حسین بهشتی فرد / 9631011

تمرین دوم داده کاوی

بخش تئوری

---

1)

الف)

مرحله اول:

Datas		
	x	y
A1	1	2
A2	6	3
A3	8	4
A4	2	5
A5	7	5
A6	4	6
A7	5	7
A8	2	8

Centeroid		
	x	y
1	4	5.66
2	6	5
3	2.5	4

i	1	2	3
A1	4.732398969	5.830952	2.5
A2	3.328002404	2	3.640055
A3	4.330773603	2.236068	5.5
A4	2.106086418	4	1.118034
A5	3.071742177	1	4.609772
A6	0.34	2.236068	2.5

A7	1.672004785	2.236068	3.905125
A8	3.078246254	5	4.031129

مرحله دوم:

Datas		
	x	y
A1	1	2
A2	6	3
A3	8	4
A4	2	5
A5	7	5
A6	4	6
A7	5	7
A8	2	8

Centeroid		
	x	y
1	3.66	7
2	7	4
3	1.5	3.5

i	1	2	3
A1	5.663532467	6.324555	1.581139
A2	4.634177381	1.414214	4.527693
A3	5.275945413	1	6.519202
A4	2.599153708	5.09902	1.581139
A5	3.893019394	1	5.700877
A6	1.056219674	3.605551	3.535534
A7	1.34	3.605551	4.949747
A8	1.937937047	6.403124	4.527693

مرحله سوم:

Datas		
	x	y
A1	1	2
A2	6	3
A3	8	4
A4	2	5
A5	7	5
A6	4	6
A7	5	7
A8	2	8

Centeroid		
	x	y
1	3.66	7
2	7	4
3	1.5	3.5

همانطور که میبینیم centroid ها تغییر نکرد. بنابراین الگوریتم به پایان میرسد و خوشه های ما بصورت بالا

خواهد بود.

ب)

مرحله اول:

Datas		
	x	y
A1	1	2
A2	6	3
A3	8	4
A4	2	5
A5	7	5
A6	4	6

A7	5	7
A8	2	8

i	1st cluster medoid(1, 2)	2nd cluster medoid(6, 3)	3rd cluster medoid(8, 4)
A1	-	-	
A2	-	-	
A3	-	-	
A4	$\|2-1\| + \|5-2\| = 4$	$\|2-6\| + \|5-3\| = 6$	$\|2-8\| + \|5-4\| = 7$
A5	$\|7-1\| + \|5-2\| = 9$	$\|7-6\| + \|5-3\| = 3$	$\|7-8\| + \|5-4\| = 2$
A6	$\|4-1\| + \|6-2\| = 7$	$\|4-6\| + \|6-3\| = 5$	$\|4-8\| + \|6-4\| = 4$
A7	$\|5-1\| + \|7-2\| = 9$	$\|5-6\| + \|7-3\| = 5$	$\|5-8\| + \|7-4\| = 6$
A8	$\|2-1\| + \|8-2\| = 7$	$\|2-6\| + \|8-3\| = 9$	$\|2-8\| + \|8-4\| = 10$

Total cost = 4 + 2 + 5 + 5 + 7 = 23

Datas		
	x	y
A1	1	2
A2	6	3
A3	8	4
A4	2	5
A5	7	5
A6	4	6
A7	5	7
A8	2	8

مرحله دوم:

حال مرکز خوشه اول را به A4 تغییر میدهیم.

i	1st cluster medoid(2, 5)	2nd cluster medoid(6, 3)	3rd cluster medoid(8, 4)
A1	$\|1-2\| + \|2-5\| = 4$	$\|1-6\| + \|2-3\| = 6$	$\|1-8\| + \|2-4\| = 9$
A2	-	-	-
A3	-	-	-
A4	-	-	-

A5	$\|7-2\| + \|5-5\| = 5$	$\|7-6\| + \|5-3\| = 3$	$\|7-8\| + \|5-4\| = 2$
A6	$\|4-2\| + \|6-5\| = 3$	$\|4-6\| + \|6-3\| = 5$	$\|4-8\| + \|6-4\| = 4$
A7	$\|5-2\| + \|7-5\| = 5$	$\|5-6\| + \|7-3\| = 5$	$\|5-8\| + \|7-4\| = 6$
A8	$\|2-2\| + \|8-5\| = 3$	$\|2-6\| + \|8-3\| = 9$	$\|2-8\| + \|8-4\| = 10$

$$\text{Total cost} = 4 + 2 + 3 + 5 + 3 = 17$$

همانطور که میبینیم total cost کمتر شد، بنابراین جابجایی را انجام می‌دهیم.

Datas		
	x	y
A1	1	2
A2	6	3
A3	8	4
A4	2	5
A5	7	5
A6	4	6
A7	5	7
A8	2	8

مرحله سوم:

حال مرکز خوشه سوم را به A5 جابه‌جا می‌کنیم.

i	1st cluster medoid(2, 5)	2nd cluster medoid(6, 3)	3rd cluster medoid(7, 5)
A1	$\|1-2\| + \|2-5\| = 4$	$\|1-6\| + \|2-3\| = 6$	$\|1-7\| + \|2-5\| = 9$
A2	-	-	-
A3	$\|8-2\| + \|4-5\| = 7$	$\|8-6\| + \|4-3\| = 3$	$\|8-7\| + \|4-5\| = 2$
A4	-	-	-
A5	-	-	-
A6	$\|4-2\| + \|6-5\| = 3$	$\|4-6\| + \|6-3\| = 5$	$\|4-7\| + \|6-5\| = 4$
A7	$\|5-2\| + \|7-5\| = 5$	$\|5-6\| + \|7-3\| = 5$	$\|5-7\| + \|7-5\| = 4$
A8	$\|2-2\| + \|8-5\| = 3$	$\|2-6\| + \|8-3\| = 9$	$\|2-7\| + \|8-5\| = 8$

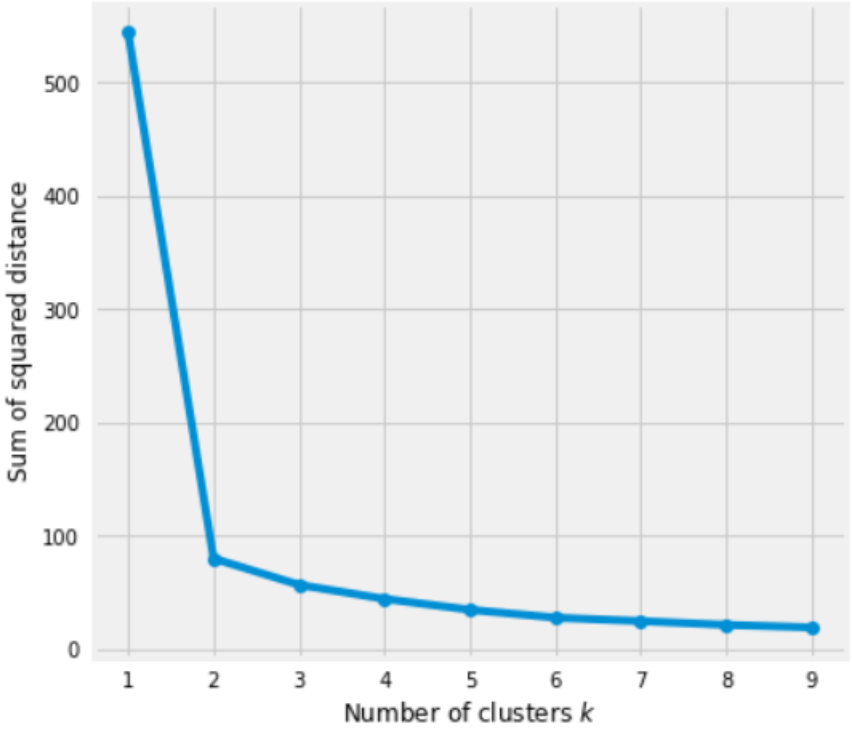
$$\text{Total cost} = 4 + 2 + 3 + 5 + 3 = 17$$

می‌بینیم که total cost تغییر نکرد درنتیجه این جابه‌جایی را انجام نمی‌دهیم.

2)

روش Elbow:

یکی از روش هایی است که برای پیدا کردن k مناسب استفاده می‌شود. در این روش هدف این است که k ای را پیدا کنیم که میزان SSE آن مناسب باشد و از آن شماره به بعد تغییرات چندانی نداشته باشیم. به این منظور برای k های مختلف مجموع فاصله مربعی(فاصله اقلیدوسی) محاسبه می‌شود، که معمولا نمودار این فاصله ها به ازای k بصورت یک بازو شبیه به شکل زیر می‌شود. حال k مناسب بر اساس ناحیه آرنج از جایی که تقریبا نمودار کم شیب و صاف می‌شود را انتخاب می‌کنیم.



در اینجا k=2 گزینه بدی نیست. هرچند که k های بعدی مجموع فاصله کمتری دارند اما هرچه تعداد k زیاد شود، مدل به سمت overfitting می‌رود و عملکرد مناسبی نخواهد داشت. در واقع همان بحث تریدآف است.

3)

الف)k-medoids

این الگوریتم شبیه به الگوریتم k-mean می‌باشد با این تفاوت که به جای میانگین از medoid استفاده می‌کنیم. در الگوریتم k-mean در هر مرحله برای تعیین centroid از میانگین دیتاهای خوشه استفاده می‌شد. اما در این الگوریتم از medoid آن ها استفاده می‌شود. که برخلاف k-mean، این نقاط حتما جز

خود دیتاها هستند. مزیت این الگوریتم این است که نسبت به outlier ها و نویز ها مقاوم تر است. در الگوریتم k-mean در صورت وجود outlier میانگین به شدت تغییر می‌کرد و به خوبی خوشه بندی انجام نمی‌شد اما medoid نسبت به outlier مقاوم تر است و در این موارد بهتر عمل می‌کند. اما پیچیدگی این الگویتم O(ik(n-k)2) می‌باشد که بسیار زیاد است و این الگوریتم نیز مقیاس پذیر نیست و برای دیتاست های بزرگ مناسب نیست. همینطور به علت رندم انتخاب شدن medoid ها می‌تواند نتایج متفاوتی را بدهد. الگوریتم k-medoid، np hard می‌باشد و الگوریتم PAM یک هیوریستیک برای آن است که از فاصله منهتن استفاده می‌کند و البته لزوما جواب بهینه را نمی‌دهد.

ب) CLARA

الگوریتم CLARA به اینصورت عمل می‌کند که در ابتدا تعدادی سمپل از مجموعه دیتا انتخاب می‌کند، سپس الگوریتم PAM را روی هر سمپل اجرا می‌کند و در آخر بهترین خوشه بندی را به عنوان خروجی می‌دهد.

مزیت این الگوریتم این است که برای دیتاست های بزرگ مناسب تر است.

اما معایب آن این است که کارامدی وابسته به سایز سمپل ها است. همچنین خوشه بندی ای که ارائه میشه بر اساس سمپل ها است نه کل دیتاست و اگر سمپل هایی که انتخاب می‌کنیم بایاس باشند و شبیه به هم باشند، خوشه بندی خوبی نخواهیم داشت.

ج) DBSCAN

الگوریتم DBSCAN یک الگوریتم چگالی بیس می‌باشد که می‌تواند خوشه بندی غیر خطی را داشته باشد. در این الگوریتم در ابتدا یک دیتا بصورت دلخواه انتخاب می‌شود، سپس با تمامی نقاط قابل دسترس بر اساس EPS(به عنوان شعاع دایره) ترکیب می‌شود. حال اگر این p حداقل تعداد MinPts که انتخاب شده است را داشته باشد یک core خواهد بود و به عنوان یک خوشه درنظر گرفته می‌شود. در غیراینصورت یک border است که جز point های قابل دسترس خواهد بود.

از مزیت های این الگوریتم همانطور که گفته شد می‌تواند خوشه بندی غیر خطی را انجام دهد که در k-mean و PAM نمی‌توانستیم. همینطور نیاز نیست که تعداد k را خودمان مشخص کنیم و صرفا دو متغیر EPS و MinPts انتخاب می‌شود و آن حالت رندم بودن الگوریتم های قبلی را ندارد. همینطور این الگوریتم به outlier ها حساس نیست و می‌تواند این نقاط را دیتا ها را نیز مشخص کند.

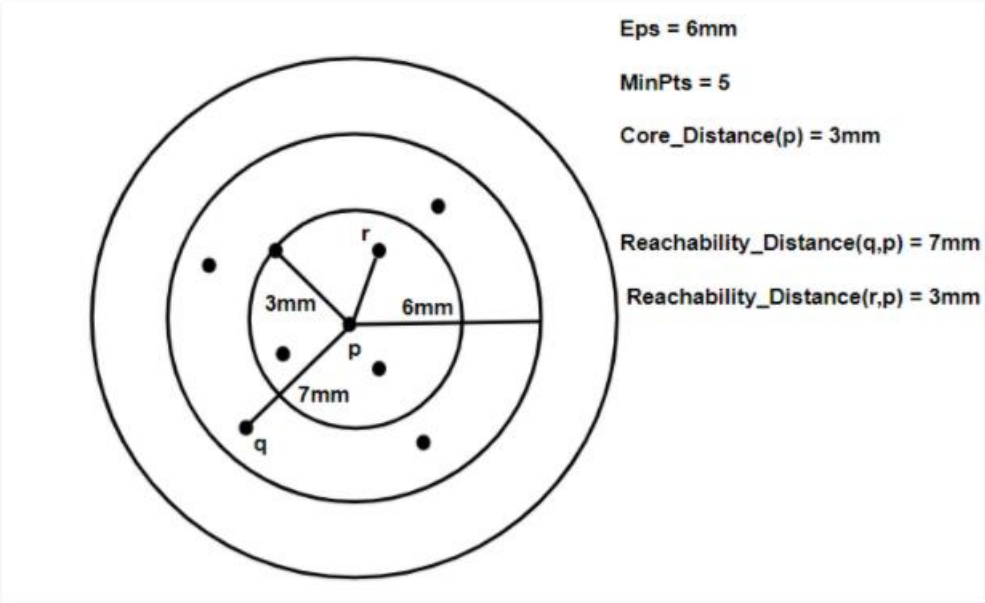
اما این الگوریتم براساس EPS که انتخاب می‌شود خوشه بندی را انجام می‌دهد و ممکن است در مواردی دیتا هایی که جز یک خوشه می‌توان در نظر گرفت را خوشه جدا و outlier درنظر بگیرد. همینطور زمانی که چگالی دیتا ها منظم نباشد ممکن است خوشه بندی به خوبی صورت نگیرد. از نظر پیچیدگی نیز ممکن است در دیتاست های بزرگ این محاسبه فاصله وقت‌گیر باشد.

د) OPTICS

مانند الگوریتم DBSCAN است با این تفاوت که دو پارامتر اضافه دارد که کمک کند خوشه بندی بهتر انجام گیرد و با تغییرات EPS خوشه بندی ها مناسب باشد. پارامتر Core distance که درواقع فاصله نقاط درون EPS از مرکز است. و reachability distance در واقع فاصله نقطه تا مرکز یک core است که بصورت زیر می‌باشند. که بر اساس این فاصله ها خوشه بندی را انجام می‌دهد و بررسی می‌کند که یک داده را جز خوشه قرار دهد یا خیر.

$$\text{core-dist}_{\varepsilon,MinPts}(p) = \begin{cases} \text{UNDEFINED} & \text{if } |N_\varepsilon(p)| < MinPts \\ MinPts\text{-th smallest distance in } N_\varepsilon(p) & \text{otherwise} \end{cases}$$

$$\text{reachability-dist}_{\varepsilon,MinPts}(o,p) = \begin{cases} \text{UNDEFINED} & \text{if } |N_\varepsilon(p)| < MinPts \\ \max(\text{core-dist}_{\varepsilon,MinPts}(p), \text{dist}(p,o)) & \text{otherwise} \end{cases}$$



از مزایای این الگوریتم این است دیگر آن مشکل DBSCAN را ندارد و به EPS وابسته نیست.

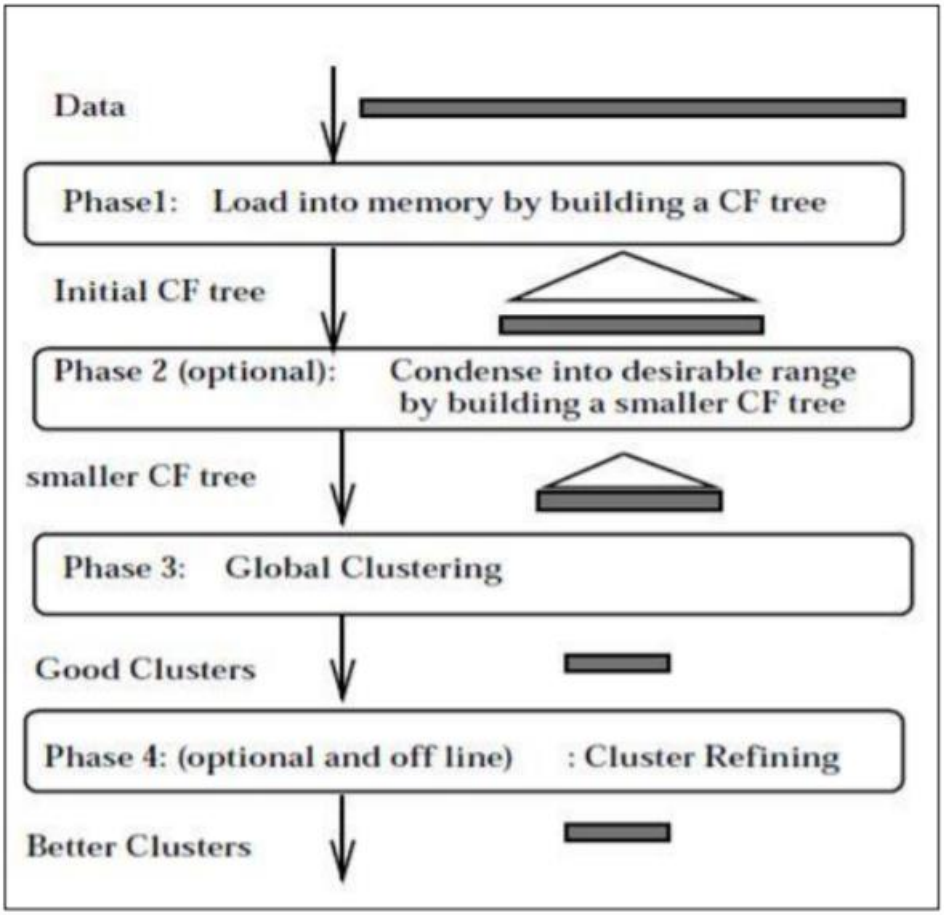
اما در اینجا نیز الگوریتم وابسته به ایم دو پارامتر تعریف شده می‌باشد. همینطور هزینه مموری بسیار زیاد است. این الگوریتم تنها برای داده های numeric می‌باشد.

ه) BIRCH

این الگوریتم جز الگوریتم های سلسله مراتبی است که بخصوص برای دیتاست های بزرگ مورد استفاده قرار می‌گیرد.

بطور کلی این الگوریتم در فاز اول با استفاده از توسعه افزایشی درخت ویژگی ها(clustering features) را می‌سازد. چون افزایش بصورت افزایشی است از حافظه اصلی می‌تواند استفاده کند که بسیار سریع‌تر می‌شود.

در فاز بعد با استفاده از درخت ساخته شده و برگ های آن، خوشه ها را بهتر و بهتر می‌کند.



مزایا:

از مزایای این الگوریتم همانطور که گفتیم برای دیتاست های بزرگ مناسب است و این الگوریتم مقیاس پذیر است. فقط یک بار دیتابیس را اسکن می‌کند و چون درخت را بصورت افزایشی می‌سازد، استفاده از ممووری بهینه است.

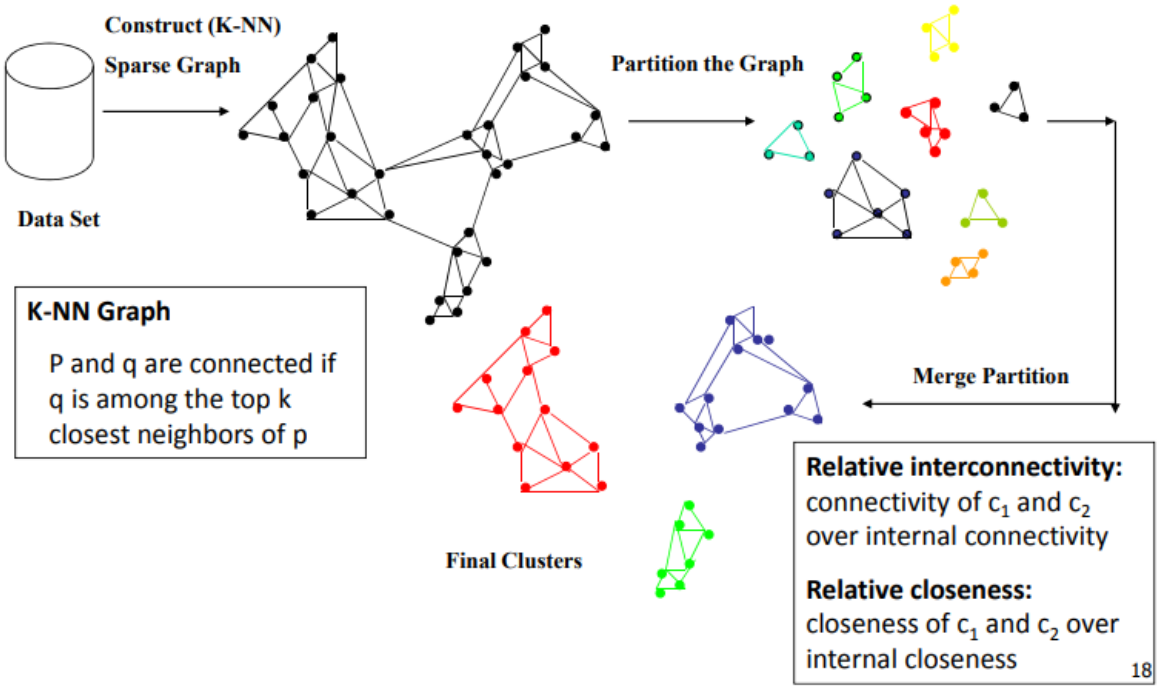
معایب:

این الگوریتم تنها برای داده های numeric می‌باشد. همینطور به ترتیب رکورد های دیتا حساس است.

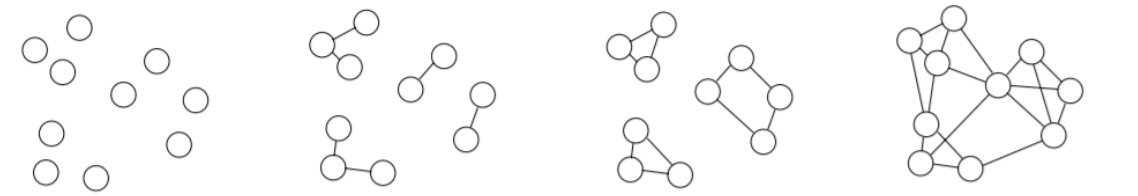
و) CHAMELEON

این الگوریتم نیز جز الگوریتم های سلسله مراتبی است که برای داده های شبیه به گراف می‌باشد.

روند کلی این الگوریتم بصورت زیر می‌باشد.



در ابتدا با استفاده از KNN Graph داده ها را بصورت یک گراف به یک دیگر merge می‌کنیم.



(a) Original Data in 2D (b) 1-nearest neighbor graph (c) 2-nearest neighbor graph (d) 3-nearest neighbor graph

K-NN graph بصورت بالا است که در هر مرحله به اینصورت با یکدیگر متصل می‌شوند.

سپس این گراف به زیر گراف های کوچک شکسته می‌شود. سپس بر اساس دو معیار Relative interconnectivity و Relative closeness با یکدیگر ادغام و خوشه ها را تشکیل می‌دهند.

فرمول Relative interconnectivity بصورت زیر می‌باشد که $EC_{\{ci, cj\}}$ برابر مجموع وزن یال های بین این دو نود می‌باشد.

$$RI(C_i, C_j) = \frac{|EC_{\{C_i,C_j\}}|}{\frac{|EC_{C_i}|+|EC_{C_j}|}{2}},$$

فرمول Relative closeness بصورت زیر است که صورت مخرج میانگین وزن یال های بین این دو نود و در مخرج نیز min-cut bisector دو خوشه C_i و C_j می‌باشد.

$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i,C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}},$$

مرتبه زمانی این الگوریتم برابر .O(nm + n log n + m2 log m)

مزیت ها:

می‌توان شکل های پیچیده را بخوبی خوشه بندی کند.

Incremental است.

معایب:

مرتبه زمانی برای ابعاد بالا $O(n^2)$ است.

4)

الف) درست: درواقع در این الگوریتم، تشکیل خوشه به شعاع EPS وابسته است. Point انتخاب و با نقاط density-reachable ترکیب شذه و اگر تعداد MinPts را داشته باشند، خوشه ها را تشکیل می‌دهند. درغیر این صورت border خواهند بود.

ب) درست: زیرا outlier ها نقاط پرت و دور هستند بنابراین وارد خوشه ها نمی‌شوند و همینطور این الگوریتم این داده ها را به عنوان outlier مشخص می‌کند.

ج) نادرست: در صورت استفاده نکردن از indexing بدترین حالت مرتبه $O(n^2)$ دارد. زیرا در بدترین حالت تمامی نقاط با یکدیگر مقایسه می‌شوند.

د) درست: فقط نیاز به مشخص کردن دو پارامتر EPS و MinPts دارد و بر اساس این دو پارامتر خود خوشه بندی را انجام می‌دهد.

5)

الف) این خوشه بندی را با الگوریتم های خوشه بندی خطی نمی‌توان انجام داد و k-mean نمی‌تواند اینگونه خوشه بندی کند. DBSCAN برای این مدل مناسب است.

ب) با هر دو می‌توان این خوشه بندی را انجام داد اما داده ها پراکنده هستند و چگالی در نقاط مختلف گونه ای است که درنتیجه DBSCAN به خوبی عمل نمی‌کند و احتمالا نمی‌تواند به این خوشه بندی مورد نظر برسد. در نتیجه k-mean مناسب تر است.

ج) در این دیتاست نیز از هر دو الگوریتم میشه استفاده کرد. اما چون داده های خوشه های قهوه ای و صورتی، همینطور قرمز و سبز به هم متصل هستند، الگوریتم DBSCAN این خوشه ها را ترکیب می‌کند. در نتیجه k-mean مناسب تر است.

د) این دیتاست را نمی‌توان با الگوریتم های خطی خوشه بندی کرد زیرا بر اساس میانگین و میانه و ... نمی‌توان به همچین خوشه‌بندی‌ای رسید. الگوریتم DBSCAN برای این مدل مناسب است.

6)

Single-Link:

در هر مرحله مینیموم فاصله را پیدا می‌کنیم، خوشه ها را ایجاد می‌کنیم و جدول را بر اساس خوشه های تشکیل شده آپدیت می‌کنیم.

برای آپدیت فاصله هر خوشه از مینیموم استفاده می‌کنیم بصورت زیر:

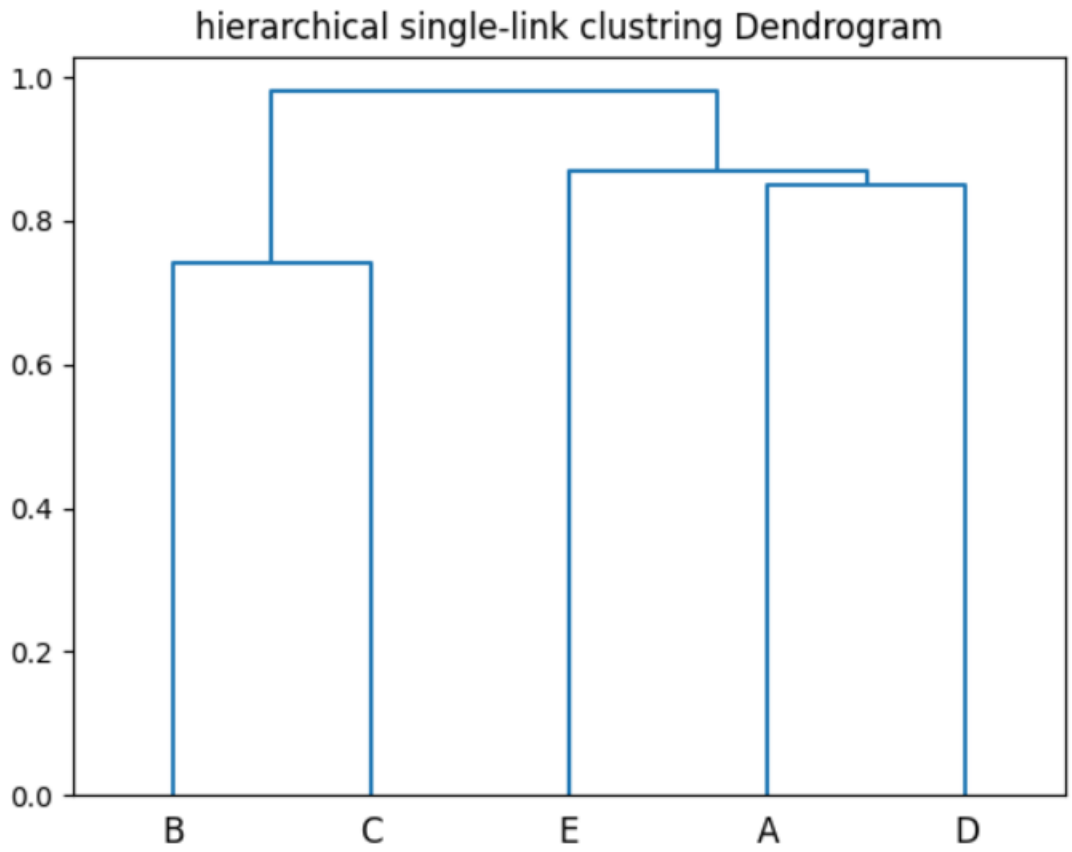
MIN[dist(X, Y), Z)] = MIN[dist(X, Z), (Y, Z)]

	A	B	C	D	E
A		1.23	2.44	0.85	2.04
B	1.23		0.74	1.2	0.98
C	2.44	0.74		1.34	1.4
D	0.85	1.2	1.34		0.87
E	2.04	0.98	1.4	0.87	

	A	B, C	D	E
A		1.23	0.85	2.04
B, C	1.23		1.2	0.98
D	0.85	1.2		0.87
E	2.04	0.98	0.87	

	A, D	B, C	E
A, D		1.2	0.87
B, C	1.2		0.98
E	0.87	0.98	

	A, D, E	B, C
A, D,		0.98
B, C	0.98	



Complete-Link:

در هر مرحله مینیموم فاصله را پیدا می‌کنیم، خوشه ها را ایجاد می‌کنیم و جدول را بر اساس خوشه های تشکیل شده آپدیت می‌کنیم.

برای آپدیت فاصله هر خوشه از ماکسیموم استفاده می‌کنیم بصورت زیر:

MAX[dist(X, Y), Z)] = MAX[dist(X, Z), (Y, Z)]

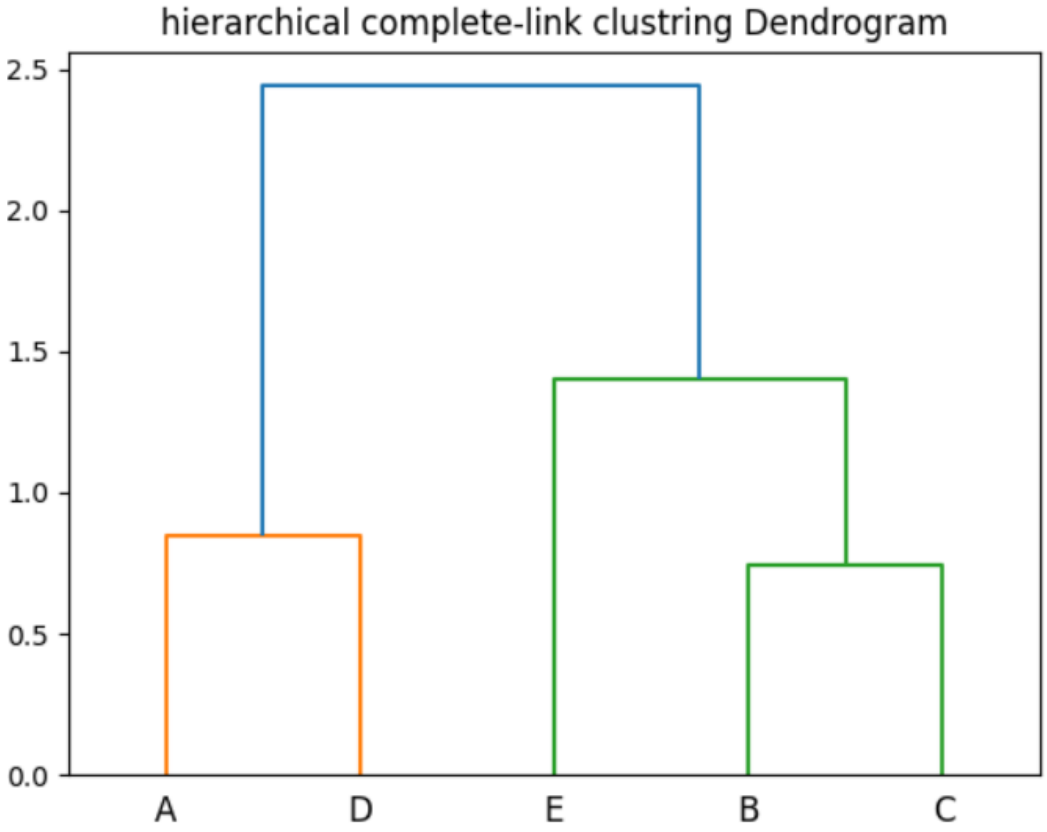
	A	B	C	D	E
A		1.23	2.44	0.85	2.04
B	1.23		0.74	1.2	0.98
C	2.44	0.74		1.34	1.4

D	0.85	1.2	1.34		0.87
E	2.04	0.98	1.4	0.87	

	A	B, C	D	E
A		2.44	0.85	2.04
B, C	2.44		1.34	1.4
D	0.85	1.34		0.87
E	2.04	1.4	0.87	

	A, D	B, C	E
A, D		2.44	2.04
B, C	2.44		1.4
E	2.04	1.4	

	A, D	B, C, E
A, D		2.44
B, C,	2.44	


hierarchical complete-link clustring Dendrogram
2.5
2.0
1.5
1.0
0.5
0.0
A
D
E
B
C