data, som):
    error = 0
    for entry in data:
        distances = np.zeros((som.shape[0], som.shape[1]))
        for row in range(som.shape[0]):
            for col in range(som.shape[1]):
                distances[row, col] = np.sqrt(np.sum((som[row,col
                    ] - entry) ** 2))

        neuron_winner = np.unravel_index(np.argmin(distances),
            distances.shape)
        error += np.sqrt(np.sum((som[row,col] - entry) ** 2))
    return error / len(data)
```

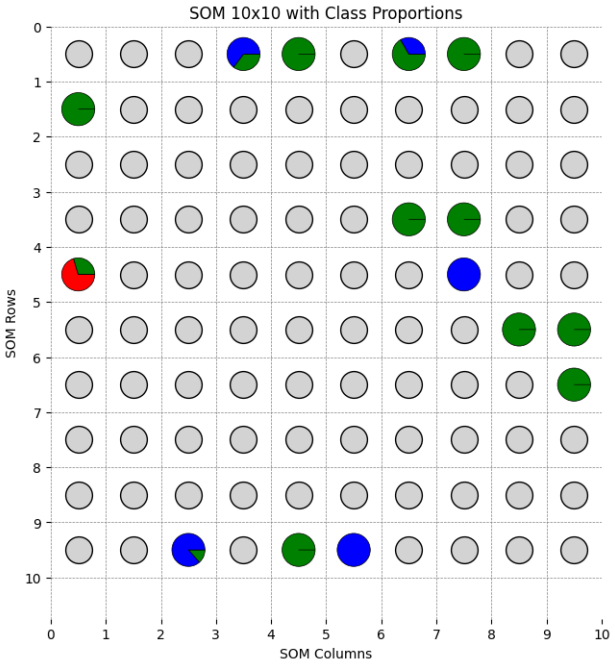
Ši funkcija apskaičiuoja kvantavimo paklaidą (quantization error), kuri matuoja vidutinį atstumą tarp duomenų vektorių ir jų artimiausių SOM neuronų. Kiekvienam duomenų įrašui apskaičiuojami atstumai iki visų neuronų, o artimiausias neuronas (winner neuron) nustatomas pagal mažiausią Euklido atstumą. Šio atstumo kvadratas pridedamas prie bendros paklaidos. Galiausiai, bendra paklaida padalijama iš duomenų kiekio, grąžinant vidutinę kvantavimo paklaidą, kuri rodo, kaip gerai SOM tinklas aprašo duomenis.

2. Tyrimas

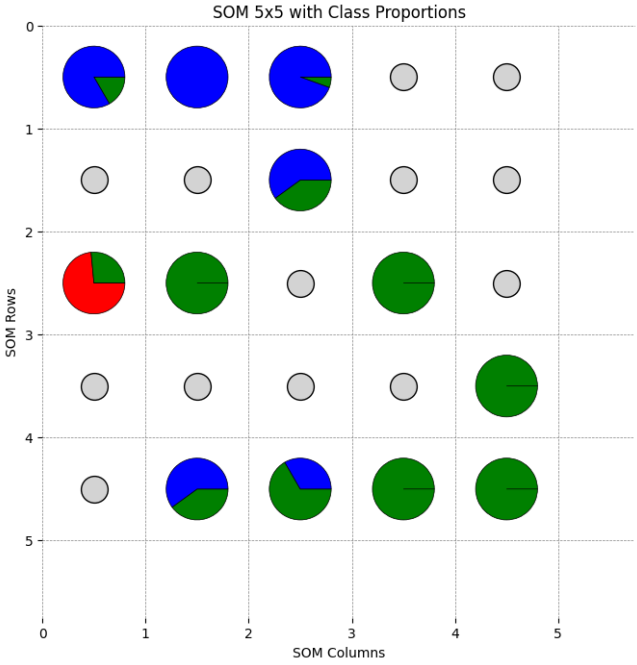
Tyrimo metu buvo apmokyti du SOM tinklai ant viso duomenų rinkinio, pirmą kartą po 10 epochų, kitą kartą po 100 epochų ir galiausiai po 1000 epochų. Gauti žemėlapiai pateikiami paveikslėliuose žemiau (1, 2, 5, 4, 3, 6). Apskritimo dalis atitinka dalį įrašų priskirtų tam neuronui nugalėtojui. Raudona spalva pažymėta klasė - Iris-setosa, žalia spalva - Iris-versicolor, mėlyna spalva - Iris-virginica.



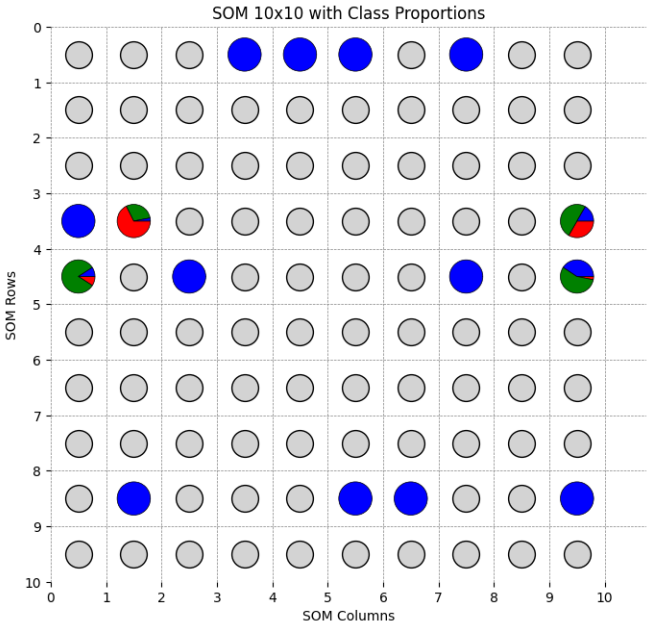
1 pav. SOM 5x5, 10 epochų



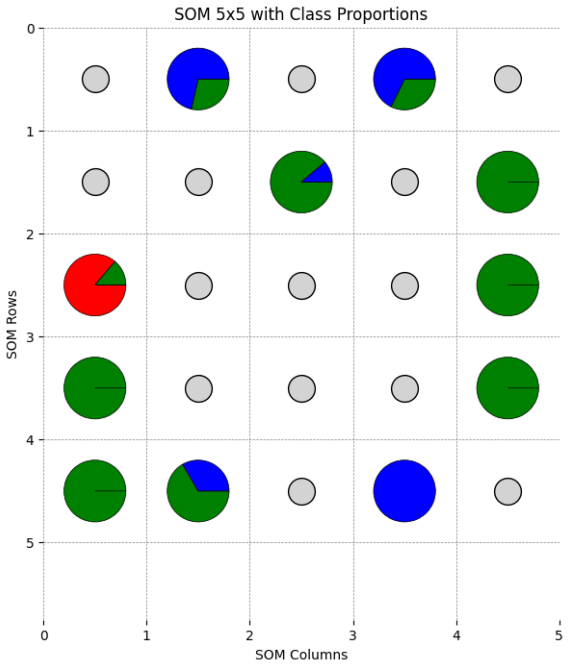
3 pav. SOM 10x10, 100 epochų



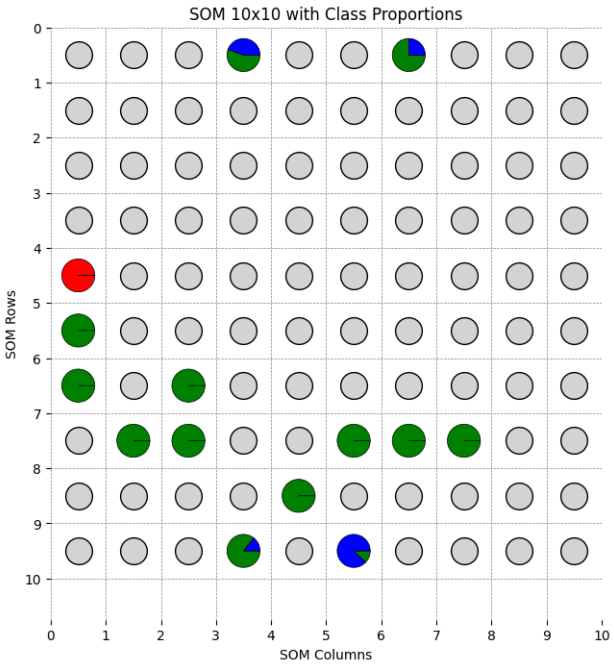
2 pav. SOM 5x5, 100 epochų



4 pav. SOM 10x10, 10 epochų



5 pav. SOM 5x5, 1000 epochų



6 pav. SOM 10x10, 1000 epochų

1 lentelė. Modelių metrikos

Epochų skaičius	10	100	1000
SOM 5x5 kvantavimo paklaida	0,5808	0,47697	0,4790
SOM 10x10 kvantavimo paklaida	0,5872	0,4774	0,4790

Išvados

1. Tyrimo metu SOM tinklas efektyviai sugrupavo Iris duomenis į tris klases, atitinkančias gėlių rūšis: Iris-setosa, Iris-versicolor ir Iris-virginica.
2. Iris-setosa klasė nuosekliai išskiriama ir aiškiai atskiriama nuo kitų dviejų klasių, o Iris-versicolor ir Iris-virginica klasės dažnai turi dalinį persidengimą dėl jų panašių atributų.
3. Tyrime naudojant mažesnį tinklą (5x5), duomenų grupės buvo labiau apibendrintos, tačiau didesnis tinklas (10x10) suteikė detalesnį vaizdą. Dideli tinklai leidžia smulkiau reprezentuoti klases, tačiau gali padidinti skaičiavimo sudėtingumą.
4. Eksperimentai parodė, kad tinklo konvergencija gerėja didinant epochų skaičių iki tam tikros ribos. Pvz., rezultatai po 1000 epochų buvo stabilūs ir tik šiek tiek skyrėsi nuo 100 epochų.
5. Visi atributai buvo tinkamai normalizuoti, kad būtų užtikrinta vienoda įtaka mokymo procesui. Tai ypač svarbu SOM algoritmui, nes jis remiasi Euklido atstumu, kuris yra jautrus skirtingiems skalių dydžiams.