تینا غلامی ۹۵۳۱۳۰۷
دکتر ناظم فرد

HW 2 -

Question 1)

ایده‌ی اصلی الگوریتم "mini batch kmeans" این است که در هر iteration، از data، sample هایی به عنوان batch های رندم و کوچک ولی با سایز fix و مشخصی را انتخاب می‌کند تا آن‌ها در memory ذخیره کند. و در iteration بعدی، Sample های رندم جدیدی را انتخاب می‌کند و به همین ترتیب، در هر iteration، cluster update می‌شود تا در نهایت همگرایی (convergence) رخ دهد.

در واقع در این الگوریتم، کلاستر ها توسط ترکیب خطی و معدلی از prototype و data، update می‌شوند و یک learning rate که با افزایش تعداد #iteration ها کاهش می‌یابد، به آن‌ها اعمال می‌شود.

در واقع $\text{learning rate} = \frac{1}{\#\text{ data assigned to the clusters}}$ ⇐ به دلیل همگرایی با عدم تغییر در کلاستر ها رخ می‌دهد. همچنین با افزایش تعداد cluster ها و تعداد data های خوشه‌بندی شده، مقدار زمان صرف جویی شده در محاسبات نیز افزایش می‌یابد.

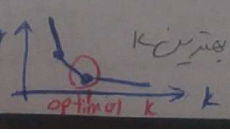
زمان runtime(s) در الگوریتم mini-batch kmeans نسبت به الگوریتم kmeans در هر iteration و با افزایش data های خوشه‌بندی شده، مشخصاً کمتر است. در واقع چون برای update کردن cluster ها از روش gradient descent استفاده می‌شود، الگوریتم سریعتری نسبت به روش kmeans خواهیم داشت.

در عین حال، مشکل الگوریتم mini batch kmeans این است که (علی رغم سریعتر بودنش نسبت به الگوریتم kmeans) نتایج متفاوتی در خوشه‌بندی کردن data نسبت به الگوریتم kmeans عادی می‌دهد. در واقع با مقایسه‌ی نتایج clustering در این دو الگوریتم، مشاهده می‌کنیم که نقاطی وجود دارند که توسط الگوریتم mini batch kmeans، در خوشه‌های متفاوتی با خوشه‌های الگوریتم kmeans قرار می‌گیرند.

Question 2)

در الگوریتم kmeans، در حالت عادی centroid ها به صورت رندم انتخاب می‌شوند ولی اگر این انتخاب به حالت نهایی آن‌ها نزدیک نباشد، پس الگوریتم خوشه‌بندی باید در تعداد #iteration های بیشتری انجام شود تا به همگرایی برسد و این یعنی ↑ runtime ⇐ ↓ efficiency. ⇐ به طور کلی دو کاری می‌توانیم انجام دهیم: یا Centroid را بهتر کنیم در ابتدا (Initialization) (مشکل instance based بودن)

و یا k را به صورت optimal انتخاب کنیم. که در این حالت می‌توانیم از نمودار k-Elbow برای انتخاب k بهتر استفاده کنیم که نموداری است که مقدار خطای clustering را به حسب هر k نشان می‌دهد و در آن، در نقطه‌ی آرنج یا همان Elbow، بهترین k را خواهیم داشت.



Scanned by CamScanner

برای حالت انتخاب بهینه‌تر centroids ها در ابتدا (یا همان روش های Improved centroids Initialization) به جای اینکه centroids ها را رندم انتخاب کنیم، می‌توانیم از روش The Naive Sharding Centroid Initialization Algorithm استفاده کنیم.

در این روش، بدین صورت عمل می‌کنیم:

step 1: یک ستون جدید به data اضافه می‌کنیم و برای هر سطر آن، مجموع داده‌های آن سطر را وارد می‌کنیم.

step 2: داده‌های این ستون جدید را از عددی به نزولی، Sort می‌کنیم.

step 3: کل سطرها را k تا k از بالا به پایین جدا می‌کنیم و به هر قسمت اصطلاحاً یک "shard" می‌گویند.

step 4: برای هر shard، مجموع داده‌هایش (به غیر از ستونی که در step 1 درست کردیم) و نیز mean آن‌ها را حساب کرده و در یک سطر جدید وارد می‌کنیم. این سطر جدید یک centroid خوب است.

step 5: هر centroid در هر row shard را به centroid های جدید اضافه می‌کنیم.

step 6: return (centroids)

روش دیگر، الگوریتم kmeans ++ است که به صورت زیر کار می‌کند:

step 1: اولین centroid از روی data point ها به صورت رندم انتخاب می‌شود.

step 2: برای هر data point، فاصله‌اش را از نزدیک‌ترین centroid قبلی پیدا می‌کنیم.

step 3: centroid بعدی را طوری انتخاب می‌کنیم که احتمال انتخاب شدنش متناسب با فاصله‌اش از نزدیک‌ترین centroid انتخاب شده قبلی باشد (نقطه‌ای که بیشترین فاصله از centroid قبلی داشته باشد، احتمالاً به عنوان centroid بعدی انتخاب می‌شود.)

step 4: step 2 و 3 را تا زمانی که k تا centroid (و cluster) پیدا شوند، ادامه بده.

روش دیگر، Ward's method است. این یک روش hierarchial clustering است و می‌توانیم آن را به جای کل داده‌ها اگر زیاد بودند، روی sample ای از آن انجام داد. سپس mean های k تا cluster، همان initial centroids هستند.

Scanned by CamScanner

Question 3)

A) درست → در خوشه‌بندی در الگوریتم DBSCAN، نقاطی به نام core و border داریم که براساس معیارهای فاصله و شعاع همسایگی (Eps) تعریف شده در یک کلاستر قرار می‌گیرند. در واقع نقطه‌ی border نقطه‌ای است که density-reachable با نقطه‌ی core باشد یعنی باید در محدوده فاصله‌ای (distance threshold) از آن قرار گرفته باشد تا بتواند با آن در یک cluster قرار بگیرد.

B) غلط → الگوریتم DBSCAN می‌تواند arbitrary shape را هم می‌تواند خوشه‌بندی کند و پیش‌فرض‌های خاصی روی توزیع داده‌ها ندارد. non-linearly seprable هم باشد یعنی خوشه‌هایی با

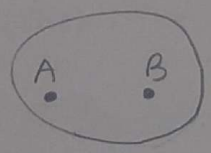
C) غلط → الگوریتم DBSCAN اتفاقاً دارای time complexity پایینی است که برابر $O(n \log n)$ است.

D) درست → این ویژگی در واقع یکی از نقاط مثبت الگوریتم DBSCAN نسبت به الگوریتم k-means است. در kmeans نیاز به داشتن k داریم ولی در DBSCAN نه. چون به صورت dynamically خوشه‌بندی را انجام می‌دهد.

E) درست → این الگوریتم نسبت به noise و outlier دارای robustness هست زیرا به هنگام کلاسترینگ، این نقاط را جزو نقاط core و border که در خوشه‌بندی نقاط می‌شوند، قرار نمی‌دهد و در واقع چون نقاط outlier با نقاط core density-reachable نیستند، در یک خوشه قرار داده نمی‌شوند. پس DBSCAN نسبت به این نقاط، robust و مقاوم است.

Question 4) روش Hierarchial Agglomerative Clustering (HAC):

A)



min distance = dist(A,B) = ۰٫۱۲ =>

=> updating the distance: min(dist(A,B), C) = min(dist(A,C), (B,C)) = min(۰٫۵۱, ۰٫۲۵) = ۰٫۲۵

=> min(dist(A,B), D) = min(dist(A,D), (B,D)) = min(۰٫۸۴, ۰٫۱۹) = ۰٫۱۹

=> min(dist(A,B), E) = min(dist(A,E), (B,E)) = min(۰٫۲۸, ۰٫۷۷) = ۰٫۲۸

=> min(dist(A,B), F) = min(dist(A,F), (B,F)) = min(۰٫۳۴, ۰٫۹۱) = ۰٫۳۴

Scanned by CamScanner

	$((A,B),(C,D)),F)$	E
$((A,B),(C,D)),F)$	0	
E	0,۲۸	0

دوباره از اول :
ماتریس فاصله داریم

$\Longrightarrow$ min distance = dist $(C,D) = 0{,}14 \Rightarrow$

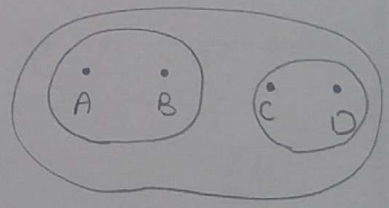
$\Rightarrow$ updating the distance : min dist $((C,D),(A,B)) = \min(\text{dist}(C,(A,B)),(D,(A,B))) = \min(0{,}25, 0{,}14) = 0{,}14$

min dist $((C,D),E) = \min(\text{dist}(C,E),(D,E)) = \min(0{,}70, 0{,}45) = 0{,}45$

min dist $((C,D),F) = \min(\text{dist}(C,F),(D,F)) = \min(0{,}93, 0{,}20) = 0{,}20$

دوباره ماتریس فاصله داریم :

$\Longrightarrow$ min distance = dist $((A,B),(C,D)) = 0{,}14 \Rightarrow$

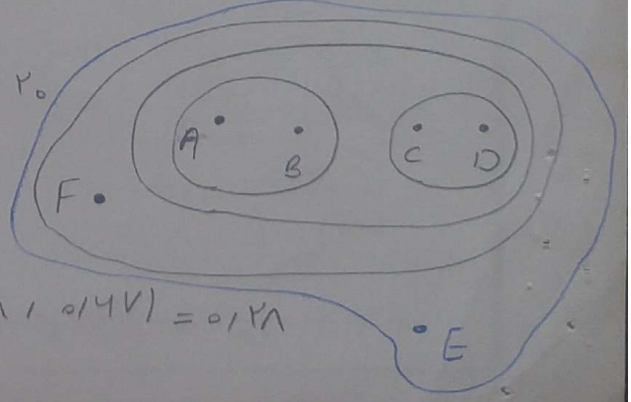


$\Rightarrow$ updating the distance : min dist $(((A,B),(C,D)),E) = \min(((A,B),E),((C,D),E)) = \min(0{,}28, 0{,}45) = 0{,}28$

min dist $(((A,B),(C,D)),F) = \min(((A,B),F),((C,D),F)) = \min(0{,}34, 0{,}20) = 0{,}20$

دوباره ماتریس فاصله داریم :

$\Longrightarrow$ min distance = dist $(((A,B),(C,D)),F) = 0{,}20$



$\Rightarrow$ updating the distance : min dist $(((A,B),(C,D)),F),E) =$

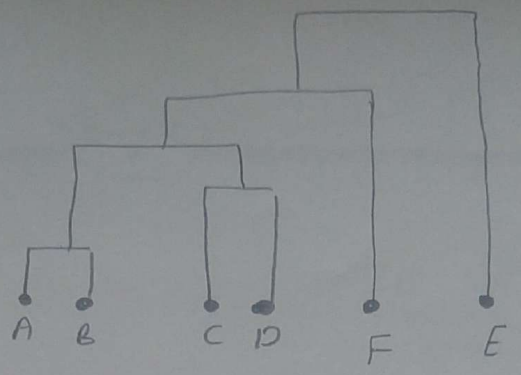
$\min(((A,B),(C,D)),E),(F,E)) = \min(0{,}28, 0{,}47) = 0{,}28$

$\Rightarrow$ در مرحله‌ی آخر، E را به کلاستری که ساخته‌ایم تا اینجا، اضافه می‌کنیم

Scanned by CamScanner

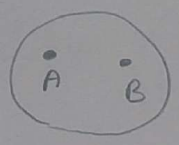
ص ۵

=> یعنی dendogram به صورت :
مقابل ظاهر شد



B) در قسمت complete link ، در مرحله‌ی updating the distance ، به جای انتخاب min مقدار ، باید max مقدار را انتخاب کنیم.

min distance = dist (A, B) = ۰/۱۲ =>



=> updating the distance : max(dist (A,B), C) = max ((A,C), (B,C)) = max(۰/۵۱ , ۰/۲۵) = ۰/۵۱

=> max(dist (A,B), D) = max ((A,D), (B,D)) = max (۰/۸۴ , ۰/۱۹) = ۰/۸۴

=> max(dist (A,B), E) = max ((A,E), (B,E)) = max(۰/۲۸ , ۰/۷۷) = ۰/۷۷

=> max(dist (A,B), F) = max ((A,F), (B,F)) = max (۰/۳۴ , ۰/۴۱) = ۰/۴۱

=>

	((A,B),F)	((C,D),E)
((A,B),F)	۰	
((C,D),E)	۰/۹۳	۰

Scanned by CamScanner

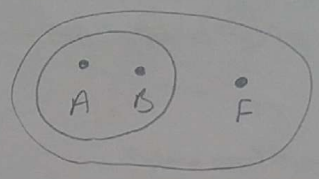
$\Rightarrow$ min distance $= dist(C, D) = 0{,}14$ $\Rightarrow$

$\Rightarrow$ updating the distance: $\max(dist(C,D), (A,B)) = \max((A,B), C), ((A,B), D)) = \max(0{,}20, 0{,}18) = 0{,}18$

$\Rightarrow \max(dist(C,D), E) = \max(dist(C,E), (D,E)) = \max(0{,}70, 0{,}50) = 0{,}70$

$\Rightarrow \max(dist(C,D), F) = \max(dist(C,F), (D,F)) = \max(0{,}93, 0{,}20) = 0{,}93$

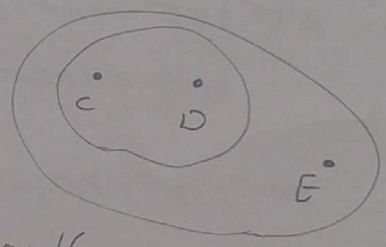
$\Rightarrow$ min distance $= dist((A,B), F) = 0{,}41$ $\Rightarrow$



$\Rightarrow$ updating the distance: $\max(dist((A,B), F), (C,D)) = \max(((A,B), (C,D)), (F, (C,D))) = \max(0{,}18, 0{,}93)$
$= 0{,}93$

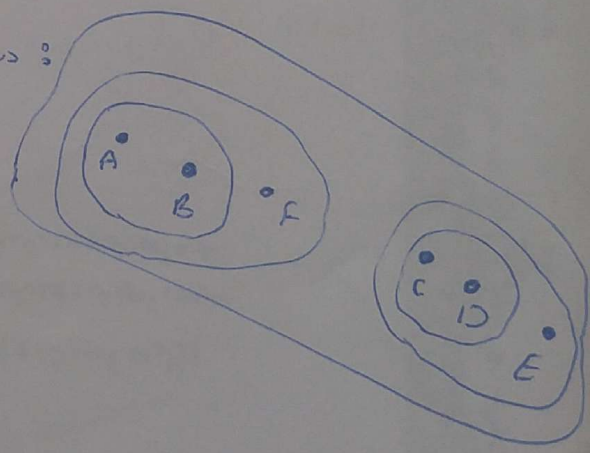
$\Rightarrow \max(dist((A,B), F), E) = \max(((A,B), E), (F, E)) = \max(0{,}77, 0{,}47) = 0{,}77$

$\Rightarrow$ min distance $= dist((C,D), E) = 0{,}70$ $\Rightarrow$



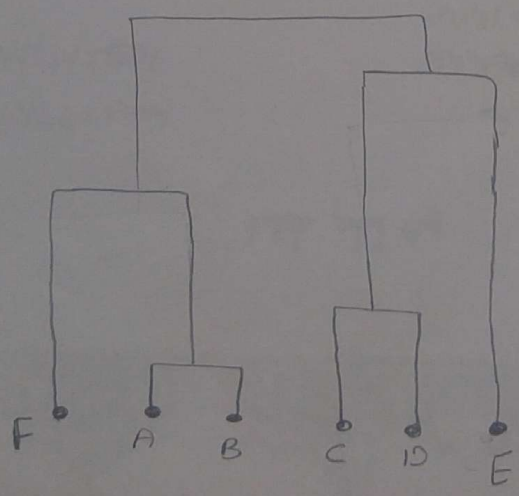
$\Rightarrow$ updating the distance: $\max(dist((C,D), E), ((A,B), F)) = \max(((C,D), ((A,B), F)), (E, ((A,B), F))) =$
$\max(0{,}93, 0{,}77) = 0{,}93$

$\Rightarrow$ در مرحله آخر، دو کلاستر ساخته شده، را با هم merge می‌کنیم.



$\Rightarrow$

$\Rightarrow$ dendogram یعنی: به صورت مقابل خواهد شد

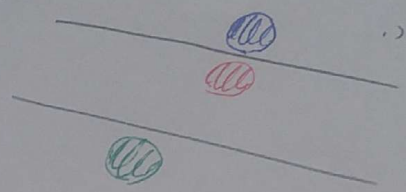


Scanned by CamScanner

Question 5)

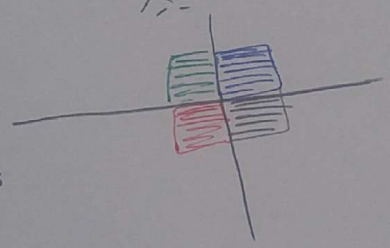
A) DBSCAN → چون در شکل (A) داده‌ها به صورت linearly seprable نیستند، و پراکندگی شان به صورت density based است، پس روش kmeans را نمی‌توان استفاده کرد.

B) هر دو روش kmeans , DBSCAN → زیرا در این جا داده‌ها هم linearly seprable هستند و هم به صورت density based. پس از هر دو روش خوشه‌بندی می‌توان استفاده کرد.



: یک نمونه خوشه‌بندی خطی با kmeans می‌تواند به صورت مقابل باشد.

C) هر دو روش kmeans , DBSCAN → چون مانند حالت قبل، داده‌ها هم linearly seprable هستند و هم براساس توزیع density در کنار هم قرار گرفته‌اند پس می‌توان از هر دو روش استفاده کرد.



: یک نمونه جداسازی خطی

D) DBSCAN → در اینجا چون داده‌ها به صورت خمیده و غیرخطی توزیع شده اند، پس قابل جداسازی خطی نیستند پس فقط از الگوریتم DBSCAN که براساس density داده‌ها کار می‌کند، می‌توان استفاده کرد.

Scanned by CamScanner