Pēteris Račinskis
pr20015

Mājas darbs 2.5: EM, PageRank

Uzdevums 1

a. Seed parametra nozīme

Vienkārši darbinot EM algoritmu ar noklusējuma parametriem un dažādām seed vērtībām top skaidrs, ka rezultāts nav stabils - tiek iegūti dažādi klasteru skaiti. Pēc tam mainot *seed* ar konkrētu klasteru skaitu redzams, ka pie katra klasteru skaita iegūtais rezultāts ir stabils - mainās klasteru secība rezultātu izdrukā, bet iegūtais punktu iedalījums, šķiet, vienmēr (vai vismaz ļoti bieži) ir tāds pats.

b. Elkoņa metode, automātiska klasteru skaita noteikšana

Automātiskā klasteru skaita noteikšana nav stabila, tāpēc, lai salīdzinātu to ar elkoņa likumu, nepieciešams noturēt konstantu nejaušo *seed* parametru. Pie *seed* = {246236, 100}, kas automātiskā variantā atrod attiecīgi 5 un 2 klasterus, logaritmiskās ticamības ir sekojošas:

- k=1, l1=-3.06427, l2=-3.06427;
- k=2, l1=2.66793, l2=2.67064;
- k=3, l1=3.38786, l2=3.38786;
- k=4, l1=6.37105, l2=6.37105;
- k=5, l1=6.36237, l2=6.36237;
- k=6, l1=7.19342, l2=5.87855;
- k=7, l1=7.63957, l2=7.87008.

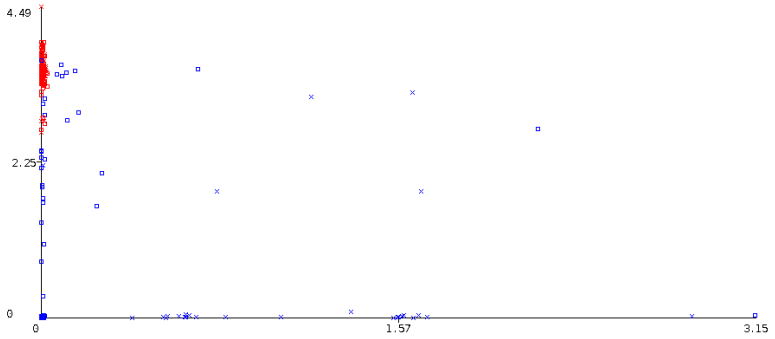
Spriežot pēc algoritma apraksta, Weka neizmanto elkoņa likumu tiešā veidā, bet gan kaut kā novērtē iegūto dalījumu, izmantojot krosvalidāciju. Redzams, ka nevar paļauties uz faktu, ka logaritmiskā ticamība vienmēr būs augoša - ir gadījumi, kad vērtība atkal samazinās. Turklāt var notikt lēcieni - šķiet, ka jau sasniegta asimptote, bet tad nākamajam klasteru skaitam lielums jau atkal ir lielāks. Tāpēc elkoņa likumu šeit grūti viennozīmīgi pielietot. Vienkārši dalot starpības starp katru punktu un tā abiem kaimiņiem, un apstājoties, kad attiecība vairs nepalielinās, abos gadījumos jāapstājas pie k=2 - taču nevar droši apgalvot, ka šis ir labs elkoņa likuma formalizējums. *Python* skripts, kas to dara, un abi saraksti ar ticamībām pielikumā.

c. Divu klasteru gadījums

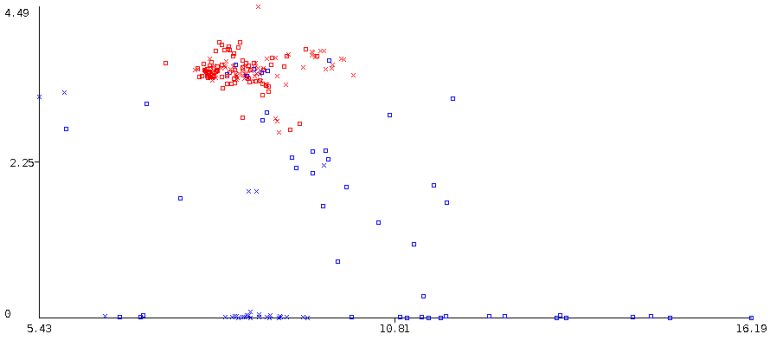
Tūlīt sākt pētīt projekcijas, darbinot EM algoritmu ar *numClusters*=2, varbūt nebūtu īpaši produktīvi, taču rezultātu izdrukā ļoti izpalīdzīgi iekļauti katram klasterim atbilstošā vidējā vērtība un dispersija katram datu kopas atribūtam.

Attribute	Cluster 0 (0.34)	1 (0.66)
RI		
mean	1.5192	1.5179
std. dev.	0.0043	0.0019
Na		
mean	13.7691	13.2188
std. dev.	1.1509	0.4625
Mg		
mean	1.0415	3.5445
std. dev.	1.3372	0.2575
Al		
mean	1.7346	1.2933
std. dev.	0.6534	0.296
Si		
mean	72.6225	72.6658
std. dev.	1.1235	0.4984
K		
mean	0.4878	0.5019
std. dev.	1.0772	0.1939
Ca		
mean	9.6926	8.572
std. dev.	2.1123	0.5517
Ba		
mean	0.5089	0.0003
std. dev.	0.7392	0.0043
Fe		
mean	0.0508	0.0603
std. dev.	0.1006	0.0952

Uzreiz redzams, ka lielākā daļa parametru ir līdzīgi abiem klasteriem un starpība starp vidējām vērtībām nav ievērojami lielāka kā standartnovirze, izņemot *Mg*, kur vienam klasterim ir izteikti nobīdīts centrs ar nelielu dispersiju, un *Ba*, kur šim pašam klasterim, šķiet, gandrīz visur ir 0. Tātad projicējot datus plaknē, kas, visticamāk, apraksta magnija un bārija īpatsvaru katra punkta - stikla parauga - saturā - iegūst izteiktu dalījumu klasteros (izmantota neliela *jitter* - nejaušu nobīžiu - vērtība, lai varētu labāk saskatīt vietas, kur daudzi punkti ir projicēti viens otram ļoti tuvu):



Redzams, ka faktiski nav sarkanā klastera punktu zem noteiktas magnija vērtības un virs noteiktas bārija vērtības. Iegarens klasteris sanāks, projicējot magniju pret jebkuru citu piejaukumu, taču pārējos gadījumos tad abi klasteri stipri pārklāsies gar otru izvēlēto asi (piemērā - kalcijs, kas nošķirt klasterus īsti nepalīdz):



Pētot klašu piederību klasterim, atklājas, ka sarkanajam klasterim pieder pārsvarā "*float processed*" stikls no ēku vai automašīnu logiem, savukārt zilajam - taras, automašīnu lampu un trauku stikls. Ēku logi, kas nav "*float processed*" relatīvi biežāk tiek iedalīti zilajā klasterī (kaut gan to vairākums ir sarkanajā).

Uzdevums 2

a. PageRank pielietojums futbola spēlētāju vērtēšanai

Process, kā tika izvēlēts šis raksts bija sekojošs: tika atvērts *Wikipedia* raksts par *PageRank* algoritmu, atrasta "*other uses*" sekcija, un izvēlēts pirmais, kas iekrita acīs, kā acīmredzami atšķirīgs no *Google* meklētāja, turklāt bija viegli pieejams bez maksas. Līdz ar to ir liela varbūtība, ka citi kolēģi to jau ir iztirzājuši, jo savā ziņā varētu teikt, ka šim rakstam ir augsts rangs starp potenciāli šajā mājas darbā apskatāmajiem. Raksta nosaukums ir A network theory analysis of football strategies, tas publicēts angļu valodā 2012 gadā un brīvi pieejams *arxiv* platformā.

Svērtais, orientētais grafs, kurā tiek meklētas sakarības, ir uzdots kaimiņu matricas (*adjacency matrix*) formā, kur A_{ij} ir piespēļu skaits no i-tā spēlētāja uz j-to. Tā kā raksts nav tikai par PageRank algoritmu, tiek apskatīti arī citi lielumi, ko var aprēķināt individuāliem spēlētājiem, to pāriem, grupām un grafam kopumā, taču konkrēti *PageRank* aprēķinam piedāvātais vienādojums ir faktiski tāds pats, kā lekciju materiālos aprakstītais, tikai ar citiem simboliem:

- x_i, x_j - spēlētāju rangi;
- L_j^{out} - virsotnes izejošo šķautņu skaits, t.i., spēlētāja kopējais padoto piespēļu skaits;
- p, q - "damping factor" ekvivalents, svari iespējām, ka tiks vai netiks padota piespēle, kad spēlētājs kontrolē bumbu
- $x_i = p \sum_{j \neq i} \frac{A_{ji}}{L_j^{out}} x_j + q$ - ranga vienādojums.

Katra spēlētāja rangs līdz ar to ir savā ziņā "popularitātes" novērtējums, jo atspoguļo to, cik daudzas piespēles spēlētājs saņem, un no cik "populāriem"

spēlētājiem. Raksta secinājumu daļā lielākā daļa teksta veltīta citu - visu grafu aprakstošu - parametru iztirzāšanai, jo, acīmredzot, autori ir uzskatījuši, ka iegūtie spēlētāju "popularitātes" novērtējumi īpašus komentārus neprasa. Subjektīvu novērtējumu varētu sniegt, pārskatot, piemēram, Vācijas komandas novērtējumu

Player	C_i	$C_B(i)$	x_i	c_i^w
Neuer	7.58	0.37	4.74	21.54
Friedrich	9.29	3.55	10.08	24.99
Khedira	8.70	10.58	11.38	26.31
Schweinsteiger	10.28	**13.17**	**17.32**	27.35
Özil	7.54	4.34	10.05	22.62
Podolski	4.91	0.22	6.66	**30.21**
Klose	0.92	0.00	2.48	14.34
Trochowski	3.00	0.00	2.85	**33.02**
Lahm	**10.60**	**11.83**	**14.65**	24.56
Mertesacker	**10.81**	3.42	13.27	26.71
Boateng	6.85	3.63	6.52	19.85

Table 4: Player scores for Germany.

un secinot ka tik tiešām, divi trekniem cipariem marķētie spēlētāji atbilstošajā kolonnā - Bastians Šveinšteigers un Filips Lāms - ir tādi par kuriem mājas darba autors, kas futbolam neseko un nekad nav sekojis līdzi, ir dzirdējis.

Pielikums

1. *Python skripts "elkoņa" meklēšanai pēc vienas iespējamas metodes*

```python
#!/usr/bin/python3
import sys
IFILE=sys.argv[1]

def curve(x1, x2, x3):
   prev = x2-x1
   nex = x3-x2
   return prev / nex

def curve_max(x_list):
   max = 0
   amax = 0
   for i in range(1, len(x_list)-1) :
       x1, x2, x3 = x_list[i-1:i+2]
       c = curve(x1, x2, x3)
       if i > 1 and not c > max:
           break
       amax = (i,x2) if c > max else amax
       max = c if c > max else max
       print(f"x = {x2} curve = {c}")
   print(f"max = {max}")
   print(f"amax = {amax}")

with open(IFILE, 'r') as f:
   xlist = [float(x) for x in f.readlines()]
curve_max(xlist)
```

2. *Logaritmisko ticamību dati - jāieliek tukšā failā, lai varētu izmantot skriptu*

seed = 100

-3.06427
2.67064
3.38786
6.37105
6.36237
5.87855
7.87008

seed = 246236

-3.06427
2.66793
3.38786
6.37105
6.36237
7.19342
7.63957