מבוא למערכות לומדות - הרצאה 9 - למידת ייצוג (הצברה)

10 ביוני 2015

1 סיכום ביניים מספר 2 - תורת ההכללה ואלגוריתמי למידה

ניזכר שהמטרה הבסיסית בלמידה חישובית היא ללמוד מיפוי

$$h^* : X \to Y$$

על סמך מדגם

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \subset X \times Y$$

חלק א - מודל PAC ותורת ההכללה: בשלושת ההרצאות הראשונות הגדרנו את מודל PAC ללמידה, ופיתחנו את תורת ההכללה, שמאפשרת לנו להבין כמה דוגמאות עלינו לראות על מנת ללמוד. במודל התיאורטי שהצגנו, הנחנו שקיימת התפלגות $\mathcal{D}$ שממנה נדגמות הדוגמאות. בהתאם לכך, הגדרנו את ההפסד של היפותזה $h : X \to Y$ ע"י

$$L_{\mathcal{D}}(h) = E_{(x,y)\sim\mathcal{D}} l(h(x), y)$$

האבחנה הראשונה שעשינו היא ש־"אין ארוחות חינם". כלומר, לא קיים אלגוריתם המסוגל ללמוד כל מיפוי כאשר כמות הדוגמאות מוגבלת. במילים אחרות, אנחנו צריכים איזשהו **ידע מוקדם** על הפונקציה h^* (או על ההתפלגות $\mathcal{D}$) על מנת להיות מסוגלים ללמוד. הדרך שבה ניתן לבטא ידע מוקדם במודל PAC היא הצבעה על **מחלקת היפותזות** (כלומר, אוסף $\mathcal{H} \subset Y^X$) בה אנו משערים שקיימת היפותזה טובה. תורת ההכללה שפיתחנו הראתה שכאשר $\mathcal{H}$ "קטנה" בהשוואה לכמות הדוגמאות שיש בידנו, השגיאה האמפירית ($L_S(h)$) של **כל ההיפותזות ב־$\mathcal{H}$** קרובה לשגיאה האמיתית ($L_{\mathcal{D}}(h)$). לכן, כל אלגוריתם המחזיר היפותזה ב־$\mathcal{H}$ עם שגיאה אמפירית טובה, יחזיר בעצם היפותזה עם שגיאה אמיתית טובה. לכן, בהינתן אלגוריתם כזה, אם הידע המוקדם שלנו נכון ובאמת קיימת ב־$\mathcal{H}$ היפותזה טובה, נוכל ללמוד.

חלק ב - אלגוריתמי למידה: בחמשת השבועות שלאחר מכן למדנו שורה של אלגוריתמי למידה:

- אלגוריתמי RLM לבעיות קמורות, בפרט רגרסייה לינארית (כלומר, רגרסייה בה ההיפותזות הן פונקציונאלים לינאריים)
- אלגוריתם ה־SVM לקלסיפיקציה בעזרת מפרדים לינאריים
- שיטות גרעין, המאפשרות לנו ללמוד במימד גבוה, וכפועל יוצא להרחיב את מחלקת ההיפותזות שלנו
- רשתות ניורונים
- עצי החלטה
- השכן הקרוב
- יערות אקראיים
- AdaBoost

משני החלקים עולה השיטה הבאה ללמוד:

1. בחרו אלגוריתם למידה
2. קבעו את הפרמטרים של האלגוריתם (הגרעין ופרמטר הרגולריזציה ב־SVM, הגודל והמבנה של הרשת עבור אלגוריתמים הלומדים רשתות ניורונים, הלומד החלש ומספר האיטרציות ב־AdaBoost וכו'.), כך שהוא יעשה אופטימיזציה על מחלקת היפותזות $\mathcal{H}$ בה אתם מעריכים ש־
   (א) יש היפותזה טובה.
   (ב) היא מספיק קטנה ביחס לכמות הנתונים שיש ברשותכם.
   (ג) אתם מערכים שהאלגוריתם יצליח למצוא בה היפותזה טובה.
3. הריצו את האלגוריתם ביחס לנתונים שאספתם.
4. בדקו את ההיפותזה שקיבלתם על מדגם בדיקה חדש.
5. אם הביצועים של ההיפותזה משביעים את רצונכם, סיימו. אחרת, חזרו לשלב 1, ושקלו את הפעלות הבאות
   (א) איסוף עוד נתונים
   (ב) שינוי הפרמטרים של האלגוריתם
   (ג) שינוי המימוש של האלגוריתם
   (ד) שינוי האלגוריתם

שגיאת קירוב, שגיאת הכללה ושגיאת האופטימיזציה - יחסי גומלין

בשימוש בלמידה חישובית, לפי השיטה שתוארה, יש לעשות הרבה מאד בחירות:

- ראשית, יש לבחור את אלגוריתם הלמידה
- לאחר מכן, צריך לבחור את הפרמטרים של האלגוריתם: הבחירות הללו ישפיעו על מחלקת ההיפותזות בה האלגוריתם יחפש היפותזה טובה.
- לבסוף, יש לבחור מימוש קונקרטי של האלגוריתם: GD, SGD או שיטה אחרת עבור SVM ורשתות ניורונים, השיטה המדוייקת בה עושים את הבחירה החמדנית בלמידת עצים, וכו'. עבור אלגוריתמים יוריסטיים (במיוחד עצים ורשתות ניורונים), הבחירה של המימוש הקונקרטי עשויה להשפיע על איכות ההיפותזה אותה ימצא האלגוריתם.

על מנת להעריך את ההשפעה של הבחירות השונות, ואת השיקולים שיש לקחת בחשבון, נוח לפרק את $L_{\mathcal{D}}(h)$ לשלושה רכיבים:

$$L_{\mathcal{D}}(h) = \underbrace{L_{\mathcal{D}}(h) - L_S(h)}_{\text{Estimation Error}} + \underbrace{L_S(h) - L_S(\mathcal{H})}_{\text{Optimization Error}} + \underbrace{L_S(\mathcal{H})}_{\text{Approximation Error}}$$

שגיאת ההכללה: שגיאת ההכללה היא ההפרש בין השגיאה האמיתית של ההיפותזה לבין השגיאה האמפירית שלה. הרכיב הזה יהיה קטן יותר ככל ש־$\mathcal{H}$ תהיה קטנה יותר, וככל שיהיו לנו יותר דוגמאות. עבור קלסיפיקציה בינארית, תורת ההכללה שפיתחנו מאפשרת לנו לחסום את שגיאת ההכללה באמצעות $\mathrm{VC}(\mathcal{H})$.

שגיאת הקירוב: שגיאת הקירוב היא השגיאה של ההיפותזה הטובה במחלקה. על מנת להקטין אותה, ניתן להקטין לנקוט בשתי דרכים:

- **ידע מוקדם.** אם אנו, או מומחה בבעיה הספציפית שעל הפרק, מצליח להצביע על מחלקה בה באמת יש היפותזה טובה, שגיאת הקירוב תהיה קטנה. לכן, לפני שמשתמשים באלגוריתמי למידה, כדאי להבין טוב את הבעיה שעומדת לפנינו, ולחשוב באיזו משפחה של פונקציות נוכל למצוא היפותזה הקרובה להיפותזה אותה אנו רוצים ללמוד.
- **שימוש במחלקה עשירה יותר.** ככל שהמחלקה אותה נבחר תכיל יותר היפותזות, שגיאת הקירוב תהיה קטנה יותר.

שגיאת האופטימיזציה: שגיאת האופטימיזציה היא ההפרש בין השגיאה האמפירית לבין השגיאה האמפירית של ההיפותזה הטובה המחלקה. אם נשתמש באלגוריתם ERM, השגיאה הנ"ל תהיה 0. למרבה הצער, ככל הנראה, עבור רוב המחלקות לא ניתן לממש את אלגוריתם ה־ERM ביעילות. לכן, אנו נאלצים להשתמש ביוריסטיקות. כפועל יוצא, שגיאת האופטימיזציה עשויה להיות חיובית. עבור אלגוריתמים המבוססים על תחליף קמור, ניתן לחסום את הסכום של שגיאת האופטימיזציה ושגיאת האפרוקסימציה ע"י שגיאת האפרוקסימציה של התחליף הקמור. לכן, במקרה של כשלון, אנו לכל הפחות נדע שהתחליף הקמור לא מספיק עשיר. למרבה הצער, עבור יתר אלגוריתמי הקלסיפיקציה, לא קיימות כיום שיטות המאפשרות לנתח את שגיאת האופטימיזציה, והיא עשוייה להשתנות מאד כאשר עוברים בין מימושים שונים של אלגוריתמים שונים. התורה הקיימת כיום לא מאפשרת לנו לעשות ניתוח מושכל של שגיאת האופטייזציה, ובפועל מסתמכים הרבה על נסיון העבר. למשל,

- ברשתות ניורונים, ככל שהרשת עמוקה יותר, שגיאת האימון תגדל. נעיר שכיום מצליחים להגיע לביצועים טובים עם רשתות בעומק עד 20 פחות או יותר.
- הנסיון מהשנים האחרונות מראה שהרבה פעמים SGD עובד טוב יותר מ־GD כאשר משתמשים ברשתות ניורונים.
- כמו כן, פרטובציות אקראיות קטנות של המשקלות בזמן האימון לעיתים משפרת את שגיאת האימון

הערות נוספות

נעיר מספר הערות נוספות שכדאי לקחת בחשבון כאשר משתמשים בלמידה

- **ניסוי וטעייה.** אידיאלית, היינו רוצים שיהיה לנו אלגוריתם למידה אחד, שנדע שהוא "הטוב ביותר", או לכל הפחות לא רחוק מהטוב ביותר. כאמור, זה רחוק מלהיות המצב, ולאלגוריתמים\פרמטרים\מימושים שונים יש יתרונות שונים. עבור בעיות למידה פשוטות, שימוש סטנדרטי באלגוריתם פשוט (נאמר, SVM ללא גרעין) עשוי להניב תוצאות טובות. עבור בעיות קשות יותר, מבין אוסף הבחירות שניתן לעשות כאשר משתמשים בלמידה, ככל הנראה הבחירה הראשונה שנעשה לא תהיה הטובה ביותר. לכן, שימוש בלמידה, בשונה משימוש באלגוריתמים, מצריך לא מעט ניסוי וטעייה.
- **ולידציה.** חלק מהפרמטרים של האלגוריתמים הם מורכבים למדי (למשל, המבנה של רשת הניורונים). לעומת זאת חלק מהפרמטרים הם פשוטים ו־"חד מימדיים" (למשל, פרמטר הרגולריזציה, או מספר האיטרציות ב־AdaBoost). במקרים כאלו ניתן להשתמש בולידציה על מנת לבחור את הערך המתאים לפרמטר.
- **התאמת יתר (Overfit) ובדיקת ההיפותזה הסופית.** לאחר תהליך הלמידה, כדאי עלינו לבדוק את ההיפותזה שייצרנו. לעיתים משתמשים באלגוריתמים ללא חסם הכללה, או עם חסם הכללה גרוע, ובמקרה כזה יש סכנה לעשות התאמת יתר. גם אם משתמשים באלגוריתמים עבורם מובטח שלא נעשה התאמת יתר, לעיתים בכל זאת עושים התאמת יתר, הנובעת, למשל, מהעובדה שאנו משתמשים שוב ושוב באלגוריתמים שונים על אותו מדגם אימון. מחסם הופדינג, על מנת לשערך את $L_{\mathcal{D}}(h)$ עדי כדי ϵ, אנו זקוקים ל־$O\left(\frac{1}{\epsilon^2}\right)$ דוגמאות. נעיר שהדוגמאות הללו צריכות להיות **דוגמאות חדשות, ובלתי תלויות** בדברים שעשינו קודם - בפרט, החסם לא תקף כאשר משתמשים בדוגמאות שאימנו עליהן את האלגוריתם, או אפילו השתמשנו בהן על מנת לבדוק \ לאמן אלגוריתמים אחרים בתהליך של ניסוי וטעייה. נעיר שבפועל לא משתמשים כל פעם בדוגמאות חדשות. לכן, על מנת להשאיר את תהליך הבדיקה אמין, בד"כ משתמשים במדגם בדיקה גדול יחסית (כ־50־10 אחוז מכלל הדוגמאות).
- **דרישות נוספות מההיפותזה הנלמדת.** לעיתים, אנו נרצה למצוא היפותזה עם תכונות מסויימות, מעבר להיותה היפותזה עם שגיאה נמוכה. למשל, היפותזה שיהיה ניתן להעריך במהירות, או שתתפוס מעט מקום בזיכרון. שיטות מסויימות, למשל שיטות הלומדות מפרידים לינאריים דלילים כמו AdaBoost, מייצרות היפותזות כאלו.

2 מה הלאה? למידת ייצוג

עד כה לא דנו כלל בדרך בה הקלטים מיוצגים. כלומר, הנחנו שהמרחב X הינו נתון. לייצוג של הקלטים יש השפעה רבה על הביצועים של ההיפותזה אותה נלמד. ככלל, ייצוג טוב יותר, המדגיש את החלקים ה־"חשובים" של הקלטים (ללא מידע "עודף") יאפשר לימוד טוב יותר: אם נתון לנו ייצוג כזה, יותר סביר שלהיפותזה פשוטה (כלומר כזו המגיעה ממחלקה קטנה ופשוטה) יהיו ביצועים טובים.

כיצד נייצר ייצוג טוב? ובכן, דרך אחת היא להשתמש במומחיות ובידע מוקדם. כלומר, לתת למומחה לבנות ייצוג "טוב" של הקלטים. לדוגמא:

- עבור קלטים טקסטואליים, הרבה פעמים Bag-Of-Word מהווה ייצוג טוב.
- עבור קלטים שהם קבצי שמע, הרבה פעמים ייצוג טוב מבוסס על פירוק לתדרים.

אנו כמעט ולא נדבר בקורס על ייצוגים המתאימים לתחומים ספציפיים, שכן דיון בהם שייך לתחום הקונקרטי (זיהוי שפה, זיהוי דיבור, ראייה ממוחשבת, וכו') אליו הם שייכים. עם זאת, מלבד שימוש במומחיות, לעיתים ניתן להשתמש בקלטים עצמם על מנת **ללמוד** ייצוג טוב. כלומר, בהינתן הרבה קלטים

$$x^1, \ldots, x^m \in X$$

אנו נרצה ללמוד מיפוי $\Psi : X \to X'$ כך שאם נייצג כל קלט $x \in X$ ע"י $\Psi(x) \in X'$, נקבל ייצוג "טוב" של הקלטים. כלומר, X' יהיה מרחב פשוט יותר מהמרחב המקורי (ממימד נמוך $\setminus$ עם מספר קטן של נקודות $\setminus$ ...), וההעתקה Ψ תשמר את החלקים ה"מהותיים" של נקודות המדגם, ולא "תאבד מידע חשוב". השבוע ובשבוע הבא נלמד על דרכים ללמוד מיפויים כאלו. נעיר מספר הערות לפני כן:

1. בלמידת ייצוג, אנו מסתכלים על דוגמאות **לא מתויגות**. כלומר על איברים ב־X, ולא ב־$X \times Y$. הרבה פעמים, דוגמא לא מתוייגת תהיה הרבה יותר זולה מדוגמה מתוייגת:

   (א) בראייה ממוחשבת, ניתן למצוא ברשת בקלות מיליארדי תמונות. לעומת זאת, יקר יותר להשיג תמונות מתוייגות.

   (ב) באופן דומה, בעיבוד שפה, יש המון טקסט ברשת ובמקומות האחרים, אך הרבה פחות טקסט מתוייג

   (ג) בביולוגיה, לעיתים, על מנת לקבל תיוג יש לעשות ניסוי

   לאור זאת, אחת המוטיבציות ללמידת ייצוג, היא ללמוד גם מהדוגמאות הלא מתוייגות, כלומר, להשתמש בהן על מנת לשפר את תהליך הלימוד. מכאן, לימוד ייצוג לפעמים נקרא **למידה לא מפוקחת (unsupervised learning)**, בשונה ממה שלמדנו עד עכשיו, הנכנס לקטגוריה של **למידה מפוקחת (supervised learning)**

2. בכל האלגוריתמים שנלמד, X יהיה $\mathbb{R}^n$. ונסמן ב־$d(x,y) = \|x-y\|$ את המרחק האוקלידי. כמו כן, כשנדבר על מדגם, הכוונה תהיה **למדגם לא מתוייג**

$$S = \{x^1, \ldots, x^m\} \subset \mathbb{R}^n$$

3. בדומה ללמידה מפוקחת, ניתן לדבר על מודל פורמאלי הדומה למודל PAC. אנו לא נעשה זאת, גם מפאת חוסר זמן, וגם בגלל שבניגוד ללמידה מפוקחת, עדיין אין מודל סטנדרטי.

ניתן לחלק את הגישות ללמידת ייצוג לארבעה סוגים.

1. **טרנספורמציות פשוטות.** דרך פשוטה, אך לעיתים אפקטיבית ללמוד ייצוג היא ע"י הפעלת טרנספורמציה פשוטה (בד"כ לינארית) על $\mathbb{R}^n$, הגורמת להתפלגות להיות "נקייה ומסודרת". דוגמא בסיסית היא **מרכוז וסטנדרטיזציה**. נקבע קואורדינטה $j \in [n]$. נסמן ב־

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^i, \quad \sigma_j = \sqrt{\sum_{i=1}^{m}\left(x_j^i - \mu_j\right)^2}$$

את התוחלת וסטיית התקן (האמפיריות) של הקאורדינטה ה־j. שימו לב שהגדלים הללו נלמדים מהמדגם הלא מתוייג שבידנו. נגדיר

$$\Psi(x) = \left(\frac{x_1 - \mu_1}{\sigma_1}, \ldots, \frac{x_n - \mu_n}{\sigma_n}\right)$$

הפעלת Ψ על המדגם מייצרת מדגם בו לכל קאורדינטה יש תוחלת 0 וסטיית תקן 1. לכן, במובן מסויים, כל הקואורדינטות הומוגניות, ואין קואורדינטות שיותר "בולטות" מהאחרות.

2. **הצברה (Clustering).** בשימוש בהצברה עבור למידת ייצוג, אנו מחפשים אוסף של $k \ll m$ נקודות (הנקראות **מרכזים**) $c^1, \ldots, c^k \in \mathbb{R}^n$ המייצגות טוב את המדגם. כלומר, לרוב הדוגמאות x^i קיים מרכז c^j הקרוב ל־x^i. אוסף כזה של מרכזים, מוביל לייצוג קומפקטי של X ע"י מספר סופי וקטן (k) של נקודות.

3. **הורדת מימד (Dimension Reduction).** בהורדת מימד אנו מחפשים מרחב לינארי V ממימד $k \ll d$ המייצג טוב את המדגם. למשל מרחב $V \subset \mathbb{R}^n$ בו לרוב הדוגמאות x^j קיים $v \in V$ הקרוב ל־x^j. מרחב V כזה, מוביל לייצוג קומפקטי של X ע"י מרחב לינארי ממימד נמוך.

4. **למידת מילון (Dictionary Learning).** בלמידת מילון אנו מחפשים אוסף של $k \ll m$ נקודות (הנקראות **מילים**) $v_1, \ldots, v_k \in \mathbb{R}^n$ המייצגות טוב את המדגם במובן שרוב הדוגמאות x^j הן, בקירוב, צירוף לינארי של מספר קטן של מילים.

אנו נלמד בעיקר הצברה, הורדת מימד, ולמידת מילון.

3 הצברה - אלגוריתם ה־k-means

יהא נתון מדגם $S = \{x^1, \ldots, x^m\} \subset \mathbb{R}^n$. אלגוריתמי k-means מנסים למצוא k מרכזים $C = \{c^1, \ldots, c^k\} \in \mathbb{R}^n$ הממזערים את סכום (או, באופן שקול, ממוצע) ריבועי המרחקים של נקודות המדגם מהמרכז הקרוב ביותר אליהן. קונקרטית, לכל נקודה $x \in \mathbb{R}^n$ נסמן ב־

$$d(x, C) := \min_{1 \le j \le k} d(x, c^j)$$

את המרחק בין x לבין המרכז הקרוב ביותר ל־x. אלגוריתמי k-means מנסים למזער את פונקציית המטרה

$$L^{\text{k-means}}(C) := \sum_{i=1}^{m} d^2(x^i, C)$$

הבעייה החישובית של למצוא אוסף מרכזים C הממזער את $L^{\text{k-means}}(C)$ הינה בעיה קשה חישובית. לבעיה קיימים אלגוריתמי קירוב, אך אנו נלמד על יוריסטיקה פופולארית, שלעיתים נקראית בעצמה k-means (למעשה, וראינט של היוריסטיקה שנלמד, הנקרא k-means++ הוא אלגוריתם קירוב עם יחס קירוב $O(\log(k))$). היוריסטיקה הנ"ל תחזיק אוסף של מרכזים

$$C = \{c^1, \ldots, c^k\} \subset \mathbb{R}^n$$

כמו כן, חלוקה של המדגם ל־k תתי קבוצות

$$S = C_1 \dot\cup \ldots \dot\cup C_k$$

הקבוצה **(צביר \ cluster)** C_j תכיל את הנקודות המתאימות למרכז c^j, כלומר, את אוסף נקודות המדגם שהמכרז הקרוב ביותר אליהן הוא c^j. היוריסטיקה תתבסס על שיטה הנקראית **Alternate Minimization** (עוד נפגוש אותה פעמיים בשלושת השבועות הקרובים): האלגוריתם יתחיל עם מרכזים אקראיים, ואז, בכל שלב יבצע:

- **אופטימיזציה על החלוקה:** האלגוריתם יעדכן את החלוקה כך שלכל j, C_j תכיל את הנקודות המקיימות $d(x, c^j) = d(x, C)$.
- **אופטימיזציה על המרכזים:** האלגוריתם יעדכן את המרכזים כך שלכל j, c^j ימזער את סכום ריבועי המרחקים בינו לבין הנקודות ב־C_j

כיצד יתבצע כל אחד משני השלבים? ובכן, בהינתן המרכזים, ברור כיצד לעדכן את החלוקה - פשוט נשים ב־C_j את כל הנקודות שהמרכז הקרוב ביותר אליהן הוא c_j. מה לגבי עדכון המרכזים בהינתן החלוקה? על מנת לעדכן את המרכז ה־j עלינו למצוא $c^j \in \mathbb{R}^n$ הממזער את

$$\sum_{x \in C_j} \|c^j - x\|^2$$

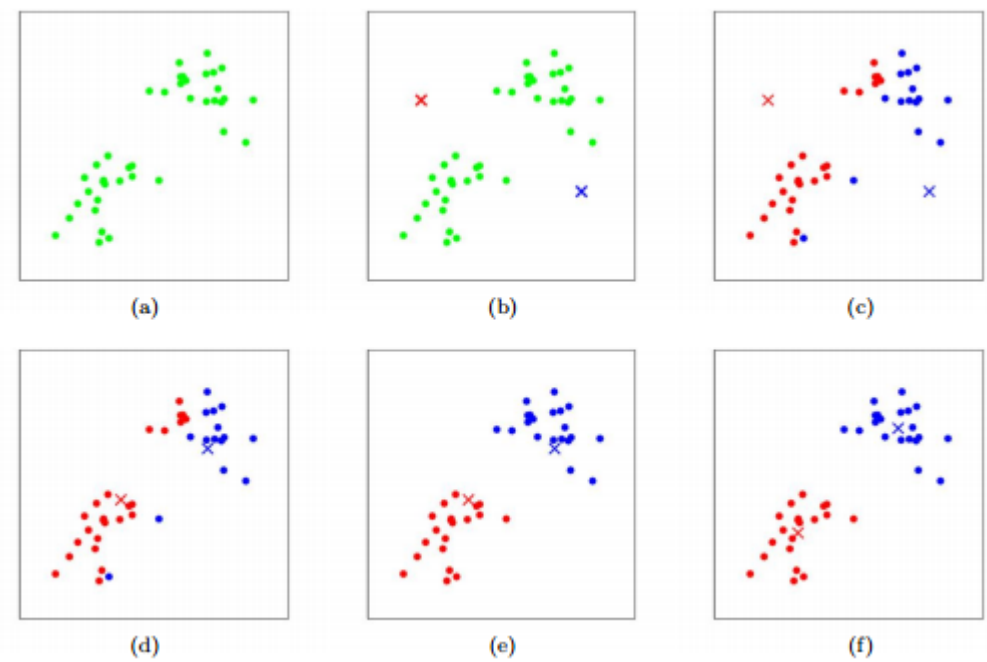
למרבה המזל, לבעיית האופטימיזציה הנ"ל קיים פתרון סגור - c_j הוא פשוט ממוצע הנקודות ב־C_j (תרגיל). נסכם:

k-means

פרמטרים: מספר מרכזים k, מספר איטרציות T.
קלט: מדגם $S = \{x^1, \ldots, x^m\} \subset \mathbb{R}^n$
1. עבור $j = 1, \ldots, k$
1.1. בחר את c_j להיות דוגמא אקראית מתוך S
2. עבור $t = 1, 2 \ldots, T$
2.2. לכל $j \in [k]$, עדכן $C_j = \{x^i \mid d(x^i, c^j) = d(x^i, \{c^1, \ldots, c^k\})\}$
2.3. לכל $j \in [k]$, עדכן $c^j = \frac{1}{|C_j|} \sum_{x^i \in C_j} x^i$
3. החזר את $c^1, \ldots, c^k$

3.1 הערות והדגמות

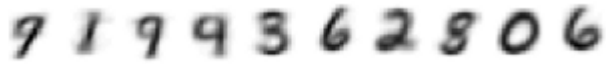
דוגמא סינטטית. בתמונה הבאה מופיעה הדגמה של ריצת האלגוריתם, עם שני מרכזים, על מדגם ב־$\mathbb{R}^2$:



בתמונה השמאלית העליונה מתואר המדגם. בתמונה (b) מופיעים שני מרכזים שנבחרו באקראי בשלב 1. של ריצת האלגוריתם (האלגוריתם שמתואר בתמונה מעט שונה בכך

שבחירת המרכזים שלו איננה בהכרח מתוך המדגם). בתמונות (f)-(c) מתוארות שני איטרציות של האלגוריתם.

דוגמא אמיתית - MNIST. בתמונה הבאה מופיעים 10 מרכזים שהתקבלו בהרצה של k-means על MNIST:



נשים לב שנראה שהמרכזים שנמצאו לא מייצגים מספיק טוב את המדגם - למשל, הספרות 5 ו־7 כלל לא מיוצגות. יכול להיות שאם היינו משתמשים ביותר מרכזים (נאמר, 100) כבר היינו מקבלים ייצוג טוב. בהקשר הזה, נעיר שהרבה פעמים הדוגמאות עצמן מאד מורכבות, ולא סביר שנוכל לייצגן באמצעות מספר קטן של מרכזים. במקרים כאלו, עדיין יכול להיות מועיל לעשות הצברה על **חלקים של הדוגמאות**. למשל, אם הדוגמאות הן תמונות בגודל 100×100, ניתן לעשות הצברה על על תמונות בגודל 10×10, המהוות ריבוע בתמונות המקוריות. ואז, ניתן לקבל ייצוג של התמונות הגדולות (100×100), למשל באופן הבא - נחלק כל תמונה כזו ל־100 תמונות קטנות בגודל 10×10, ונייצג כל תת־תמונה באמצעות המרכז המתאים.

אתחול ראשוני ו־k-means++. אנו הצענו דרך מסויימת לאתחל את המרכזים (פשוט לבחור k דוגמאות באקראי). ישנן עוד דרכים לאתחל את הנקודות. האלגוריתם k-means++ זהה לאלגוריתם שהצגנו, מלבד האתחול, שמתבצע באופן הבא:

k-means++ intialization

פרמטרים: מספר מרכזים k.
קלט: מדגם $S = \{x^1, \ldots, x^m\} \subset \mathbb{R}^n$
1. אתחל $C = \emptyset$
2. עבור $t = 1, \ldots, k$
2.1. בחר דוגמא ב־S באקראי, כך שהסיכוי לבחור את x_i הינו $\frac{d^2(x_i,C)}{\sum_{j=1}^m d^2(x_j,C)}$
2.2 הוסף את הדוגמא שנבחרה ל־C.
3. החזר את C

נשים לב שבשלב ה־t, שיטת האתחול הנ"ל תבחר בסיכוי גבוה יותר, דוגמאות ללא מרכז קרוב. ניתן להראות שאם C הוא הפלט של האתחול, ו־C^* היא קבוצת k המרכזים האופטימלית, אז

$$E[L^{\text{k-means}}(C)] \leq O\left(\log(k)\right) L^{\text{k-means}}(C^*)$$

(כאשר התוחלת היא על פני האקראיות הפנימית של האלגוריתם). לכן, אם אנו משתמשים בשיטת האתחול הנ"ל, אנו מקבלים אלגוריתם קירוב עם יחס קירוב $O\left(\log(k)\right)$.

3.2 אלגוריתמים נוספים ופונקציות מטרה נוספות

בנוסף, ישנם עשרות אלגוריתמי הצברה מעבר ל־k-means. תראו חלק מהם בתרגול. כמו כן, קיימות פונקציות מטרה נוספות אותן אלגוריתמי הצברה מנסים למזער, כאשר הם מחפשים מרכזים, לדוגמא:

$$L^{\text{k-medians}}(C) := \sum_{i=1}^{m} d(x^i, C)$$

$$L^{\text{k-center}}(C) := \max_{i \in [m]} d(x^i, C)$$

3.3 הצברה מעבר ללמידת ייצוג - מציאת חלוקה בעלת משמעות

אנו הצגנו אלגוריתמי הצברה בתור כלי המוצא ייצוג טוב, ע"י קירוב כל הנקודות במדגם באמצעות מספר קטן של נקודות. דרך אחרת, אפילו יותר מקובלת, להסתכל על הצברה היא בתור כלי המוצא חלוקה בעלת משמעות של אוסף אובייקטים. מציאה של חלוקות כאלו שימושית כמעט בכל תחום, למשל:

- **ביולוגיה:** ניתן לעשות הצברה על מנת למצוא חלוקה לסוגים של צמחים \ בע"ח \ גנים \ חלבונים \ ...
- **רשתות חברתיות:** ניתן להשתמש בהצברה על מנת למצוא קהילות
- **פסיכולוגיה:** ניתן להשתמש בהצברה על מנת לחלק בני אחדם לטיפוסים על סמך תכונות שלהם
- **עיבוד תמונה:** בהינתן תמונה, ניתן להשתמש בהצברה על מנת לחלק את התמונה לאובייקטים