

מבוא לבינה מלאכותית – תרגול 6

נושאים:

- למידה לא מפוקחת - אשכול:
אלגוריתמים מבוססי מרחק: Graph Partitioning, Prim, אשכול היררכי.
אלגוריתמים לאשכול ב- \mathbb{R}^n : K-means, K-medoids, Fuzzy C-means, GMM.

למידה לא מפוקחת:

ההבדל בין מה שעשינו עד היום (למידה מפוקחת) לבין למידה לא מפוקחת הוא שבלמידה לא מפוקחת אנחנו לא יודעים תיגוד לדיגמות. המשימות שאנחנו יכולים להעמיד לעצמנו הן לא משימות של סיווג (אין מחלקות לסווג אליהן), אלא משימות כמו חלוקה של הדגימות לקבוצות (אשכול), הורדת ממדים (PCA, ICA, Auto-encoding), זיהוי חריגות ועוד.
דוגמה קונקרטית: טלסקופ חלל אוסף נתונים על גרמי שמיים לא ידועים. הטלסקופ יכול לאסוף פיצ'רים של גרמי השמיים (כמו ספקטרום התדרים המגיעים מכל גרם שמיים ועוד), ומכיוון שהתיגוד לא ידוע, אין אלא לנסות ולבדוק אם יש קבוצות של כוכבים שדומים זה לזה, או אם יש כוכבים חריגים.

אשכול:

אלגוריתמי אשכול מבוססי מרחק:

לא תמיד יש לנו פיצ'רים לכל קדקוד. לעתים, מה שיש לנו הוא רק מרחקים פנימיים בין הדוגמות, אבל גם זה מספיק לנו לאשכול. נראה כעת 3 אלגוריתמים לאשכול קשיח, כלומר לכל נקודה יהיה סיכוי 1 להשתייך לאחד האשכולות ו-0 לשאר.
כאמור, יש לנו אך ורק מרחקים בין נקודות.

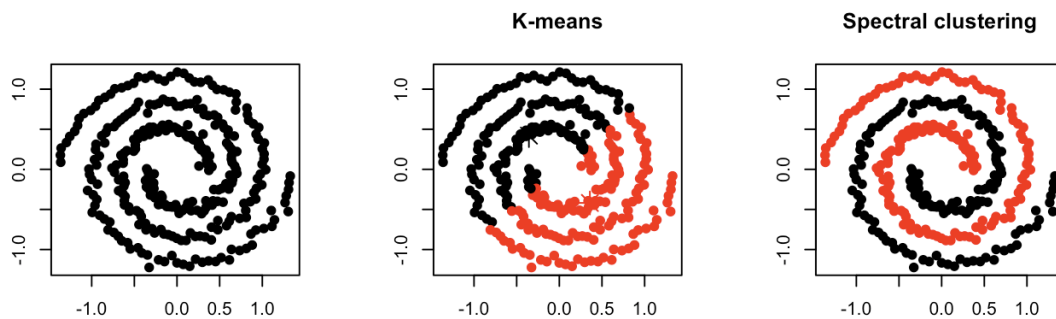
Graph Partitioning (ידוע גם כאשכול ספקטרלי):

בעזרת המרחקים נבנה גרף ממושקל: כל דגימה מהווה קודקוד, קשתות בין קודקודים ממושקלות לפי המרחקים הידועים ($w_{ij} = 1/d_{ij}$, מה שמכונה דמיון). השיטה הזו מוצאת חלוקה של כלל הגרף לשתי קבוצות A, B זרות כך ש- $\sum_{i \in A, j \in B} w_{ij}$ מינימלי, כלומר הקודקודים השונים ביותר ימצאו את עצמם בקבוצות זרות, וקודקודים דומים ימצאו באותה קבוצה (בהערת אגב, אפשר להשתמש בה כדי לחלק ליותר קבוצות, בעזרת שימוש ביותר וקטורים עצמיים).

בקצרה (הסבר מפורט יותר בהרצאה): מחשבים לפלסיאן של הגרף, מחשבים לו ערך עצמי ווקטור עצמי שני (הערך העצמי הראשון חסר משמעות עבורנו – ע"ע 0 עם ו"ע שכל הרכיבים שלו שווים לאותו קבוע). כעת, לכל נקודה אפשר לשייך גודל מסוים שניתן לה בהתאמה לרכיב בווקטור. כעת, נפריד את הנקודות לפי ערך סף שכל הנקודות עם רכיבים בגודל גדול ממנו ישתייכו ל-A והשאר ל-B.

היתרון באלגוריתם זה מגיע מהגדרת הבעיה. הוא מסוגל לפתור בעיית אשכול שמבוססת רק על המרחק של הנקודות אלה מאלה, ולא על ייצוג וקטורי לכל נקודה. אלגוריתמים שנראה בהמשך מניחים שלאשכולות יש מרכזים שסביבם מפוזרות (בצורה כלשהי) הנקודות, אבל זה לא תמיד נכון.

למשל:



יש לאלגוריתם גם חסרון עיקרי – חישוב ערכים עצמיים בהינתן סט אימון גדול הוא משימה שלוקחת הרבה זמן לחישוב.

:Prim

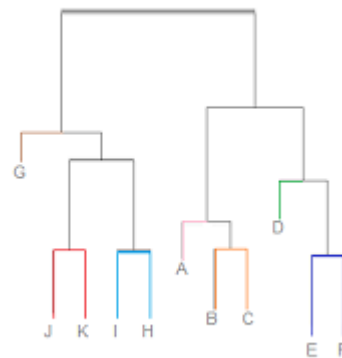
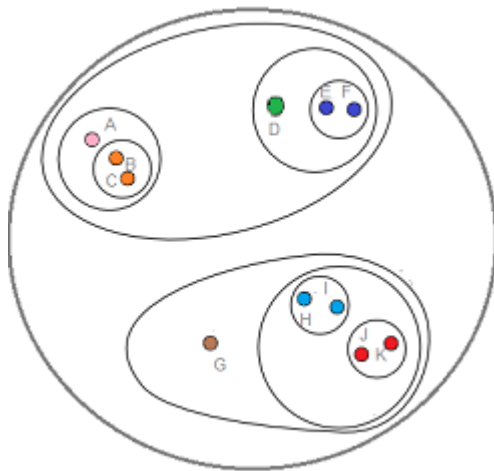
אלגוריתם Prim הוא אלגוריתם שמוצא עץ פורש מינימלי בגרף ממושקל חיובי. הפעם, מקבוצת הנקודות שלנו נבנה גרף ממושקל עם משקלים שהם המרחקים עצמם. הפעלת האלגוריתם על גרף (אם הכל אידיאלי) משאירה עץ שבו שני סוגי קשתות: הרבה קשתות במשקלים נמוכים במיוחד (בין דוגמות שנמצאות באותו אשכול) ומעט קשתות במשקלים גבוהים (בין דוגמות מאשכולות שונים). נותר להסיר את המשקלים הגבוהים ויישאר לנו אשכולות נפרדים.

הבעיה העיקרית באלגוריתם זה – אין לנו דרך ממשית לדעת כמה טוב הצלחנו, זה אלגוריתם היוריסטי. למרות זאת, זה אלגוריתם פשוט מאוד שבעבר (לפני שרשתות נירונים הפכו למוצלחות מאוד) השתמשו בו ל-image segmentation.

אשכול היררכי:

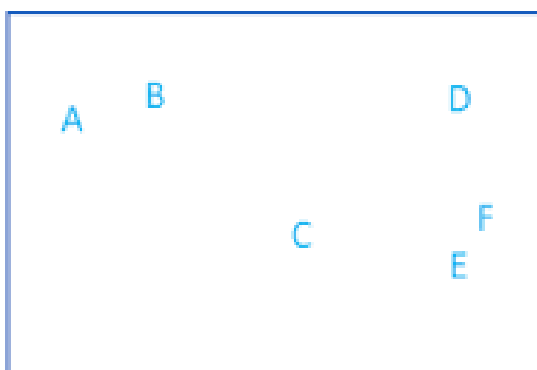
בגישה שנלמד לשיטה זו (bottom up), האלגוריתם מתחיל מאשכולות המורכבים מנקודות בודדות ובכל איטרציה שלו מחברים בין שני אשכולות שהמרחק ביניהם הוא הקטן ביותר. אפשר להשתמש בפונקציות שונות לחישוב מרחקים בין אשכולות (כמו מרחק מינימלי בין נקודות מאשכולות שונים, מרחק ממוצע ועוד).

מה שנוצר הוא דנדרוגרם (dendrogram):

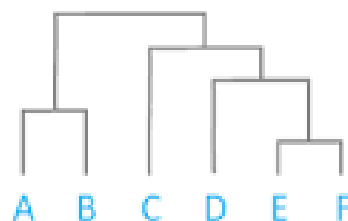


נחליט על רמה מסוימת בעץ, שממנה נסתכל (כלומר, אין לנו שום אינטרס להשתמש בשלב הסופי של האלגוריתם, שם הכל מחובר לאשכול אחד ענק. נצטרך "לחתוך" ברמה מסוימת כך שנקבל מספר אשכולות).

אלגוריתם זה בעייתי גם הוא – שוב אין לנו פונקציית מטרה שרוצים למזער. מעבר לכך, האלגוריתם יכול פשוט שלא לעבוד כמו שצריך כשהדנדרוגרם עשוי להיות כמו שרואים למטה – מכיל אשכול אחד גדול שבמרבית האיטרציות פשוט מתווספות אליו עוד ועוד נקודות.



Dendrogram



אלגוריתמי אשכול ב- \mathbb{R}^n :

כעת, בשונה מקודם, לכל דגימה משויכת נקודה במרחב ממש. נלמד אלגוריתמים שמשתמשים במיקומים האלה במרחב כדי לחלק את הנקודות לאשכולות. נתחיל מרקע קצר לגבי מאיפה ביצוע האלגוריתמים מגיע – תהליך EM:

Expectation Maximization (EM)

נחזור למה שהגדרנו בתחילת הקורס, עם קצת תוספת: יש לנו אוסף תצפיות X , אוסף משתנים סמויים של המערכת Z (אלה משתנים שאין לנו גישה אליהם. למשל, משתנה לכל דגימה שקובע לאיזה אשכול שייכת הדגימה), ואוסף פרמטרים θ שאנחנו מחפשים, כך ש- $P(\theta|X, Z)$ יהיה מקסימלי. החלפנו את הסיכוי הזה בפונקציה קשורה לפונקציה זו, פונקציית ה-likelihood, $P(X, Z|\theta) = L(\theta; X, Z)$, ואמרנו שבמקום למקסם את המקורית, נמקסם את זו. הבעיה היא שכאשר יש לנו גם משתנים חבויים, המקסום נעשה בעייתי יותר. EM הוא אלגוריתם איטרטיבי שמחפש את מה שאנחנו בדיוק רוצים על ידי הפעלה חוזרת של שני שלבים:

1. שלב ה-Expectation – בהינתן הערכה קיימת של הפרמטרים $\theta^{(t)}$ (וכן X , שתמיד ידוע לנו), מחשבים את ערך התוחלת של לוג הנראות. למעשה, בשלב זה מבצעים הערכה של ערכי המשתנים הסמויים, Z , בהתבסס על ההערכה הנוכחית של הפרמטרים (לכן יש הקוראים לשלב זה Estimation).
2. שלב ה-Maximization – לאחר שקבענו את המשתנים הסמויים, מבצעים הערכה מחודשת לפרמטרים, $\theta^{(t+1)}$, שיהיו הערכים שעבורם ערך התוחלת של לוג הנראות, בהינתן הקביעה הנוכחית של המשתנים הסמויים שלנו, מקסימלי.

ראיתם בהרצאה שהאיטרציות של האלגוריתם הזה אכן מבצעות צעדים לכיוון הנכון, כלומר שהאלגוריתם ממקסם.

K-means:

זהו אלגוריתם לאשכול קשיח (שוב, כל נקודה תשתייך לאשכול אחד בלבד). האלגוריתם הזה משתמש ב- K מרכזי אשכולות שמתעדכנים באיטרציות ובהתאם שיוך הנקודות לאשכולות משתנה.

אלגוריתם:

1. אתחול אקראי של מרכזים μ_1, \dots, μ_k .

2. לכל איטרציה:

- שיוך כל נקודה i לאשכול $j = \arg \min_n \|x_i - \mu_n\|^2$ (כאשר המרחק הוא אוקלידי).

- עדכון המרכזים: $\mu_n^{new} = \frac{1}{m_n} \sum_{j \in Cluster_n} x_j$ כאשר m_n מספר הנקודות באשכול ה- n .

יש לציין: האלגוריתם יתכנס למינימום כלשהו על השגיאה $L = \sum_{n=1}^k \sum_{j \in Cluster_n} \|x_j - \mu_n\|^2$ (ובגלל זה לוקחים מטריקה אוקלידית ולא כל מטריקה אחרת), אבל הוא לא מוכרח להיות מינימום גלובלי, וזה נכון גם לאלגוריתמים אחרים שמופיעים כאן.

שימו לב שהשיוך של כל נקודה לאשכול הוא המשתנה הסמוי כאן, ואכן החלק האיטרטיבי באלגוריתם הוא של EM.

K-medoids:

הפעם, משתמשים במרחב רק במה שיש. כלומר, למרכזי האשכולות לוקחים נקודות קיימות.

אלגוריתם:

1. אתחול אקראי של k נקודות מתוך נקודות הקלט שמשמשות מרכזים.

2. שיוך כל נקודה למרכז הקרוב אליה, לפי מטריקה שנבחרה מראש.

3. כל עוד פונקציית העלות $\sum_{C_i} \sum_{P_i \in Cluster_i} \|P_i - C_i\|$ (לפי המטריקה שבחרנו) יורדת:

לכל מרכז ולכל נקודה שאינה המרכז, מבצעים החלפה בין הזוגות וחישוב העלות מחדש.

אם היא ירדה, חוזרים ל-2 עם המרכזים המוחלפים, אחרת ממשיכים לבדוק החלפות

אחרות. אם אין החלפה שמורידה את העלות, האלגוריתם יעצור.

אלגוריתם זה דומה מאוד ל-K-means, אבל K-medoids הוא קצת יותר יציב מ-K-means, מכיוון שהוא ממזער פונקציית שגיאה שמבוססת על מרחקים כלליים בין זוגות הנקודות ולא בהכרח על סכום מרחקים אוקלידיים. לעומת זאת, האלגוריתם הזה פחות יעיל חישובית מ-K-means ואתחולים שונים שלו עלולים להביא לפתרונות שונים לגמרי.

זהו אלגוריתם שאינו מבוסס EM, אם כי ניתן ליישם את השיטה גם בצורה מבוססת EM.

:Gaussian Mixture Model (GMM)

זהו אלגוריתם אשכול רך. הסיבה – כל נקודה לא משויכת לאשכול יחיד אלא יש לה סיכויים בין 0 ל-1 להשתייך לכל אשכול, והאשכול שנבחר להתאים לכל נקודה יהיה זה שהסיכוי של הנקודה להשתייך לשם יהיה הכי גבוה.

הפעם, נניח שקיימים K אשכולות, לכל אשכול יש מרכז $\vec{\mu}_j$, מטריצת שונות Σ_j שמתארת את השונות של כל הנקודות שבאשכול, וסיכוי אפריורי (כלומר, בלי קשר למרחקים וכו') של נקודה בדאטא להשתייך לאשכול π_j . לכל נקודה יש משתנה סמוי המסמן לאיזה מרכז הוא משתייך z_{ij} (דגימה i נלקחה מאשכול j).

גם כאן משתמשים ב-EM (למעשה, זו הדוגמה הקלאסית לשימוש ב-EM). הנוסחאות מעט יותר מסובכות מהמקרים האחרים, לכן לא כתבתי אותן כאן אלא רק צירפתי את [הקישור הבא להסבר מפורט](#).

הערה – בדרך כלל, חישוב המטריצות Σ_i סובל מאוד מרעש (בגלל ריבוי פרמטרים להערכה בעזרת לא מספיק נקודות). לכן, לעתים מניחים הנחות מקלות על מבנה המטריצה (למשל: אלכסונית, זהה עבור כל האשכולות, כדורית = שונות שווה בכל כיוון ועוד).

:Fuzzy C-means

זהו אלגוריתם אשכול רך שמשלב אלמנטים מ-K-means ומ-GMM, ושוב נעשה בו שימוש ב-EM.

הפעם, מגדירים את פונקציית המטרה להיות $L = \sum_{i \in \text{Points}, j \in \text{Centers}} u_{ij}^m \|\vec{x}_i - \vec{\mu}_j\|^2$, עבור פרמטר $1 \leq m < \infty$. נדרוש גם שלכל נקודה i , $u_{ij} \geq 0$, $\sum_j u_{ij} = 1$ מסמל את השיוך של נקודה לאשכול, זה כמו $P(z_{ij})$ ב-GMM). שלבי העדכון המתקבלים הם:

:Expectation

$$u_{ij} = \frac{1}{\sum_{k \in \text{Centers}} \left(\frac{\|\vec{x}_i - \vec{\mu}_j\|}{\|\vec{x}_i - \vec{\mu}_k\|} \right)^{\frac{2}{m-1}}}$$

:Maximization

$$\mu_j = \frac{\sum_i u_{ij}^m \vec{x}_i}{\sum_i u_{ij}^m}$$

נשים לב למקרים מיוחדים: אם $m \rightarrow \infty$, אז סיכויי השיוך של נקודה לכל אשכול הופכים להיות אחידים, שאיפה לערך שיהיה שווה לאחד חלקי מספר האשכולות.

אם $m \rightarrow 1$, אנחנו שואפים למקרה פרטי של GMM כדורי, שדומה ל-K-means בכך שעבור האשכול הקרוב ביותר לכל נקודה הערך המתאים ישאף ל-1, ולכל אשכול אחר הערכים u_{ik} ישאפו ל-0. בסך הכל, m הוא היפר-פרמטר של המודל שקובע כמה מחפשים שהשיוך של הדאטא לאשכולות יהיה קשיח (ככל ש- m גדול יותר, כך האשכול קשיח פחות).