Analiza pohitritve algoritma gručenja DBSCAN z lokalno-občutljivim zgoščevanjem

Niko Lukač, Borut Žalik, Domen Mongus, Domen Kavran, David Jesenko, Štefan Kohek

Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko,
Koroška cesta 46, 2000 Maribor, Slovenija.
E-pošta: {niko.lukac, borut.zalik, domen.mongus, domen.kavran1, david.jesenko, stefan.kohek}@um.si

Analysis of DBSCAN clustering algorithm's speedup by using locality-sensitive hashing

In this paper an analysis of the improved DBSCAN (Density-based spatial clustering of applications with noise) algorithm is performed using Locality-sensitive hashing based on random geometric projections. The paper firstly provides theoretical background of DBSCAN and DBSCAN-LSH algorithms. Then results of the experiments follow by using well known clustering benchmark datasets from UCI repository in various size and number of dimensions, where performance of DBSCAN-LSH is evaulated in terms of accuracy using well established clustering algorithms' evaluation indices. Lastly, the achieved computational speedup of using DBSCAN-LSH is evaluated.

1 Uvod

Dandanes gručenje podatkov spada med najbolj reprezentativne tehnike nenadzorovanega učenja za iskanje skritih vzorcev. V zadnjih desetletjih so bili predstavljeni različni algoritmi gručenja, ki jih v širšem delimo v hierarhične in particijske. Hierarhični algoritmi zgradijo hierarhijo nad podatki v različne nivoje gruč, pri čemer je gradnja lahko delitvena ali graditvena (t.i. aglomerativna). Znani predstavnik hierarhijskih algoritmov gručenja je algoritem CHAMELEON [1], ki večje gruče razdeli v manjše do nekega nivoja delitve. Particijski algoritmi, kjer je znani predstavnik algoritem K-means [2], sestavijo particije nad podatki, ki jih nato iterativno premikajo. Končne particije predstavljajo dobljene gruče. Pomankljivosti particijskih algoritmov so običajno vnaprej določeno število iskanih gruč, konvergenca algoritma v lokalnih minimumih ali slaba inicializacija začetnih particij. Večja pomankljivost hierarhičnih algoritmov pa je iskanje optimalnega izhodnega kriterija za ustavitev gradnje hierarhije. Tretja oblika gručenja temelji na relativni gostoti v podatkih, ki jo dosežemo na podlagi lokalne soseščine. Pri tem je znani predstavnik algoritem DBSCAN [3] (angl. Density-based spatial clustering of applications with noise), ki omogoča iskanje gruč brez definiranja števila iskanih gruč, ter hkrati omogoča boljšo robustnost do osamelcev in prisotnega šuma v več-dimenzionalnih podatkih.

V splošnem gručenje temelji na povezovanju najbližjih entitet v podatkih, ki jih lahko definiramo kot D dimenzionalne točke v evklidskem prostoru. Pri tem se lahko uporabi poljubna metrika razdalje, kjer je najpogostejša evklidska razdalja. Naivna implementacija bi zahtevala iskanje razdalje vsake točke z vsako drugo točko, s čimer bi dosegli kvadratno časovno zahtevnost $O(n^2)$ za n točk. V zadnjih dveh desetletjih so se začele uporabljati bolj učinkovite podatkovne strukture za deljenje podatkov, kot so KD-drevo in R-drevo [4], ki praviloma omogočijo iskanje najbližjih točk s časovno zahtevnostjo $O(nlog(n))$. Težava nastopi pri višje dimenzionalnih prostorih, zaradi t.i. prekletstva dimenzionalnosti [5]. V zadnjem desetletju so se zato začele razvijati in uporabljati bolj učinkovite podatkovne strukture, ki temeljijo na aproksimativnih algoritmih za iskanje soseščine ter hkrati dosegajo linearno časovno zahtevnost za n točk. Ena izmed vodilnih metod na tem področju je lokalno-občutljivo zgoščevanje (angl. locality sensitive hashing, LSH) [5], ki preslika višjedimenzionalni prostor v nižjega ter tako omogoča aproksimativno iskanje najbližjih točk. Tako so se tudi razvili aproksimativni algoritmi za gručenje, ki temeljijo na metodi LSH, kot na primer različice K-means LSH [6] in DBSCAN-LSH [7, 8, 9].

Glavna pomankljivost algoritmov gručenja temelječih na LSH je korektna določitev prostih parametrov za doseganje visoke natančnosti pri krajšem času izvajanja, zaradi aproksimativnega iskanja. S povečanjem števila zgoščevanj lahko tudi dosežemo višjo natančnost, vendar posledično dobimo daljši čas izvajanja. V danem prispevku predstavimo rezultate delne občutljivostne analize izbire prostih parametrov algoritma DBSCAN-LSH na podlagi LSH, ki temelji na naključnih geometrijskih projekcijah, ter analiziramo vpliv na natančnost in hitrost izvajanja. Pri tem DBSCAN-LSH primerjamo z osnovnim algoritmom DBSCAN nad izbranimi večdimenzionalnimi podatkovnimi množicami, kjer upoštevamo znane metrike za računanje natančnosti algoritmov gručenja.

Prispevek je sestavljen iz treh poglavij. V naslednjem poglavju predstavimo teoretično ozadje algoritmov DBSCAN in DBSCAN-LSH. V tretjem poglavju predstavimo rezultate delne občutljivostne analize. V zadnjem poglavju podamo zaključek.

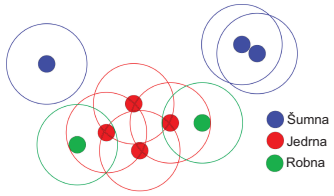
2 Teoretično ozadje

V naslednjem podpoglavju najprej podamo opis splošnega algoritma DBSCAN, nato v drugem podpoglavju sledi opis izboljšave na podlagi LSH.

2.1 Algoritem DBSCAN

DBSCAN klasificira vhodne točke podatkov v naslednje kategorije (glej sliko 1), na podlagi dveh prostih parametrov ϵ in $minPts$:

- **Jedrna točka**: je točka, ki ima vsaj $minPts$ sosednjih točk, ki so oddaljene od dane točke za ϵ ali manj.
- **Robna točka**: je točka, ki ni jedrna, vendar je oddaljena od jedrne točke za ϵ ali manj.
- **Šumna točka**: je točka, ki ni jedrna ali robna. Dane točke se zavržejo in ne pripadajo gručam.



Slika 1: Ilustracija treh kategorij vhodnih točk pri algoritmu DBSCAN.

Psevdokoda 1 prikazuje implementacijo algoritma DBSCAN. Algoritem obišče vsako neobiskano točko, ter preveri njeno soseščino s t.i. iskanjem relativne okolice (angl. range search) z radijem ϵ (vrstice 3-6). Pri tem si sproti vodimo interno polje OB obiskanih točk. V primeru, da je v njeni okolici vsaj $minPts$ dosegljivih točk, se definira kot jedrna točka, sicer kot šumna (vrstice 7-9). V primeru, da se gre za jedrno točko, uvedemo novo gručo (vrstica 15). Nato razširimo okolico jedrne točke s t.i. propagacijo okolice skozi ϵ-dosegljive točke in njihovih okolic (vrstice 11-18). V dani propagaciji točke označimo kot obiskane in jim dodelimo kategorijo robnih točk, hkrati pa dane točke pripadajo novi gruči od trenutne jedrne točke.

2.2 Algoritem DBSCAN-LSH

Algoritem DBSCAN-LSH poteka podobno kot izvorna različica algoritma, kjer pa je največja sprememba pri iskanju ϵ-okolice, saj je le to pohitreno z metodo LSH. Osnova LSH je konstruiranje zgoščevalne funkcije h, pri uporabi katere se vhodne točke razporedijo v vedra (angl. buckets), ki jih sestavlja zgoščevalna tabela. Za dano okolico ϵ med točkama q_1 in q_2 velja naslednja trditev [5]:

$$P[h(q_1) = h(q_2)] \geq P_1 \; pri \; d(q_1, q_2) \leq \epsilon, \quad (1)$$

$$P[h(q_1) = h(q_2)] \leq P_2 \; pri \; d(q_1, q_2) \geq c\epsilon, \quad (2)$$

kjer je d evklidska razdalja, ter P_1 verjetnost, da dani dve točki pripadata istem jedru po zgoščevanju s funkcijo h glede na njuno ϵ okolico. Obratno obstaja verjetnost P_2 (kjer velja $P_2 < P_1$), da točki pripadata istemu

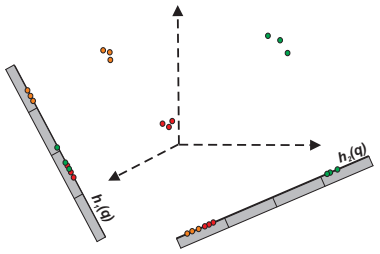
```
   Vhod: P, ε, minPts
   Izhod: C
1  C = {}, N = {}, OB = {}
2  foreach q ∈ P do
3      if q ∈ OB then
4          continue
5      OB = OB ∪ {q}
6      O_q = range_search(q, P, ε)
7      if |O_q| < minPts then
8          N = N ∪ {q} // Šumna točka
9          continue
10     NC = {q} // Nova gruča.
11     foreach p ∈ O_q do
12         if p ∈ OB then
13             continue
14         OB = OB ∪ {p}
15         NC = NC ∪ {p} // Pripada novi gruči
16         O_p = range_search(p, P, ε)
17         if |O_p| ≥ minPts then
18             O_q = O_q ∪ O_p // Razširimo okolico
19     C = C ∪ NC
```

Psevdokoda 1: Algoritem gručenja DBSCAN.

jedru, kljub c-krat večji oddaljenosti od ϵ. $c \in [0, \infty]$ je poljubno izbrana konstanta. Zgoščevalna funkcija, ki omogoča takšni dve verjetnosti je t.i. lokalno-občutljiva. V tem prispevku se bomo osredotočili na družino funkcij LSH na podlagi naključnih geometrijskih projekcij višjedimenzionalnega prostora v nižjega z uporabo operacije skalarnega produkta. Tako h definiramo kot [10]:

$$h^{(\mathbf{a},b)}(q) = \left\lfloor \frac{\mathbf{a} \cdot q + b}{w} \right\rfloor, \quad (3)$$

kjer je $\mathbf{a}$ naključni vektor iz naravne porazdelitve števil, b je naključna vrednost v območju $[0, w]$, w pa je širina vedra (tj. ločljivost kvantizacije). Primer uporabe dveh zgoščevalnih funkcij LSH v 3D prostoru je ilustriran na sliki 2. Točke, ki se nahajajo v istem vedru so t.i.



Slika 2: Ilustracija uporabe dveh zgoščevalnih funkcij LSH na podlagi naključne geometrijske projekcije v 3D prostoru.

aproksimativni sosedi. Za definiranje ekstaktne soseščine nato izračunamo eksaktno evklidsko razdaljo med aproksimativnimi sosedi. Kot je razvidno na sliki 2, lahko imata dve različni zgoščevalni funkciji različni rezultat zgoščevanja. Morebitne težave nastanejo, če točke, ki niso eksaktni sosedi, se nahajajo v istem vedru (t.i. kolizija zgoščevanja) ali v sosednjem (t.i. zgrešena projekcija). Zaradi danih dveh težav se v praksi uporablja več naključnih zgoščevalnih funkcij za doseganje večje natančnosti.

V tem prispevku uporabimo metodo E2LSH [11], ki temelji na dvostopenjskem zgoščevanju, ter je v praksi prikazala dobre rezultate glede minimiziranja kolizij [11]. Najprej definiramo množico L_1 zgoščevalnih funkcij $h(q)$, kjer vsaki določimo naključna parametra $\mathbf{a}$ in b,

Tabela 1: Rezultati algoritma DBSCAN nad izbranimi podatkovnimi zbirkami repozitorija UCI.

Zbirka	# dimenzij	# vzorcev	# razredov	ϵ	$minPts$	RI	Jaccard	ARI	NMI
smile	2	1000	4	0,05	5	1	1	1	1
shuttle	9	46464	5	0,02	6	0,8679	0,8266	0,5133	0,6553
iris	4	150	3	0,09	4	0,8970	0,7043	0,7544	0,7542
wine	13	178	3	0,36	5	0,7410	0,4926	0,4556	0,5968
ionosphere	34	351	2	0,75	6	0,8527	0,7614	0,7033	0,5792
seeds	7	210	3	0,20	3	0,7210	0,4631	0,4148	0,4909
glass	9	214	7	0,08	2	0,6515	0,1981	0,0952	0,2909

w pa je prosto nastavljiv parameter. Nato skonstruiramo L_2 zgoščevalnih funkcij $g(q)$, kjer vsaka funkcija $g(q)$ sestavlja L_3 ($L_3 \leq L_1$) unikatnih $h(q)$ funkcij:

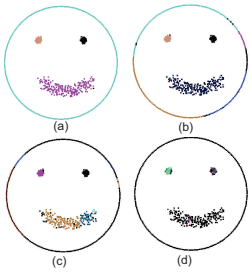
$$g(q) = \sum_{i=1}^{L_1} h_i(q). \tag{4}$$

Dani način zgoščevanja LSH omogoča višjo robustnost pred zgrešeno projekcijo ali kolizijami. Prosti parametri so tako L_1, L_2, L_3 in w. Pri algoritmu DBSCAN-LSH najprej izvedemo zgoščevanje vseh točk s funkcijami g. Nato pri preiskovanju soseščine posamezne točke q upoštevamo le aproksimativne sosede, do katerih še dodatno izračunamo eksaktno evklidsko razdaljo ter preverimo oddaljenost glede na parameter ϵ. Prav tako uporabimo optimizacijo t.i. metode DBSCAN++ [12], kjer ne obiščemo vseh točk, ampak le ožji reprezentativni izbor točk. V tem prispevku to dosežemo, če upoštevamo eno točko za vsako vedro v $g(q)$, torej število začetnih obiskovanih točk (in posledično maksimalno št. jedrnih točk) je odvisno od števila veder vseh funkcij g (tj. vrstica 2 v psevdokodi 1). Pri tem predpostavimo, da vedra predstavljajo aproksimativne pod-gruče realnih gruč. V prispevku smo prav tako upoštevali t.i. optimizacijo multiprobe-LSH [13], kjer pri definiranju aproksimativnih sosedov znotraj posameznega vedra preiščemo še neposredni sosednji vedri v zgoščevalni tabeli, saj s tem zmanjšamo učinek zgrešene projekcije pri manjših L_2 in L_3.

3 Rezultati

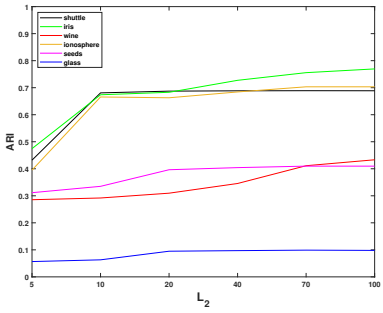
Za izvedbo analize parametrov DBSCAN-LSH smo izbrali različne podatkovne zbirke iz repozitorija podatkov za strojno učenje UC Irvine (UCI)[1], ki so predstavljeni v tabeli 1. Dani repozitorij je bil izbran zaradi omogočanja lažje reproduktivnosti in primerjavo rezultatov. Vrednosti v tabeli predstavljajo povprečni rezultat po 10-kratni ponovitvi algoritma nad vsako zbirko. V dani tabeli so prav tako predstavljeni rezultati DBSCAN z izbranima ϵ in $minPts$. Vhodne podatke smo najprej normalizirali z max-min normalizacijo pred izvedbo gručenja. Za meritev natančnosti smo uporabili znane metrike natančnosti algoritmov gručenja kot so Rand (angl. Rand Index, RI), Jaccard, prilagojen Rand (angl. Adjusted Rand Index, ARI), in normalizirana skupna informacija (angl. Normalized Mutual Information, NMI) [14]. Metriki kot sta Davies-Bouldin in Silhuete nismo vključili, saj podatki vključujejo tudi nekonveksne in neglobularne gruče.

[1] https://archive.ics.uci.edu



Slika 3: Ilustracija DBSCAN-LSH nad zbirko smile, pri čemer $L_3 = 3$ ter (a) $L_2 = 5$, (b) $L_2 = 3$, (c) $L_2 = 2$ in (d) $L_2 = 1$.

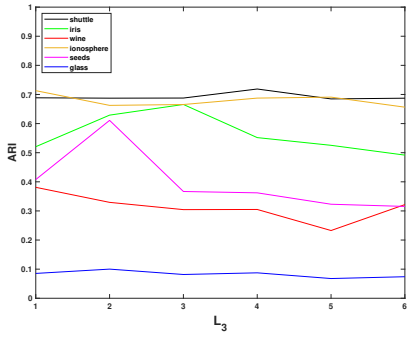
V vseh eksperimentih smo predpostavili $L_1 = 1000$ zgoščevalnih funkcij h ter $w = \epsilon/10$. Zbirko smile smo zgolj vključili za vizualno ilustracijo delovanja DBSCAN-LSH (glej sliko 3). Kot je razvidno, natančnost hitro upada z zmanjšanjem št. zgoščevalnih funkcij g. Na slikah 4 in 5 so prikazani rezultati natančnosti po metriki ARI, pri čemer smo spreminjali parameter L_2 pri $L_3 = 3$ ter parameter L_3 pri $L_2 = 15$.



Slika 4: Natančnost DBSCAN-LSH po metriki ARI, pri spreminjanju L_2 pri $L_3 = 3$.

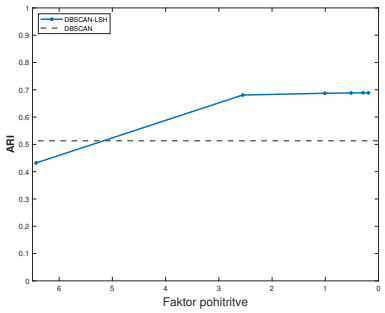
Glavna prednost DBSCAN-LSH je v času iskanja soseščine pri višjedimenzionalnih zbirkah podatkov. Za primerjavo časa izvajanja smo uporabili centralno procesno enoto AMD Ryzen 9 3900X z 12 jedri ter upoštevali večnitno implementacijo obeh algoritmov. Za dano primerjavo smo izbrali zbirko shuttle, ki ima zadostno št. vzorcev. Pri večnitni implementaciji DBSCAN-LSH smo paralelno izračunali zgoščevanje s funkcijami g. Pri osnovnem DBSCAN pa smo izvedli paralelno prostorsko indeksiranje s podatkovno strukturo KD-drevesa na podlagi knjižnice nanoflann[2]. Pri zagonu obeh algoritmov smo uporabili 24 niti. DBSCAN je v povprečju (po 10 zagonih) dosegal čas izvajanja 2876 ms za dano zbirko.

[2] https://github.com/jlblancoc/nanoflann



Slika 5: Natančnost DBSCAN-LSH po metriki ARI, pri spreminjanju L_3 pri $L_2 = 15$.

Slika 6 prikazuje odvisnost izračunanega faktorja pohitritve (tj. $2876/t$, kjer je t [ms] čas izvajanja pri DBSCAN-LSH) od metrike ARI, pri primerjavi med algoritmoma DBSCAN-LSH in DBSCAN.



Slika 6: Razmerje med faktorjem pohitritve in metriko ARI pri primerjavi DBSCAN-LSH z DBSCAN, pri čemer spreminjamo $L_2 = \{5, 10, 20, 40, 70, 100\}$ ter ohranimo $L_3 = 3$ (meritve si sledijo od leve proti desni).

4 Zaključek

V danem prispevku smo predstavili delno občutljivostno analizo parametrov (tj. število zgoščevalnih funkcij) algoritma DBSCAN-LSH, pri čemer smo se osredotočili na družino lokalno-občutljivih zgoščevalnih funkcij na podlagi geometrijskih projekcij. Rezultati večnitne implementacije prikazujejo možnost pohitritve v primerjavi s klasičnim algoritmom DBSCAN. Pri tem velja, da zaradi povečanja števila funkcij (za doseganje višje natančnosti) tudi hitro narašča čas izvajanja. V prispevku se nismo osredotočili na implementacijo na grafičnih procesnih enotah, ki bi omogočale še hitrejše izvajanje v doglednem času. Prav tako nismo analizirali vpliv parametra w ter upoštevali ostalih zgoščevalnih funkcij iz znanih družin LSH, kot je na primer Hamming-ov LSH, kar je delo za prihodnje.

Zahvala

Algoritem DBSCAN-LSH je bil razvit in preizkušen na platformi STALITA v okviru ciljnega raziskovalnega programa V2-2260, ki sta ga sofinancirala Javna agencija za znanstvenoraziskovalno in inovacijsko dejavnost Republike Slovenije ter Ministrstvo za notranje zadeve. Raziskovalni program št. P2-0041 je sofinancirala Javna agencija za znanstvenoraziskovalno in inovacijsko dejavnost Republike Slovenije.

Literatura

[1] G. Karypis, E. H. Han, V. Kumar: Chameleon: Hierarchical clustering using dynamic modeling. Computer, 32 (8), str. 68-75, 1999.

[2] J. MacQueen: Classification and analysis of multivariate observations. 5th Berkeley Symp. Math. Statist. Probability, str. 281-297, 1967, Los Angeles, ZDA.

[3] M. Ester, H. P. Kriegel, J. Sander, X. Xu: A density-based algorithm for discovering clusters in large spatial databases with noise. 2nd International Conference on Knowledge Discovery and Data Mining, 96 (34), str. 226-231, 1996, Portland, ZDA.

[4] L. Balasubramanian, M. Sugumaran: A state-of-art in R-tree variants for spatial indexing. International Journal of Computer Applications, 42 (20), str. 35-41, 2012.

[5] P. Indyk, R. Motwani, P. Raghavan,S. Vempala, S:. Locality-preserving hashing in multidimensional spaces. 29TH annual ACM symposium on Theory of computing, str. 618-625, 1997, Texas, ZDA.

[6] L. Paulevé, H. Jégou, L. Amsaleg: Locality sensitive hashing: A comparison of hash function types and querying mechanisms. Pattern recognition letters, 31 (11), str. 1348-1358, 2010.

[7] A. Keramatian, V. Gulisano, M. Papatriantafilou, P. Tsigas: IP. LSH. DBSCAN: Integrated Parallel Density-Based Clustering Through Locality-Sensitive Hashing. Euro-Par 2022: Parallel Processing: 28th International Conference on Parallel and Distributed Computing, Glasgow, str. 22–26, 2022, UK.

[8] N. Lukač, D. Jesenko, M. Bizjak, B. Žalik: GPU-based DBSCAN clustering on locality sensitive hashing. 7th international conference of engineering and applied sciences (ICEAS 2017), str. 14-19, Toronto, Kanada, 2017.

[9] Y. Shiqiu, Z. Qingsheng. DBSCAN clustering algorithm based on locality sensitive hashing. In Journal of Physics: Conference Series, IOP Publishing, 1314 (1), str. 012177, 2019.

[10] A. Andoni, P. Indyk: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Communications of the ACM, 51 (1), str. 117-122, 2008.

[11] A. Andoni, M. Datar, N. Immorlica, P. Indyk, V. Mirrokni: Locality-sensitive hashing using stable distributions. Nearest-Neighbor Methods in Learning and Vision, str. 61-72, 2005.

[12] H. Jiang, J. Jang, J. Lacki: Faster DBSCAN via subsampled similarity queries. Advances in Neural Information Processing Systems, 33, str. 22407-22419, 2020.

[13] Q. Lv, W. Josephson, Z. Wang, M. Charikar, K. Li: Multi-probe LSH: efficient indexing for high-dimensional similarity search. 33rd international conference on Very large data bases, str. 950-961, 2007, Dunaj, Avstrija.

[14] A. Amelio, C. Pizzuti: Correction for closeness: Adjusting normalized mutual information measure for clustering comparison. Computational Intelligence, 33 (3), str. 579-601, 2017.