VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Skaitmeninis intelektas ir sprendimų priėmimas

V užduotis: Saviorganizuojantys neuroniniai tinklai

Tomas Giedraitis
VU MIF Informatika
3 kursas 3 grupė

Vilnius
2021

1. Darbo tikslas

Suprogramuoti saviorganizuojančio neuroninio tinklo (dar vadinamo žemėlapiu – angl. *self organizing map*, arba *SOM*) neprižiūrimo mokymo algoritmą duomenims klasterizuoti ir apmokyti jį naudojant pasirinktus duomenis.

2. Darbo eiga

2.1. Duomenys

Pasirinkti analizuoti trijų klasių irisų duomenys[1]. Šie duomenys pasiskirstę šitaip:

- pirmąją klasę sudaro Setosa rūšis (50 duomenų įrašų).
- antroji klasė – Versicolor rūšis (50 duomenų įrašų).
- trečioji – Virginica rūšis (50 duomenų įrašų).

Klasių žymės yra 1, 2 ir 3. Kiekvienas duomenų objektas, be nurodytos jo klasės, turi dar keturis požymius, nusakytus realiais skaičiais:

1. Taurėlapio ilgis, cm
2. Taurėlapio plotis, cm
3. Žiedlapio ilgis, cm
4. Žiedlapio plotis, cm

Klasių rūšių, apibūdinami šiais požymiais, paveiksliukai pateikti 1 lentelėje:

1 lentelė. Irisų rūšių pavyzdžiai

Klasė	Klasės pavadinimas	Pavyzdys
1	Iris Setosa	
2	Iris Versicolor	

[1] https://archive.ics.uci.edu/ml/datasets.html

3	Iris Virginica	

Duomenų failas `iris.data` kartu su keliais kitais duomenų informaciniais failais pateikiamas A.1 priede.

2.1. SOM mokymo algoritmas

Pagrindinės SOM mokymo algoritmo programos sudedamosios dalys:

- Funkcija, nuskaitanti duomenis iš tekstinio failo, kurio eilutėse – irisų duomenų objektai (įrašai), kurių atributai atskiriami kableliu. Pagal funkcijai paduotą argumentą arba yra imami skirtingi mokymo ir testavimo duomenys (testavimo duomenų procentinė dalis irgi nustatoma tuo pačiu argumentu), arba visi iš failo nuskaityti duomenys naudojami ir mokymui, ir testavimui. Tiek mokymo, tiek testavimo duomenys yra šiek tiek perskirstomi taip, kad iš eilės sektų 1-os klasės duomuo, po jo – 2-os klasės, ir galiausiai – 3-ios klasės, ir tai kartotųsi vėl sekantiems trims duomenų elementams, tokiu būdu įgyvendinant duomenų permaišymą, nors ir gana trivialų šiuo atveju.
- Funkcija, apmokanti saviorganizuojantį neuroninį tinklą, nurodžius pagrindinius įvesties parametrus – SOM tinklelio dydį ir epochų skaičių. Šiame tyrime buvo naudojamas stačiakampis tinklelis. Galimi ir papildomi parametrai, jie bus taip pat aptarti. Funkcijos rezultatas – SOM neuronų svorių reikšmės, neuronai-nugalėtojai ir jiems priskirtų duomenų objektų numeriai.
- Funkcija, atliekanti SOM mokymosi kokybės patikrinimą. Ji apskaičiuoja kvantavimo bei topografinės paklaidų reikšmes.
- Funkcija, pavaizduojanti SOM lentelę (tinklelį) terminale (konsolėje), kurioje išdėstyti analizuojami duomenys. Pagal vartotojo pasirinkimą, tinklelyje gali būti nurodytos duomenų klasės arba duomenų numeriai. Tinklelyje atsiranda visi įvesties vektorių numeriai (arba jų klasės). Į viena langelį, žinoma, gali patekti daugiau nei vienas duomuo.

2.2. Programos parametrai

Duomenų parametrai:
`data = get_data(test_data_percentage=PERCENTAGE, all=ALL)` – į `data` kintamąjį išsaugoma funkcijos `get_data` grąžinama reikšmė – keturi masyvai: mokymo duomenys, mokymo duomenų klasės, testiniai duomenys, testinių duomenų klasės.

Funkcijos argumentas `test_data_percentage` nustatomas į sveikojo skaičiaus reikšmę $\texttt{PERCENTAGE} \in \{1,2, \dots, 50\}$, nusakančią, kokia procentinė duomenų dalis bus naudojama testavimo duomenims. Pagal nutylėjimą `test_data_percentage` = 10 (t.y. 10 %).

Funkcijos argumentas `all` nustatomas į loginę reikšmę $\texttt{ALL} \in \{\texttt{True}, \texttt{False}\}$, kuri nusako, ar naudoti visus įvesties duomenis tiek mokymui, tiek testavimui. Nustačius šią reikšmę į `True`, `test_data_percentage` argumentas ignoruojamas. Pagal nutylėjimą `all = False`.

Visi kiti programos parametrai nustatomi globaliais kintamaisiais:

Pagrindiniai programos parametrai:
`kx`, `ky` – SOM tinklelio dydis (eilučių ir stulpelių skaičius).
`epochs` – epochų skaičius.

Papildomi parametrai:
`VICINITIES` – nurodo, kokio „tolimumo“ kaimynai bus naudojami burbuliuko kaimynystės funkcijoje *h*. Pagal nutylėjimą reikšmė yra 3, bet galima pasirinkti ir kitą skaičių, pvz. jei būtų 10 – tai iteracijų skaičius bus dalinamas iš 10-ies ir pirmosiomis 10 iteracijų kaimynais bus laikomi net 10-os eilės kaimynai, kol galiausiai priartėjus prie paskutinių iteracijų bus tikrinami tik patys artimiausi kaimynai. Tokiu būdu galima testuoti SOM su didesniais tinkleliais.

`H_FN_KIND` – galima pasirinkti vieną iš dviejų kaimynystės funkcijų *h* – burbuliuko (pagal nutylėjimą), arba Gauso.

`ALPHA_FN_KIND` – galima pasirinkti vieną iš trijų α kaimynystės funkcijų, dar vadinamų mokymo parametru:

(1) `simple_div` (paprasta dalyba): $\alpha(t) = 1/t$
(2) `simple_div_sub` (dalyba su atimtimi): $\alpha(t) = 1 - t/T$
(3) `power` (laipsninė funkcija): $\alpha(t) = (0.05)^{t/T}$

Čia *t* yra einamosios iteracijos numeris, o *T* – visų iteracijų (per vieną epochą) skaičius.
Pagal nutylėjimą naudojama paprastos dalybos funkcija (1).

Programoje taip pat galima nustatyti `RANDOM_SEED` reikšmę, kuri pagal nutylėjimą lygi 0. Ji reikalinga fiksuoti randomizacijos pradžios tašką, kad tyrimas galėtų būti pakartojamas. Randomizacija yra atliekama prieš mokymo procesą, kuomet neuronų (vektorių) matricos M_{ij} komponenčių $m_1^{ij}, m_2^{ij}, \dots m_n^{ij}$ reikšmės nustatomos atsitiktinai intervale (0; 1).

`SHOW_CLASS_NAMES` kintamuoju galima pasirinkti, kad tinklelyje būtų atvaizduojamos įvesties duomenų klasės (pagal nutylėjimą), arba jų numeriai.

Jei norima papildomai atvaizduoti SOM neuronų svorių reikšmes po mokymosi proceso, bei neuronus-nugalėtojus ir jiems priskirtų duomenų objektų numerius bei jų klases, galima pažymėti `VERBOSE_OUTPUT` parametrą. Pagal nutylėjimą jis yra aktyvuotas.

Programa su komentarais pridedama A.2 priede. Įvestis joje šiuo metu yra nustatyta tokia: 135 mokymo duomenys (po 45 kiekvienai irisų klasei), ir 15 testavimo duomenų (po 5 kiekvienai klasei). Mokymo ir testavimo duomenų kiekį programoje taip pat galima pakeisti.

2.3. Tyrimai

Programoje yra galimybė atlikti išsamius testus, keičiant įvairius parametrus. Tik keli iš šių testų pateikiami šiame tyrimų apraše. Visų tyrimų išsamūs rezultatai (išvestys) pateikiami A.3 priede.

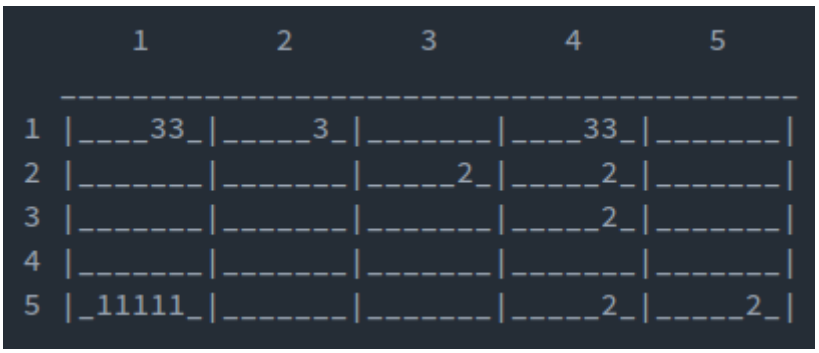
Tyrimas nr. 1
Duomenys mokymui:
135 irisų duomenys, po 35 kiekvienai klasei.

Duomenys testavimui:
15 irisų duomenų, po 5 kiekvienai klasei.

Parametrai:
Tinklelio dydis: 5×5
Epochų skaičius: 10
Pradinio tolimumo (`VICINITIES`) reikšmė: 3
Kaimynystės funkcija *h:* burbuliuko
Mokymo parametras *α:* $\alpha(t) = 1/t$
`RANDOM_SEED` reikšmė: 0

Rezultatai (nr. 1):
Kvantavimo paklaida: 0.57573
Topografinė paklaida: 0.05926
(suapvalinta iki 5 sk. po kablelio)

SOM žemėlapis:



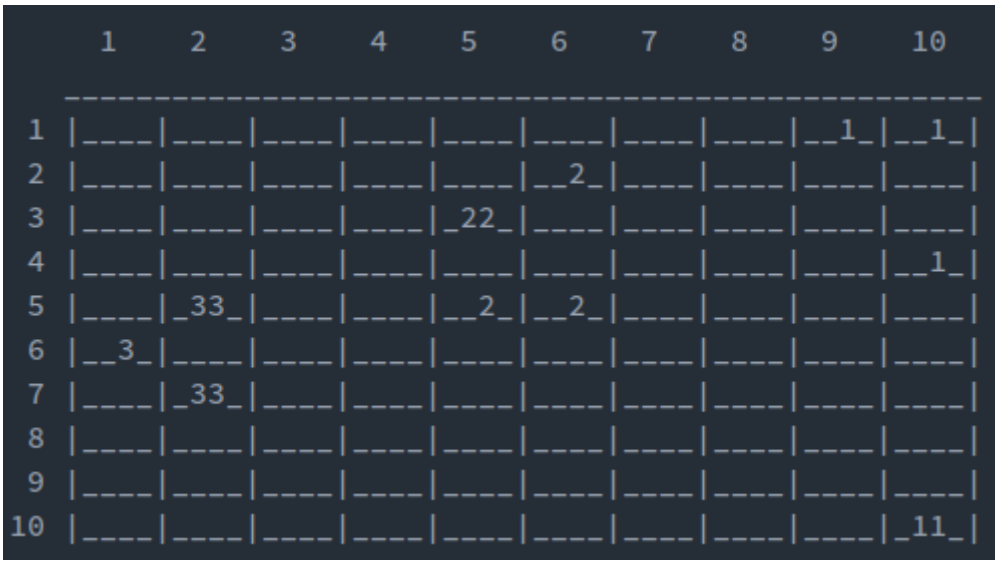
1 pav. Tyrimo nr. 1 SOM žemėlapis

Tyrimas nr. 2

Parametrai – tie patys, tik tinklelio dydis yra 10×10.

Rezultatai (nr. 2):
Kvantavimo paklaida: 0.36548
Topografinė paklaida: 0.05185

SOM žemėlapis:



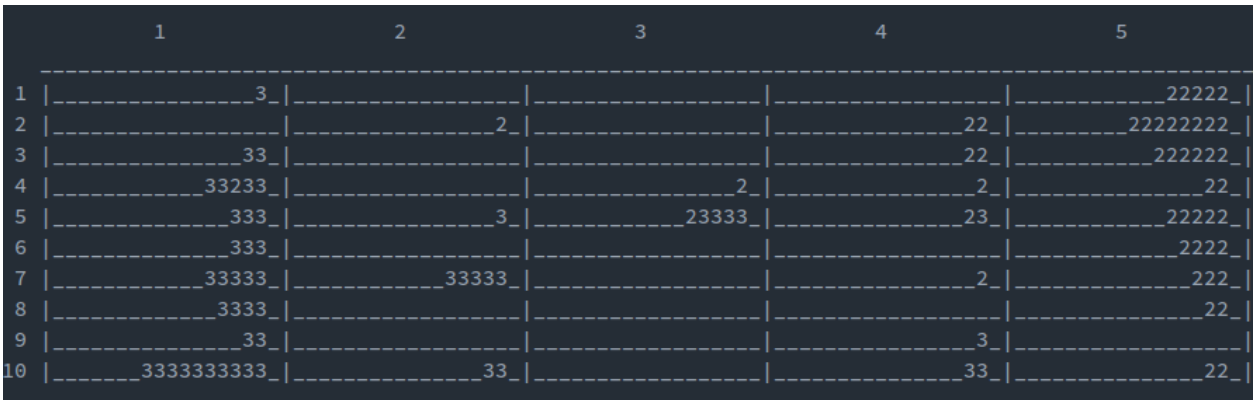
2 pav. Tyrimo nr. 2 SOM žemėlapis

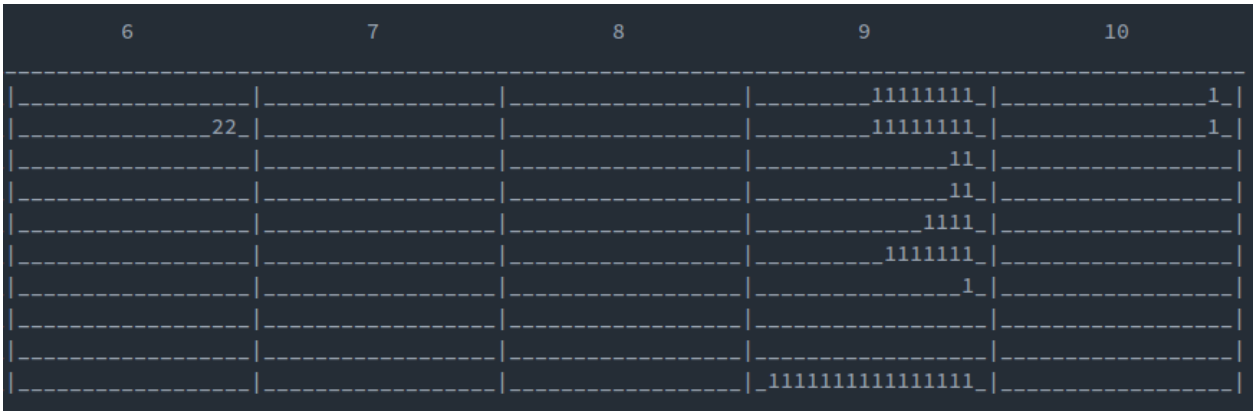
Tyrimas nr. 3

Naudoti tie patys parametrai, tinklelio dydis – 10×10, tik mokymo ir testavimo aibės šįkart sutampa, ir jas sudaro visi 150 irisų duomenų.

Rezultatai (nr. 3):
Kvantavimo paklaida: 0.37476
Topografinė paklaida: 0.02000

SOM žemėlapis (išvesties tinklelis padalintas į du paveiksliukus):





3–4 pav. Tyrimo nr. 3 SOM žemėlapis

Tyrimų nr. 1, 2, 3 interpretacija

Dar sykį pateikiame tyrimų mokymosi paklaidas vienoje vietoje:

Tyrimas nr. 1:

Kvantavimo paklaida: 0.57573

Topografinė paklaida: 0.05926

Tyrimas nr. 2:

Kvantavimo paklaida: 0.36548

Topografinė paklaida: 0.05185

Tyrimas nr. 3:

Kvantavimo paklaida: 0.37476

Topografinė paklaida: 0.02000

Klasterizavimas:
Iš pateiktų rezultatų ryškiai matome duomenų grupavimosi tendencijas.

Pirmuoju (tyrimas nr. 1) ir antruoju (tyrimas nr. 2) atveju, galima sakyti, kad klasterius galima pilnai atskirti, tinklelyje apvedus uždaras kreives aplink klasterius (tiksliau – juos atskirti nuo kitų klasių tinklelyje kažkokia tai uždara geometrine figūra, tokia, kad visos vienodos klasės patektų į jos vidų, ir joje nebūtų nė vieno kitos klasės duomens).

Tuo tarpu trečiuoju atveju, kai buvo panaudoti visi mokymosi duomenys testavimui, klasteriai su klasėmis 2 ir 3 yra šiek tiek persipynę tarpusavyje, t.y. abiejų klasių elementai kelis kartus patenka į vieną langelį ([4;1], [5;3] ir [5;4] langeliuose). Kaip ir žinome iš šių irisų duomenų, 2-os ir 3-ios klasės (Versicolor ir Virginica) duomenys nėra tiesiškai atskiriami. Tačiau nepaisant šio nedidelio persipynimo, galima labai aiškiai įžiūrėti visus tris klasterius, na o aplink 1-ukų klases ir trečiuoju atveju galime apibrėžti uždarą kreivę – vėlgi, žinome iš irisų duomenų, kad 1-os klasės (Setosa) duomenys labiau skiriasi nuo kitų dviejų klasių, ir yra pilnai tiesiškai atskiriamos nuo jų.

Klasterizavimas ir paklaidos:

Kadangi pradžioje testavimui panaudojome tik 15 įvesties duomenų, matome, kad lyg ir pilnai užteko 5×5 dydžio tinklelio klasteriams pamatyti ir atskirti. Tačiau 10×10 tinklelio panaudojimas davė beveik 2 kartus mažesnę kvantavimo paklaidą, vadinasi, išmokyto tinklo neuronai labiau prisiderina prie mokymo aibės vektorių. Tą galima paaiškinti tuo, kad atsiranda daugiau neuronų, taigi kiekvienam įvedimo vektoriui atsiranda daugiau ir skirtingų kandidatų į neuroną nugalėtoją – daugiau skirtingų euklido atstumų, iš kurių galima pasirinkti mažiausią. Taip pat sumažėjo ir topografinė paklaida sumažėjo (14 % sumažėjimas).

Trečiuoju atveju (tyrime nr. 3) kvantavimo paklaida gavosi kaip ir antruoju – ji beveik nepasikeitė, nors testavimui naudojome ne atskirą aibę, o tą pačią mokymo aibę. Ir pačių testavimo duomenų buvo nebe 15, o 150. Pagal tai galima daryti prielaidą, kad tyrime nr. 2 testavimo duomenys buvo ganėtinai panašūs į mokymo duomenis, t.y. ši imtis gerai atspindėjo visus duomenis.

Topografinė paklaida tyrime nr. 3 gavosi mažiausia. Tai galėjo gautis dėl to, kad pačių duomenų buvo daugiau, na o duomenys, pagal 1-ą ir 2-ą tyrimą, turi tendenciją grupuotis kartu, tai padidėjus duomenims, padidėjo ir santykinai arčiau sugrupuotų duomenų, nei toliau.

3. Išvados

Buvo suprogramuotas SOM tipo neuroninis tinklas duomenims klasterizuoti, ir buvo atlikti neurono mokymai ir testavimai su irisų duomenimis, keičiant SOM tinklo parametrus.

Buvo atlikti tyrimai, kaip vienas pagrindinių SOM parametrų – tinklelio dydis – daro įtaką SOM rezultatams (tikrinti 5×5 ir 10×10 tinkleliai), ir kaip kito rezultatai, iš pradžių testavimui pateikus atskirus duomenis, o po to – ir visus tuos pačius mokymosi duomenis.

SOM rezultatai buvo matuojami per kvantavimo ir topografinę paklaidas, o taip pat ir vizualiai žvelgiant į klasterius tinklelyje. Rezultatai atitiko irisų duomenų pasiskirstymą, o testuojant su visais duomenimis, pamatėme ir 2-os bei 3-ios irisų klasių tiesišką neatskiriamumą.

Programoje taip pat įgyvendinta galimybė keisti kitą pagrindinį SOM tinklo parametrą – epochų skaičių. Taip pat, yra galimybė keisti ir papildomus SOM parametrus – kaimynų pradinį tolimumą, kaimynystės funkcijos h bei jos viduje esančios funkcijos α tipus. Tyrimai keičiant šiuos parametrus palikti kaip galimybė, ir užeina už šios ataskaitos ribų.

Įsitikinome, kad SOM tinklas gali puikiai veikti kaip neprižiūrimo mokymosi neuroninis tinklas duomenims klasterizuoti, ir taip pat jis turi puikią galimybę vizualizuoti klasterius.

Taip pat SOM suteikia galimybę vis platinti savo žvilgsnio lauką nuo tankiausių klasterių ir įžvelgti klasterių klasterius, o tai irgi yra labai naudinga informacija. Taip pat naudinga yra galimybė tiek atvaizduoti įvesties duomenų klases, tiek jų eilės numerius, ir tai duoda reikalingos informacijos, priklausomai nuo klasterizavimo tikslo, ir, bet kokiu atveju, galima atvaizduoti ir viena, ir kita.

SOM architektūra skiriasi nuo įprastų neuroninių tinklų architektūrų, tačiau tai gali būti ir privalumas, nes SOM architektūra nėra sudėtinga ir yra nesunkiai suprogramuojama. Tokioje architektūroje lengviau yra aptikti jos lėtesnes dalis, silpnąsias puses ir tuomet greičiau optimizuoti.

A Priedai

A.1 Aplankas `dataset/` su irisų duomenimis:

`Index`
`iris.data`
`iris.names`

A.2 SOM įgyvendinanti program `som.py`

A.3 Aplankas test_`results/` su tyrimų rezultatų pilnomis išvestimis:

`test_no1.txt`
`test_no2.txt`
`test_no3.txt`