VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMOS

Skaitmeninis intelektas ir sprendimų priėmimas
Saviorganizuojantys neuroniniai tinklai

Darbą atliko:
Pijus Zlatkus

Vilnius
2024

Turinys

1. **Užduoties tikslas** **3**
2. **Analizuojami duomenys** **3**
3. **SOM mokymo algoritmas** **3**
4. **Gauti rezultatai** **6**
5. **Išvados** **9**
6. **Šaltiniai** **10**

1. Užduoties tikslas

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

2. Analizuojami duomenys

Analizėje naudojami Irisų duomenys, kurie yra gerai žinomas duomenų rinkinys, sudarytas iš trijų skirtingų Iris gėlių rūšių (Setosa, Versicolor, ir Virginica). Kiekvienai duomenų eilutei yra keturi matavimai: taurėlapių ilgis ir plotis bei vainiklapių ilgis ir plotis.

SOM apmokymo duomenims atsisiųsti ir užkrauti buvo panaudota *sklearn* biblioteka.

3. SOM mokymo algoritmas

SOM mokymo algoritmas buvo suprogramuotas naudojant *numpy* ir *tensorflow* kodo bibliotekas, kurios skirtos dirbti su skaičiavimais ir dirbtinio intelekto modeliais. Rezultatų vizualizacijai buvo panaudota *matplotlib* kodo biblioteka.

Kadangi SOM yra įvairių kodo realizacijų, tai šiam tyrimui realizuoti buvo pasirinktas mokymo algoritmo pseudo-kodas iš „Saviorganizuojantys neuroniniai tinklai“ paskaitos skaidrių.

```
function SOM_training(X, M, ê, k_x, k_y)
// įvestis: X–duomenų aibė, M– pradiniai neuronai, ê – tinklo mokymo epochų skaičius,
// k_x, k_y – eilučių ir stulpelių skaičius
// išvestis: M – neuronai
BEGIN
FOR t=1 TO ê
   FOR l=1 TO m // duomenų aibės vektorius X_l pateikiamas į neuroninį tinklą
      FOR i=1 TO k_x
         FOR j=1 TO k_y
            ||M_ij − X_l|| := sqrt(Σ_{p=1}^{n} (m_p^ij − x_lp)^2)  // skaičiuojamas Euklido atstumas
         END
      END
      c := arg min_{i,j} {||X_l − M_ij||} // M̂_c – vektoriaus X_l neuronas nugalėtojas
      FOR i=1 TO k_x
         FOR j=1 TO k_y
            M_ij(t+1) := M_ij(t) + h_ij^c(t)(X_l − M_ij(t)) // SOM mokymo taisyklė
         END
      END
   END // visų vektorių peržiūrėjimo pabaiga
END // mokymo pabaiga
RETURN M
END
```

1 pav. *SOM mokymo algoritmo pseudo-kodas*

1 pav. SOM mokymo algoritmo pseudo-kodas atvaizduotas mokymo algoritmo pseudo-kodas palieka keletą interpretacijų kaip galima realizuoti šį algoritmą – leidžia pasirinkti kaimynystės funkciją SOM mokymo taisyklei. Populiariausios yra burbuliuko ir Gauso kaimynystės funkcijos. Šiai realizacija buvo pasirinkta Gauso kaimynystės funkcija:

$$h_{ij}^{c}(t) = \alpha(t) \cdot exp\left(\frac{-\left\|R_c - R_{ij}\right\|^2}{2\left(\eta_{ij}^{c}(t)\right)^2}\right)$$

Čia t – iteracijos numeris, $h_{ij}^{c}(t)$ – kaimynystės funkcija, $\eta_{ij}^{c}(t)$ – kaimynystės eilės numeris. Taip pat SOM tinklo mokymo kokybei nustatyti reikia pasirinkti paklaidos funkciją. Kaip paklaidos funkcija buvo pasirinkta kvantavimo paklaidos funkcija.

```python
# Funkcija tinklo mokymui
def train(self, data):
    self.weights_history = []  # Sąrašas svorio istorijai saugoti
    with tf.compat.v1.Session(graph=self.graph) as sess:
        sess.run(self.init_op)
        for epoch in range(self.num_epochs):
            for sample in data:
                feed_dict = {self.input: sample, self.iteration: epoch}
                sess.run(self.training_op, feed_dict=feed_dict)
            self.weights_history.append(sess.run(self.weights))
        self.trained_weights = sess.run(self.weights)
```

2 pav. *SOM mokymo algoritmo realizacija*

```python
# Funkcija kvantizacijos klaidai apskaičiuoti
def quantization_error(self, data):
    error = 0
    with tf.compat.v1.Session(graph=self.graph) as sess:
        sess.run(self.init_op)
        for sample in data:
            bmu_idx = sess.run(self._get_bmu_index(), feed_dict={self.input: sample})
            bmu_weight = sess.run(self.weights[bmu_idx])
            error += np.linalg.norm(sample - bmu_weight)
    return error / len(data)
```

3 pav. *Kvantavimo paklaidos funkcijos realizacija*

```python
# Apskaičiuojama kaimynystės funkcija
neighborhood_func = tf.exp(-tf.reduce_sum(tf.square(bmu_loc - self._location_vectors()), axis=1) / (2 * tf.square(sigma_op)))
```

4 pav. *Kaimynystės funkcijos realizacija*

Aukščiau pateiktuose **2 pav.**, **3 pav.**, **4 pav.** paveikslėliuose yra pateiktas SOM algoritmo realizacijos ištraukos. Šis kodas gali prisitaikyti prie žemėlapio dydžio ir epochų skaičiaus, todėl galima lengvai gauti ir palyginti mokymo rezultatus.

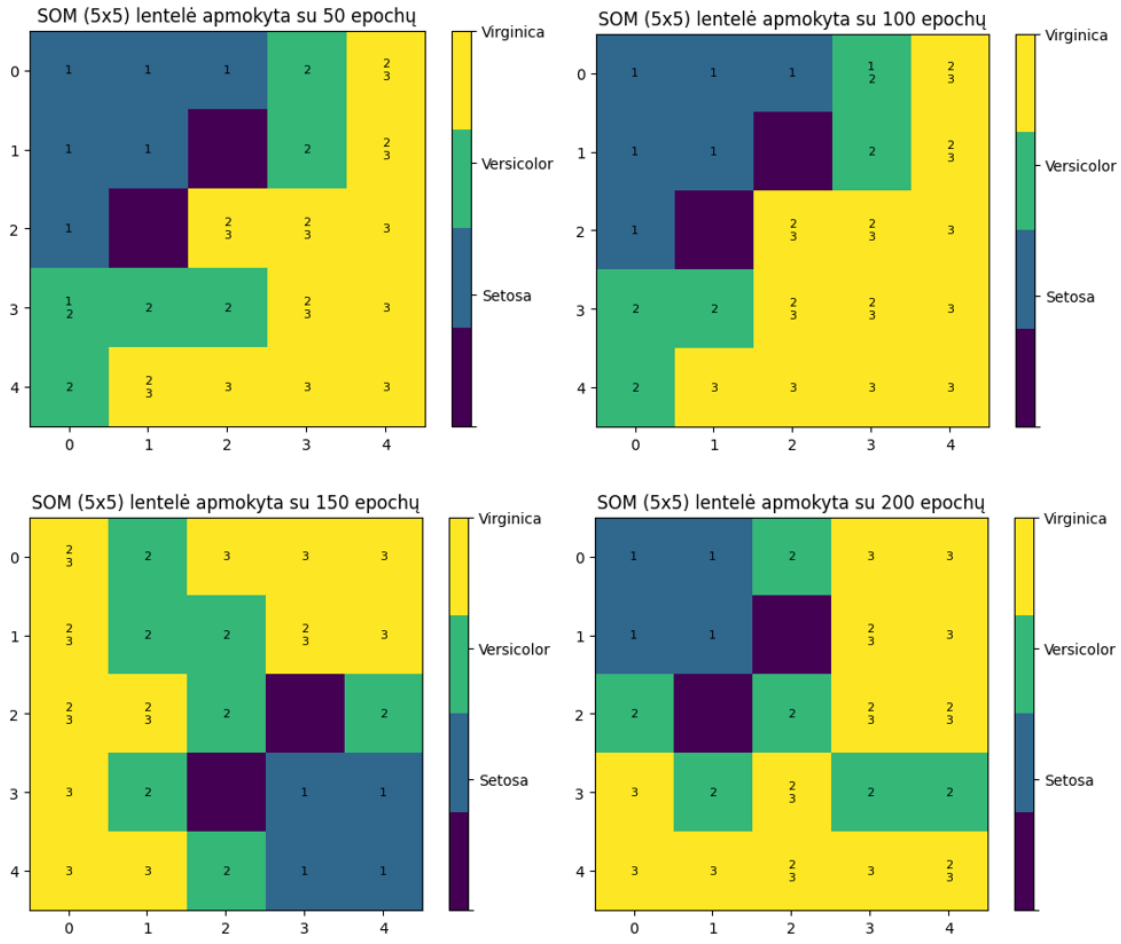
4. Gauti rezultatai

Tyrimas buvo atliktas apmokant 5x5 ir 10x10 dydžio žemėlapius su 50, 100, 150 ir 200 epochų iteracijų.

1 lentelė. *Kvantavimo paklaidos*

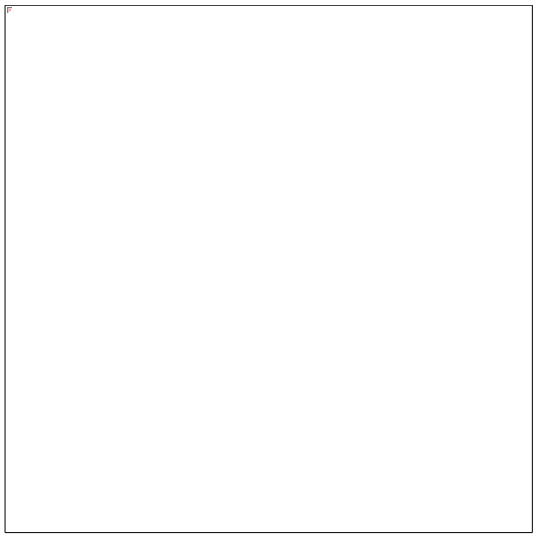
	SOM dydis	
Epochų skaičius	**5x5**	**10x10**
50	1,153	0,856
100	1,107	0,746
150	0,999	0,843
200	1,045	0,808

1 lentelė pateikiami kvantavimo paklaidos dydžiai skirtingiems SOM tinklo dydžiams bei skirtingiems epochų skaičiams. Kvantavimo paklaida matuoja vidutinį atstumą tarp duomenų taškų ir jų artimiausių neuronų SOM tinkle, todėl mažesnė kvantavimo paklaida rodo geresnę tinklo kokybę. Matome, kad didėjant epochų skaičiui, kvantavimo paklaida mažėja tiek 5x5, tiek 10x10 tinklams. Tai rodo, kad ilgiau mokant tinklą, neuronų tinklas geriau adaptuojasi prie duomenų ir sumažina klaidą, tačiau 10x10 dydžio tinklas turi mažesnę kvantavimo paklaidą lyginant su 5x5 tinklu esant visiems epochų skaičiams. Tai rodo, kad didesnis tinklas gali geriau išmokti duomenų struktūrą ir sumažinti klaidą.



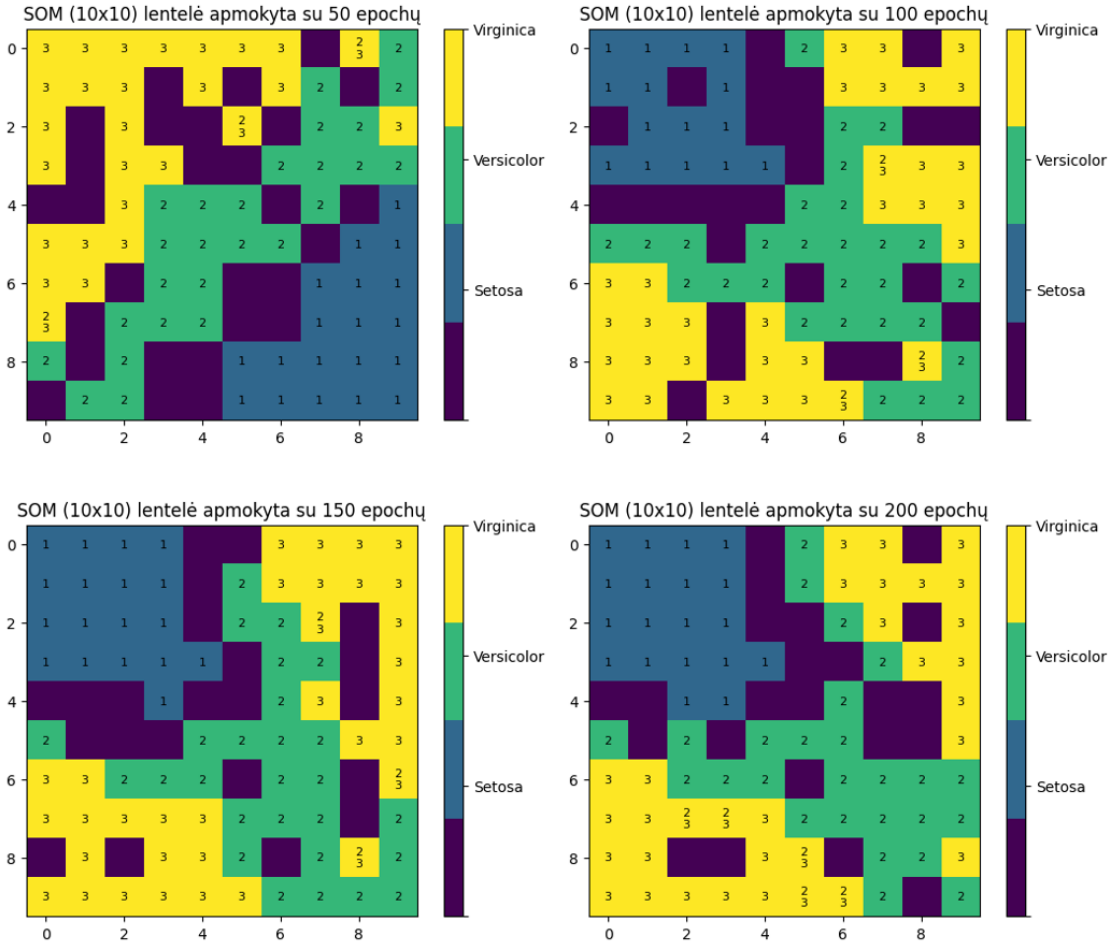
5 pav. *Gauti SOM 5x5 žemėlapiai nuo epochų skaičiaus*

5 pav. kiekvienas žemėlapis atspindi neuronų pasiskirstymą tinkle po tam tikro mokymo etapų skaičiaus (50, 100, 150, 200 epochų). Tai leidžia matyti, kaip tinklas adaptuojasi ir mokosi atsižvelgiant į epochų skaičių. Tačiau iš šių žemėlapių negalima nustatyti kaip tiksliai yra suklasterizuojami duomenys.

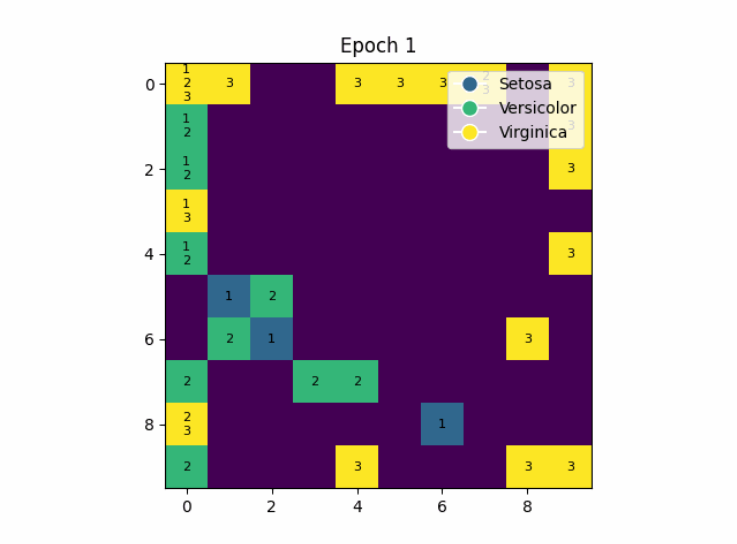


6 pav. *SOM 5x5 žemėlapis kiekvienoje epochoje mokymo metu*

6 pav. pateiktas SOM 5x5 žemėlapis, kuris parodo neuronų išdėstymą kiekvienoje epochoje mokymo metu. Tai padeda stebėti, kaip keičiasi tinklo struktūra per mokymo laikotarpį.



7 pav. *Gauti SOM 10x10 žemėlapiai nuo epochų skaičiaus*



8 pav. *SOM 10x10 žemėlapis kiekvienoje epochoje mokymo metu*

7 pav. ir **8 pav.** paveikslėliuose duomenys yra panašūs kaip ir 5x5 žemėlapiuose, tačiau galima pastebėti, kad jie yra detaliau atvaizduojami ir galima lengviau pastebėti susigrupavusius duomenys bei jų išsidėstymo struktūrą.

5. Išvados

Atlikus saviorganizuojančių neuroninių tinklų (SOM) mokymo algoritmų tyrimą, buvo gauti šie rezultatai ir padarytos išvados:

1. **Kvantavimo paklaida:**

- Matant kvantavimo paklaidos rezultatus lentelėje, pastebėta, kad kvantavimo paklaida mažėja didėjant epochų skaičiui tiek 5x5, tiek 10x10 dydžio tinkluose. Tai rodo, kad tinklas geriau mokosi ir adaptuojasi prie duomenų struktūros, kai mokymo trukmė yra ilgesnė.
- 10x10 dydžio tinklas turėjo mažesnę kvantavimo paklaidą lyginant su 5x5 tinklu visiems epochų skaičiams. Tai rodo, kad didesnis tinklas gali geriau išmokti duomenų struktūrą ir sumažinti klaidą, todėl jis yra efektyvesnis šiam uždaviniui.

2. **Mokymo rezultatai vizualizacijoje:**

- Žemėlapiuose su 5x5 tinklu (**5** ir **6 pav.** SOM 5x5 žemėlapis kiekvienoje epochoje mokymo metu) neuronų pasiskirstymas parodo, kaip tinklas adaptuojasi prie duomenų epochų skaičiui didėjant. Tačiau šie žemėlapiai yra mažiau detalūs ir juose sunkiau matyti duomenų klasterizaciją.

- Žemėlapiuose su 10x10 tinklu (**7 pav.** Gauti SOM 10x10 žemėlapiai nuo epochų skaičiaus ir **8 pav.** SOM 10x10 žemėlapis kiekvienoje epochoje mokymo metu) neuronų pasiskirstymas yra detalesnis, leidžiantis lengviau pastebėti duomenų klasterizaciją ir jų išsidėstymo struktūrą. Tai dar kartą patvirtina, kad didesnis tinklas yra efektyvesnis ir geba tiksliau atspindėti duomenų struktūrą.

Šis tyrimas parodė, kad didesnio dydžio SOM tinklai yra efektyvesni ir gali geriau mokytis duomenų struktūros, sumažindami kvantavimo paklaidą ir tiksliau atvaizduodami duomenų klasterizaciją.

6. Šaltiniai

- Skaitmeninis intelektas ir sprendimų priėmimo skaidrės „Saviorganizuojantys neuroniniai tinklai“