Ataskaita

Laimonas Beniušis, 1410102

5 Užduotis
SOM (Self Organizing Map) mokymo algoritmo įgyvendinimas ir jo panaudojimas

Programos aprašas:
JAVA kalba
3 klasės:
- SOM (pagrindinis algoritmas)
- CSVparser (csv tipo bylų skaitytuvas)
- Vector (klasė grupuoti vektoriams (1xN matricoms) ir duomenims)

test.java kuri aprašo 3 SOM naudojimo atvejus
Jeigu norite matyti visą SOM mokymo procesą, pakeisti sukurto SOM objekto parametrą trace = true

SOM algoritmas:
Inicializacija:
1. Pasirinkti neuronų skaičių
2. Nustatyti visiems neuronams atsitiktinius svorius

Mokymas:
1. Pasirinkti 1 iš įvedimo vektorių
2. Pamatuoti Euklido atstumą su visai neuronais, tokiu būdu rasti artimiausią
3. Artimiausio neurono svorius pakoreguoti, kad jis priartėtų prie įvedimo vektoriaus
4. Pakoreguoti artimiausio neurono kaimynus
5. Pamažinti daugiklius, kurie turi įtaką svorių keitimui
6. Kartoti

Kaimynų svorių kaita priklauso nuo atstumo, kuris mažėja nuo iteracijų skaičiaus. Taip pat tolimesni kaimynai yra koreguojami mažiau negu artimesni. Atstumas tarp kaimynų yra skaičiuojamas kaip Manhattan atstumas lentoje (4-connectivity). Detalesnį aprašą (įgyvendinimą) galite rasti programos kode.

Mokymo iteracija – įvedimo vektoriaus pasirinkimas ir neuronų svorių adaptavimas pagal jį.
Taip pat galima SOM mokyti su visais įvedimo vektoriais X kartų. Tuomet X būtų epochų skaičius.

Gauti rezultatai
Mokymo iteracijų kiekis – 1000
Kiekvienos iteracijos metu, svorių pokytis sumažėja 1% (decayRate = 0.99)
Analizei naudojami duomenys - Irisų duomenų byla. (120 įrašų, 40 kiekvienos klasės)
Duomenys yra normalizuojami [0;1] intervale
Duomenų klasės:

1. Iris-setosa
2. Iris-versicolor
3. Iris-virginica

Klusterizavimas 10x10

```
1  1  1     1        1  1  1
1  1     1  1        2     1
   1  1        2  2  2
               2  2  2  2  2
2  2           2  2  2     2  2
2  2     2     2  3  3  3
         2     2  3     3     3
3  3     3  3  3  3     3
3     3     3                 3
3  3  3  3  3  3  3  3  3
```

Gautos paklaidos:
Kvantavimo: 0.07946
Topologinė: 0.2

Klusterizavimas 5x5

```
3  3  3  3  3
3  2  3  3  3
3  2  2  2
3  2  2        1
   2  1  1  1
```

Gautos paklaidos:
Kvantavimo: 0.103759
Topologinė: 0.1

Po 500 iteracijų „decay“ parametras tampa <0.007, todėl didelių pokyčių po 500-os iteracijos nėra, todėl verta sumažinti iteracijų limitą pusiau.
Žinoma, su mažai iteracijų, SOM neįsisavina duomenų klusterių.

Taip pat galima analogiškai klusterizuoti testavimo (naujus, nematytus) duomenis.

Rezultatams turi įtakos:

- Neuronų skaičius
- Iteracijų skaičius
- decayRate

Rezultatams neturi įtakos:

- Pradinių neuronų svorių reikšmės (su 0 ar su 1 buvo gauti panašaus tikslumo klusteriai)

SOM naudojimas duomenų klasifikavimui

SOM gali būti naudojamas duomenų įrašų atskirimui į klases.
SOM apmokomas su 120 Irisų įrašų ir pats atskiria juos į klases (klusterius).
Šiuo atveju SOM turi tik tiek neuronų, kiek yra duomenų klasių (3), todėl nėra prasmės iteracijos metu keisti jo kaimynų. Taip pat inicializacijos metu nereikėtų nenaudoti atsitiktinų svorių, nes gali būti kad jie mažai skirsis ir klaidingai neatskirs 3 klasių, o tik pvz 2. (Pasitaikė toks atvejis)

Kada SOM yra apmokytas, suvedami nauji (nematyti) testavimo duomenys (30 įrašų) ir SOM priskiria jiems klusterį. Jeigu testavimo įrašo klasės sutampa su SOM skirtu klusteriu, tada SOM teisingai priskyrė klusterį.
Iteracių skaičius – 1000
decayRate – 0.99

0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
2	1
1	1
2	1
1	1
1	1
1	1
1	1
1	1
1	1
1	1
2	2
1	2
2	2
2	2
2	2
2	2
1	2
2	2
2	2
2	2

Kaip matome, SOM padarė 4 klaidas (nesutampa klusterio-klasės poros reikšmės)
Klusteriai gali išsidėstyti ir kitaip, todėl naudotojui reikia interpretuoti rezultatą naudojant SOM kaip klasifikavimo įrankį.