**תרגיל כיתה 12- k-means**

**אלגוריתם K-means**

עד כה תרגלנו שיטת האִשכּול היררכי, שהיא דוגמא לגישה 'מלמטה למעלה' (bottom-up). עכשיו נתרגל חלוקה לקבוצות בשיטת k-means, שהיא דוגמא לגישה ההפוכה – 'מלמעלה למטה' (top-down).

מטרת האלגוריתם למזער את השונות בתוך כל קבוצה.

האלגוריתם:

1. אתחול: תייג כל דוגמה במספר קלסטר אקראי.
2. חזור עד להתכנסות:
	1. לכל קלסטר חשב צנטרואיד (הנקודה ה'ממוצעת' בקלסטר(.
	2. תייג כל נקודה בהתאם לתיוג הצנטרואיד הקרוב אליה ביותר.

הערה: שלב האתחול יכול להיעשות גם בדרכים אחרות. למשל על ידי מיקום הצנטרואידים באופן אקראי, ושיוך כל דוגמא לצנטרואיד הקרוב ביותר.

ה'מרחק' בין הנקודות לצנטרואידים לא חייב להיות מחושב דווקא אוקלידית. אפשר לחילופין להשתמש בהגדרת 'מרחק' לפי קורלציה, או כל הגדרת 'מרחק' אחרת. כמובן שבחירות שונות של הגדרת 'מרחק' עשויות להוביל לחלוקות שונות לקבוצות.

כדי שהאלגוריתם יעצר מתישהו, צריך להגדיר תנאי עצירה (הגדרה להוראה "חזור עד ל*התכנסות*" לעיל). בהרצאה ראיתם 3 אפשרויות לתנאי עצירה:

1. כשבכל ריצה של הלולאה אין יותר מעבר של דוגמאות מקבוצה לקבוצה.
2. כשבכל ריצה של הלולאה השינוי בממוצע המרחקים של הדוגמאות מהצנטרואידים קטן מערך מסויים.
3. לאחר מספר קבוע מראש של הרצות.

כדי להמחיש ויזואלית את אופן פעולת האלגוריתם, ניכנס לאתר <https://www.naftaliharris.com/blog/visualizing-k-means-clustering/> המממש את האלגוריתם ומציג את החלוקה לקבוצות בכל איטרציה.

הערה: k-means מאפשר ניתוח של נתונים במימד גבוה, ובזה עיקר הכוח שלו. בתרגול זה נעבוד בעיקר עם נתונים ב 2D, לצרכי ויזואליזציה.

מכאן ואילך נשתמש בפונקציהkmeans המובנית ב-R ונכנסים את התוצאה למשתנה kData:

kData = kmeans(Data, centers = 2)

**הקלט:**

Data – מטריצת הנתונים. כל שורה מייצגת דוגמה, לפיכך מספר השורות במטריצה הוא כמספר הדוגמאות ומספר העמודות במטריצה הוא כמימד הדוגמאות.

centers – מספר הקלסטרים שאנו מעוניינים לקבל בסוף התהליך.

**הפלט:**

הפלט הוא במבנה נתונים מסוג list אותו אינכם נדרשים להבין לעומק. בקצרה, אובייקט של list יכול להכיל בתוכו אובייקטים מסוגים שונים (למשל, גם character, גם numeric וגם data frame במקביל). ניתן לגשת לאלמנטים באמצעות הסימן $.

kData$cluster - וקטור שאורכו כמספר הדוגמאות ובו התיוג של כל דוגמה בהתאם לקלסטר אליו היא משתייכת.

kData$centers – מטריצה בה בכל שורה יושב הצנטרואיד של כל קלסטר בהתאמה )לפיכך מספר השורות בה שווה ל- k ומספר העמודות שווה למימד הדוגמאות(.
kData$size – וקטור עם k איברים, ובו גודל כל אחד מהקלאסטרים.

kData$withiness – וקטור עם k איברים, ובו סכום המרחקים בתוך כל אחת מהקבוצות.

ניתן לצייר את התוצאה של חלוקת הנתונים שלנו ל-2 קלאסטרים:

plot(Data, col=kData$cluster)



**חסרונות של k-means**

נדבר כעת על כמה חסרונות של k-means, ונציע דרכים להתגבר על חלק מהם.

**תלות בתנאי התחלה**

נתונות 24 נקודות במרחב דו-מימדי. הדוגמאות עברו clustering פעמיים באמצעות שיטת k means עם k=3, תחת תנאים זהים. נתונים להלן תנאי ההתחלה של האלגוריתם בשני המקרים )צורה שונה מסמלת תיוג התחלתי שונה(.



איך תהיה החלוקה לקלסטרים לאחר התכנסות האלגוריתם?



הסיבה לכך היא ש kmeans הוא אלגוריתם אופטימיזציה שמוצא כל פעם 'מינימום מקומי', כלומר פתרון שהוא אופטימלי **בהינתן תנאי ההתחלה** (שכאמור, נקבע אקראית). כדי להגיע ל'מינימום גלובלי', כלומר לפתרון האופטימלי באמת, צריך לנסות מספר רב של תנאי התחלה שונים, ולקחת את הפתרון הכי טוב מבין כולם – זה שבו סך המרחקים של הדוגמאות מהצנטרואידים הוא הקטן ביותר.



נדגים כעת חסרונות נוספים של k-means. עבור כל הדגמה, נסו לחשוב האם החלוקה המתקבלת משקפת את החלוקה הטבעית של הנתונים, ומדוע זה קורה.

1. **קבוצות לא שוות**



1. **קבוצות בצפיפות לא אחידה**



1. **קבוצות לא "עגולות"**



**חשיבות הנרמול:**

להלן שתי דוגמאות של אשכול לפני ואחרי נרמול

 לפני הנרמול אחרי הנרמול



**קביעת מספר הקבוצות האופטימלי**

לבסוף, נדבר על חסרון של k-means לעומת אִשכּול היררכי. בניגוד לאשכול היררכי שבו החלוקה לקבוצות 'צומחת' מתוך הנתונים עצמם, ב k-means החלוקה לקבוצות נקבעת על ידי המשתמש. אך כיצד נדע מלכתחילה לכמה קבוצות יש לחלק את נתונים?

נראה כמה דרכים למצוא את מספר הקלסטרים האופטימלי.

באמצעות אשכול היררכי

אפשרות אחת היא להסתכל על דנדרוגרמות של אשכול הייררכי של הנתונים ולנסות להסיק מהן על כמות הקבוצות האופטימלית:

plot(hclust(dist(Data), method = "single"), main = "Single Linkage", labels = FALSE)

plot(hclust(dist(Data), method = "complete"), main = "Complete Linkage", labels = FALSE)

plot(hclust(dist(Data), method = "average"), main = "Average Linkage", labels = FALSE)

plot(hclust(dist(Data), method = "centroid"), main = "Centroid Linkage", labels = FALSE)



שימו לב שניסינו כל מיני אופנים של אשכול, כדי לקבל רושם שאינו תלוי בשיטה ספציפית. ניתן להתרשם שיש כנראה 4-5 קבוצות מובהקות בנתונים – קבוצות שמרחקן מהקבוצות האחרות גדול משמעותית מהמרחקים בתוך הקבוצה.

Scree plot

גישה אחרת היא קצת יותר כמותית. אם נחלק את הנתונים למספר הולך ועולה של קבוצות, השונות בתוך כל קבוצה תלך ותקטן. נרצה לראות **עד כמה** השונות קטנה עם כל תוספת של קבוצה – האם 'מרוויחים' הרבה מחלוקה ליותר קבוצות או רק מעט:

max\_clusters = 20

sum\_dist = vector(length = max\_clusters)

for (k in 1:max\_clusters) {

 kData = kmeans(Data, centers = k)

 sum\_dist[k] = kData$tot.withinss

}

plot(x = seq(1,20), y = sum\_dist, col = 'blue',

 xlab = 'num of clusters', ylab = 'sum of within-group distances')

lines(x = seq(1,20), y = sum\_dist, col = 'blue')

axis(1,labels = seq(1,20), at = seq(1,20))



קיבלנו גרף המכונה Scree Plot, וממנו אפשר ללמוד מה המספר האופטימלי של קלסטרים. מובן מאליו שככל שמחלקים ליותר קלסטרים השונות בתוך הקלסטרים קטֵנה, אך יש מספר מסויים של קלסטרים שמעבר לו כבר לא מרוויחים הרבה. התוצאה לא תמיד חד משמעית אבל בדרך כלל תיתן כיוון, ובמקרה זה אפשר להתרשם שיש 4 קבוצות, כי כשיש 4 קבוצות הוספת קבוצות נוספות כבר לא משנה הרבה את סך המרחקים בתוך הקבוצות.

מדד סילואט (Silhouette)

מדד לטיב ההשתייכות של דוגמא לקבוצה אליה היא משתייכת, ביחס למרחק שלה מקבוצות אחרות. מתמטית זה נראה כך:

$$s= \left\{\begin{matrix}1-\frac{a}{b}&a\leq b\\\frac{b}{a}-1&b\leq a\end{matrix}\right.$$

a הוא **מדד הלכידות**: ממוצע המרחקים בין הדוגמא לדוגמאות האחרות באותו קלסטר.

b הוא **מדד ההפרדה**: ממוצע המרחקים בין הדוגמא לדגמאות בקלסטר הכי קרוב אליה (חוץ מזה שאליו היא משוייכת).

הטווח של s הוא תמיד בין 1, ל- (1-). ערכים קרובים ל- 1 מבטאים התאמה טובה של הדוגמא לקלסטר שאליו היא משוייכת, ערכים שליליים מתקבלים כאשר הדוגמא מתאימה יותר לקלסטר אחר מזה שאליו היא משוייכת, וערכים קרובים לאפס מתקבלים כאשר הדוגמא מתאימה במידה שווה לקלסטר שלה ולקלסטר אחר.

הפונקציה silhouette(X,dist) מקבלת חלוקה לקלאסטרים X, ואת מטריצת המרחקים בין הנקודות dist, ומציגה גרף של ערכי סילואט של כל הדוגמאות, כשהן ממויינות לקבוצות שלהן. לפי הפרופיל המתקבל אפשר להבין עד כמה טובה החלוקה לקבוצות. נפעיל את הפונקציה על חלוקות של הדאטה ל- 2 עד 6 קבוצות ונתבונן בפרופילים שמתקבלים:

par(mfrow=c(3,2))

for(i in 2:6){

 kData = kmeans(Data, centers = i)

 plot(silhouette(kData$cluster, dist(Data)), col = 1:i, border = NA,
 do.n.k = F, do.clus.stat = T, cex.lab=1.5, cex.axis=1.5, cex.sub = 1.5,
 cex.names = 1.5, main = "")

}

dev.off()

שימו לב, על מנת להפעיל את הפונקציה silhouette יש צורך בהתקנת ה-package ששמו cluster:

if(!("cluster" %in% rownames(installed.packages()))) {install.packages("cluster")}

שימו לב שבשרטוט הפונקציה הוספנו הגדרה לגודל הטקסט (cex) והורדנו את הכותרת (main). בנוסף, הגדרנו שלא יוצגו לנו כותרות מסויימות – do.n.k. ניתן להסתיר את הממוצעים של כל אחד מהקלאסטרים באמצעות שינוי של do.clus.stat ל-TRUE.



אפשר להבין שעד 4 קבוצות, מתקבלים פרופילים מאוד רחבים – בכל קבוצה לרוב מוחלט של הדוגמאות ערך סילואט גבוה. אך כשמחלקים ל- 5 קבוצות, מופיעות קבוצות עם פרופיל צר, שבו לחלק גדול מהדוגמאות יש ערך סילואט נמוך. זה מצביע על כך שהחלוקה הזו לא מתאימה לנתונים – יש הרבה דוגמאות שקרבתן לקלסטרים אחרים דומה לקרבתן לקלסטר שלהן.

ניתן להציג זאת גם בגרף המתאר את ערך הסילואט הממוצע כתלות במספר הקלאסטרים:

silAvg = matrix(ncol = 2, nrow = 5)

for(i in 2:6){

 kData = kmeans(Data, centers = i)

 silAvg[i-1,1] = i

 silAvg[i-1,2] = mean(silhouette(kData$cluster, dist(Data))[,3])

{

plot(silAvg, xlab = "Number of clusters", ylab = "Average silhouette value")

lines(silAvg)



ראינו את הקושי בחיפוש קבוצות ברב מימד – קשה למצוא מהם המימדים הרלוונטיים. בשבוע הבא נלמד על שיטת PCA המאפשרת למצוא מהם המימדים הרלוונטיים ביותר בנתונים.