Latvijas Universitāte
Datorikas fakultāte
Maģistra studiju programma
Datizraces algoritmi
Jānis Ratnieks
st. apl. nr. jr09103

2. mājas darbs

1. Uzrakstiet vismaz 1000 zīmes garu sacerējumu: a) par Apriori algoritma pamatideju; b) par to,kādam nosacījumam ir jāizpildās, lai lielā datu tabulā asociāciju meklēšana varētu notikt ātri?

Apriori algoritms ir viens no populārākajiem asociāciju meklēšanas algoritmiem pasaulē, tas ir relatīvi vienkāršs, pat triviāls, un tādēļ to vienkāršiem vārdiem centīšos aprakstīt un noskaidrot apstākļus pie kuriem tas strādās ātri.
Apriori algoritma galvenā mēraukla ir atbalsts (*support)* un to lieto, lai atlasītu derīgās datu kolonnas no nederīgajām. Apskatām datu matricu ar m rindām un b kolonnām. Atbalstu rēķinā kā pozitīvo mērījumu rezultātu pret visiem mērījumiem, jeb $supp(x)=a/b$, kur a – pozitīvo mērījumu rezultāts un b – visu mērījumu skaits. Ja atbalsta slieksni pieņem piemēram 50%, tad visi dati, kuri parādās mazāk nekā 50% gadījumu tiek atmesti, jo nekad neveidos asociācijas un tādēļ var tikt atmesti. Nākamais solis ir apskatīt atlikušo (tādu, kuras var veidot asociācijas) datu kolonnu pārus. Šeit jāsaprot, ka ja kolonnas A un B ir derīgas, tad jāapskata tikai viens pāris AB vai BA, jo tie ir identiski. Ja atlikušās datu kolonnas ir m, tad kopējo pāru skaitu var atrast pēc vispārīgās formulas $\frac{n!}{(n-k)!}$, kur n=m un k=2, ja jāmeklē pāri. Tālāk skatās, kurš no pāriem pārsniedz 50% soli un meklē trijniekus pāriem, kuri pārsnieguši 50% barjeru. Šeit var ņemt vērā tikai tās datu rindas, kuras parādās kādā no derīgajiem pāriem, jo pārējās būs nederīgo pāru apakškopas. Šajā gadījumā izmantojot vispārīgo formulu k=3. Tā turpina ar četriniekiem, pieciniekiem utt. līdz tiek sasniegts brīdis, kad nākamo instanci vairs nevar izveidot ar pietiekamu atbalstu.
Uzreiz redzams, ka šāds algoritms ņems vērā tikai vispopulārākās datu kolonnas un nemeklēs sakarības starp mazāk popuārajām, kaut arī tājās var būt tāds, kuras izpildās ļoti bieži, piemēram, nopērkot vienu nepopulāru preci gandrīz viennozīmīgi tiks nopirkta arī cita prece. Lai varētu "noķert" arī šādas sakarības, ir izdomāti papildus mērs, kā ticamība (confidence), bet var būt arī citi, piemēram lift un cont. Ticamību aprēķina kā $conf(x \rightarrow y)=\frac{supp(x \cup y)}{a}$ jeb cik daudz reižu parādās y, kad ir x pret kopējo x pozitīvo vērtību skaitu a. Šādi var samazināt atbalsta slieksni uz piemēram 1%, bet ņemt vērā tikai tās asociācijas, kuru ticamība ir liela, piemeram 90%. Apriori algoritma būtība no tā nemainās.
Uzreiz var manīt, ka, ja atbalsta slieksnis vai ticamība ir ņemta pārāk maza, tad būs ļoti daudz datu kolonnas ar kurām jādarbojas un pie lieliem k (permutāciju formulā) apskatāmo gadījumu skaits tieksies uz n!. Lai algoritms strādātu ātri, matricai jābūt maz aizpildītai (sparse). Šādā gadījumā datu kolonnu atmešana notiks ātri un algoritms līdz ar to strādās ātri.

2. Atveriet WEKA failu *credit-g*. Izmantojiet algoritmu J48 (C4.5) divos režīmos: a) uz visiem dotajiem datiem kā uz treniņa kopas un b) sadalot dotos datus treniņa un testa kopās. Paeksperimentējiet ar parametriem *minObjNum*=1, 2, ... un *unpruned=false/true*. Kādus variantus izmēģinājāt, kādi sanāca lēmumu koka izmēri un kļūdu procenti? Ko no tā varat secināt? [WEKA vietā varat izmantot citu rīku.]

a) Izmantojot J48 algoritmu sākotnēji klasifikācijas uzdevumu veicu visiem datiem, bez treniņa kopas. Iegūtie rezultāti redzami zemāk un to precizitāte ir 85.5%.

Correctly Classified Instances	855	85.5 %
Incorrectly Classified Instances	145	14.5 %
Kappa statistic	0.6251	
Mean absolute error	0.2312	
Root mean squared error	0.34	
Relative absolute error	55.0377 %	
Root relative squared error	74.2015 %	
Total Number of Instances	1000	

---

a	b	<-- classified as
669	31	a = good
114	186	b = bad

b) Palaižot algoritmu ar dažādām *training set* un *data set* vērtībām, precizitāte samazinās visos gadījumos.

Training set size	precision
80%	77%
75%	76%
66%	72.6%

Tas ir saprotams, jo iepriekš modelis tika kalibrēts tikai uz dotajiem datiem un tika iegūta lielākā iespējamā precizitāte dotajiem datiem. Kad modeli tiek būvēts uz mazākas treniņkopas un pārbaudīts uz vēl neredzētiem datiem, kādi ir *test set*, tad skaidrs, ka precizitāte kritīsies. Šajā gadījumā redzams, ka ņemot lielāku treniņkopu, rezultāts ir ar augstāku precizitāti, to pārbaudot uz vēl neredzētiem datiem, kas liek domāt, ka uzkrājot vēl datus ir iespējams modeli padarīt precīzāku, tomēr tas nav viennozīmīgi, jo palaižot ar 90% treniņkopu, precizitāte nokrītās līdz 74%.

Pielietojot minNumObj=1 opciju un palaižot lēmumu koku visiem datiem, tiek iegūta augtāka precizitāte – 88.6%, bet sadalot treniņkopās un testa kopās 80% un 66% abos gadījumos precizitāte samazinās līdz attiecīgi 75% un 70%. Zaru un lapu skaits šajā gadījumā, salīdzinot ar minNumObj=2 ir ievērojami lielāks. Parametrs minNumObj nosaka mazāko iespējamo skaitu, ko katrs sazarojums drīkst atdalīt. Tas ir saprātīgs ierobežojums, jo aizliedz tā saucamo “overfitting” jeb pārkalibrēšanu, tiesa, uz datiem, kuriem nav testa kopas pārkalibrēšana uzrāda labāku rezultātu.

Opciju *unpruned=false/true* nomainot uz True, iegūst vēl lielāku koku, vēl lielāku precizitāti tikai treniņkopas datiem un vēl sliktāku precizitāti testa datiem. Šis parametrs ļauj “apgriezt” koka zarus, ja tie ievērojami neietekmē precizitāti un ļauj izvairīties no pārkalibrēšanas. Papildus tas padara lēmumu koku cilvēkiem vieglāk saprotamu.