

2. Machine Learning

2.1 Supervised Learning Algorithms

2.1.1 Support Vector Machines (SVM)

Support Vector Machine (SVM) הינו מודל למידה מונחית המשמש לניתוח נתונים לצורך סיווג, חיזוי ורגרסיה. המודל מקבל אוסף של דוגמאות מתויגות במרחב n -ממדי, ומנסה למצוא מישור המפריד בצורה טובה כמה שניתן בין דוגמאות האימון השייכות לקטגוריות השונות.

המסווג הנוצר באמצעות מודל SVM הינו לינארי, כאשר חלוקת הדוגמאות במרחב הווקטורי נעשית באופן כזה שיווצר מרווח גדול ככל האפשר בין המישור המפריד לבין הנקודות הממוקמות הכי קרוב אליו. מרווח זה מכונה שוליים (margin), כאשר בצד האחד של השוליים נמצאות דוגמאות עם label אחד, ובצד השני נמצאות הדוגמאות עם ה-label השני. את המישור המפריד ניתן לייצג באמצעות אוסף הנקודות \vec{x} המקיימות $\vec{w} \cdot \vec{x} - b = 0$, כאשר \vec{w} הוא וקטור נורמלי של המישור.

ננסח את האלגוריתם באופן פורמלי: נתון אוסף של n נקודות $(x_1, y_1) \dots (x_n, y_n)$, כאשר $y_i \in \{-1, 1\}$ מייצג את התיוג המתאים לדוגמה i , ו- $x_i \in \mathbb{R}^d$ הוא וקטור הפיצ'רים המתארים את דוגמה i . מודל ה-SVM מייצר מישור המפריד את המרחב לשני מרחבים שכל אחד מהם אמור להכיל בעיקר דוגמאות מסוג תיוג אחד. בנוסף, המודל מייצר שני מישורים מקבילים לו, אחד מכל צד, במרחק זהה וגדול ככל האפשר:

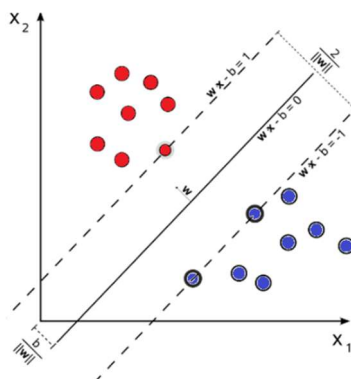
$$w^* = \underset{w}{\operatorname{argmin}} \left(\frac{1}{2} \|w\|^2 \right), s.t \forall i y_i (w \cdot x_i) \geq 1$$

כלומר, רוצים למצוא את וקטור המשקולות w^* המייצר שוליים $\text{margin} = \frac{1}{2} \|w\|^2$, כך שהדוגמאות מתויגות נכון $(y_i (w \cdot x_i) \geq 0)$ ואינן בתוך השוליים (לא מתקיים: $0 < y_i (w \cdot x_i) < \frac{1}{2} \|w\|^2$).

ישנם מספר גישות למציאת המפריד, ונפרט על כמה מהן.

Hard-Margin (hard SVM)

במצב הפשוט ביותר, המשוואה עבור כל אחד מצדיו של המפריד הינה פונקציה לינארית של המאפיינים וכל הדוגמאות אשר סווגו נכונה. מצב זה מכונה "הפרדה קשיחה" בו האלגוריתם מוצא את המישור עם השול הרחב ביותר האפשרי, ולא מאפשר לדוגמאות להיות בין הווקטורים התומכים. זוהי למעשה הפרדה מושלמת, והווקטורים התומכים הם למעשה הנקודות בקצוות השוליים, כפי שניתן לראות באיור:



איור 2.1 סיווג באמצעות אלגוריתם SVM עם מפריד בעל השוליים הרחבים ביותר. הקו האמצעי מייצג את המפריד, הקווים המקווקים מייצגים את מישורי השוליים. דוגמאות האימון המתלכדות עם מישורי השוליים נקראות וקטורים תומכים (support vectors), ומכאן נגזר שם האלגוריתם.

את המישורים בקצוות השוליים ניתן לייצג באמצעות $\vec{w} \cdot \vec{x} - b = 1$ or $\vec{w} \cdot \vec{x} - b = -1$. גאומטרית, המרחק בין שני המישורים הוא $\frac{2}{\|w\|}$, ולכן על מנת למקסם את המרחק הזה, יש מהביא למינימום את $\|w\|$. על מנת שדוגמאות האימון לא יכללו בשוליים המפרידים, יש להוסיף אילוץ לכל דוגמה i , באופן הבא:

$$y_i \cdot (\vec{w} \cdot \vec{x}_i - b) \geq 1$$

אילוץ זה מחייב שכל דוגמא תימצא בצד הנכון של המפריד. לכן, במקרה זה יש לקיים את הדרישה הבאה:

$$\min_{w,b} \|w\|^2$$

$$s.t. \quad y_i (\vec{w} \cdot \vec{x}_i - b) \geq 1 \quad \forall i = 1 \dots n$$

Soft-Margin (soft SVM)

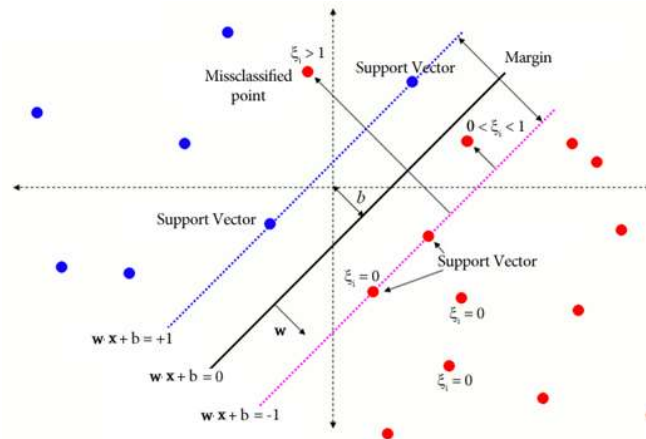
הפרדה מושלמת באמצעות מישור לינארי לעיתים קרובות איננה אפשרית, ולכן ניתן להרחיב את המודל כך שיאפשר לנקודות מסוימות לא להיות ב"צד" המתאים להן. הרחבה זו, היוצרת "הפרדה רכה", מאפשרת לטפל בבעיות שבהן אין הפרדה לינארית בין הקבוצות, כמו למשל שיש נקודות חריגות. משמעות ההרחבה היא שכל וקטור מפר לפחות אחד מהאילוץ, אך עם זאת, נרצה להגיע למצב בו האילוץ מופרים "כמה שפחות". הפרדה רכה יוצרת מצב בו יש trade-off בין רוחב השול לבין השגיאות ומציאת המשקלים האופטימליים של המסווג. בגרסה זו יש לרשום באופן מעט שונה את בעיית האופטימיזציה, כאשר מתווסף משתנה המתייחס לנקודות שאינן נמצאות בסיווג המתאים להן לפי המפריד:

$$\min \|w\|^2 + C \sum_{i=1}^n \xi_i$$

$$s.t. \quad y_i (\vec{w} \cdot \vec{x}_i - b) \geq 1 - \xi_i \quad \forall i = 1 \dots n, \xi_i \geq 0$$

לשם קבלת אינטואיציה, נשים לב לתפקיד המשתנים:

אם $\xi_i = 0$, מתקיים התנאי שנדרש בהפרדה קשיחה, כלומר הנקודה x_i גם נמצאת בצד הנכון של המפריד וגם מתקיימת הדרישה לשמירה על השוליים. אם $0 < \xi_i < 1$ אז הנקודה x_i נמצאת בצד הנכון של המפריד המסווג, אבל המסווג קרוב אליה, כך שהנקודה נמצאת בתוך השוליים. C הינו קבוע שאחראי על "ענישה" של דוגמאות שאינן בצד הנכון של המפריד. ערך C גבוה פירושו העדפת הסיווג הנכון על פני שוליים רחבים, ואילו C נמוך מעדיף הכללה (שוליים רחבים), גם במחיר שדוגמאות האימון הספציפיות אינן מסווגות נכון.



איור 2.2 סיווג באמצעות אלגוריתם SVM עם הפרדה רכה. המשתנה ξ_i שווה לאפס אם הנקודה ממוקמת בצד הנכון של המפריד. וגדול מאפס כאשר הנקודות נמצאות בצד הלא נכון של המפריד.

Non-linear Separation

מסווגים לינאריים מוגבלים ביכולת ההכללה שלהם. לכן, כאשר לא ניתן להפריד אוסף דוגמאות באמצעות מפריד לינארי, משתמשים ב"הפרדה א-לינארית". גישה זו מאפשרת להשתמש ב-SVM לסיווג לא לינארי, על ידי טרנספורמציה לא לינארית, כמו למשל "תעלול הגרעין" (Kernel Trick). בגישה זו מבצעים מיפוי לדאטה למרחב אחר, בו ניתן למצוא עברו בהפרדה לינארית, וממילא יהיה אפשר להשתמש באלגוריתם SVM. כך למשל, קיימת אפשרות ליצור מאפיינים חדשים על ידי העלאת ערכי המאפיינים הקיימים בחזקה מסוימת, הכפלתם בפונקציות טריגונומטריות וכו'.

באופן פורמלי, נחפש פונקציית מיפוי להעתקת מרחב $\psi: \mathcal{X} \rightarrow F$ כך שבמרחב F ניתן יהיה להפריד את הנתונים $\{\psi(x_i), y_i\}_{i=1}^N$ באמצעות מסווג לינארי. לשם כך, משתמשים בטריק קרנל שמקבל כקלט וקטורים במרחב המקורי ומחזיר את המכפלה הפנימית (dot product) של הווקטורים במרחב החדש (נקרא גם מרחב התכונות – feature space):

$$K(\vec{x}_i, \vec{x}_j) = \psi(\vec{x}_i)^T \psi(\vec{x}_j)$$

דוגמאות של פונקציות קרנל נפוצות:

קרנל לינארי:

$$K(\vec{x}_i, \vec{x}_j) = \vec{x}_i \cdot \vec{x}_j$$

זוהי הפונקציה הכי פשוטה, המוגדרת על ידי מכפלה פנימית של הווקטורים. במקרה זה מרחב התכונות ומרחב הקלט זהים ונחזור לפתרון בעזרת SVM לינארי:

קרנל פולינומי:

$$K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + c)^d$$

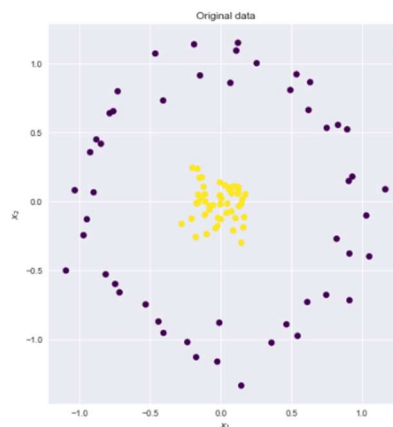
העתקה מהמרחב המקורי למרחב שמהווה פולינום ממעלה $d \geq 2$. $c \geq 0$ הוא פרמטר חופשי המשפיע על היחס בין סדר גבוה לעומת סדר נמוך בפולינום. כאשר $c = 0$, הקרנל נקרא הומוגני.

קרנל גאוסיאני:

$$K(\vec{x}_i, \vec{x}_j) = \exp(-\gamma(\vec{x}_i - \vec{x}_j)^2), \gamma > 0$$

הפרמטר γ ממלא תפקיד חשוב, ויש לבחור אותו בהתאם לבעיה העומדת בפנינו. אם הערך שלו קטן מאוד, האקספוננט ינהג כמעט באופן ליניארי וההטלה למרחב אחר מממד גבוה יותר יתחיל לאבד מכוחו הלא ליניארי. מצד שני, אם נעריך אותו יותר על המידה, הפונקציה לא תהיה סדירה וגבול ההחלטה יהיה רגיש מאוד לרעש בנתוני האימון.

המהות של טריק קרנל היא שניתן לבצע את ההעתקה גם מבלי לדעת מהי הפונקציה ψ , אלא הידיעה של K מספיקה. לצורך קבלת אינטואיציה והמחשה נביא דוגמה. נתון מערך הנתונים הבא:



ניתן לראות שלא ניתן להפריד בין הנקודות הצהובות לסגולות על ידי מישור הפרדה לינארי. לכן נחפש מרחב אחר, מאותו ממד או בעל ממד גבוה יותר, בו ניתן יהיה להפריד בין נקודות אלה באופן לינארי. לצורך כך נבצע את הפעולות הבאות:

- א. נמפה את התכונות המקוריות למרחב הגבוה יותר (מיפוי תכונות).
- ב. נבצע SVM לינארי במרחב החדש.
- ג. נמצא את קבוצת המשקולות התואמות את מישור גבול ההחלטה.
- ד. נמפה את מישור המפריד בחזרה למרחב הדו-ממדי המקורי כדי לקבל גבול החלטה לא ליניארי.

ישנם הרבה מרחבים מממדים גבוהים יותר בהם נקודות אלה ניתנות להפרדה ליניארית. נציג דוגמה אחת:

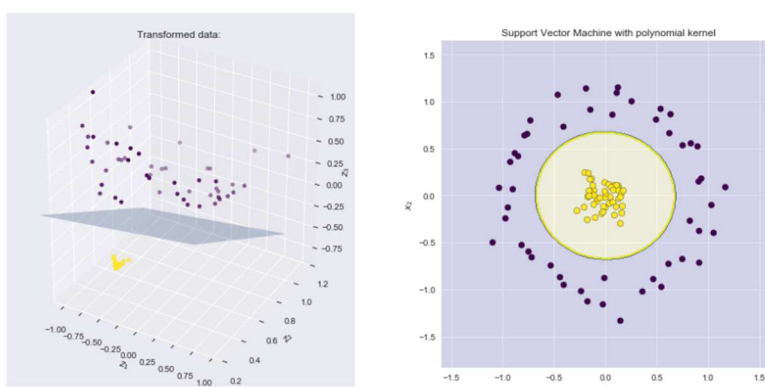
$$x_1, x_2 \mapsto z_1, z_2, z_3$$

$$z_1 = \sqrt{2}x_1x_2 \quad z_2 = x_1^2 \quad z_3 = x_2^2$$

למעשה נעזרנו בטריק קרנל. כאמור, בהינתן שקיימת פונקציה שממפה $\varphi: \mathbb{R}^n \rightarrow \mathbb{R}^m$ את הווקטורים ממרחב \mathbb{R}^n למרחב תכונות כלשהו \mathbb{R}^m , אז המכפלה הפנימית של x_1 ו- x_2 במרחב הזה היא $\varphi(x_1)^T \varphi(x_2)$. קרנל היא פונקציה K השייכת למכפלה פנימית זו, כלומר $K(x_1, x_2) = \varphi(x_1)^T \varphi(x_2)$. אם נוכל למצוא פונקציית קרנל המקבילה למפת התכונות שלעיל, נוכל להשתמש בפונקציה יחד עם SVM לינארי וכך לבצע את החישובים ביעילות.

מתברר שמרחב התכונות שלעיל תואם את השימוש בקרנל פולינומי ידוע: $K(x, x') = (x^T x')^d$. נבחר $d = 2$, נסמן $x = (x_1, x_2)^T$ ונקבל:

$$\begin{aligned} K\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} x'_1 \\ x'_2 \end{pmatrix}\right) &= (x_1x'_1 + x_2x'_2)^2 = \\ &= 2x_1x'_1x_2x'_2 + (x_1x'_1)^2 + (x_2x'_2)^2 = (\sqrt{2}x_1x_2x'_1x'_2) \begin{pmatrix} \sqrt{2}x'_1x'_2 \\ x_1'^2 \\ x_2'^2 \end{pmatrix} \\ K\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} x'_1 \\ x'_2 \end{pmatrix}\right) &= \varphi(x)^T \varphi(x') \\ \varphi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) &= \begin{pmatrix} \sqrt{2}x_1x_2 \\ x_1^2 \\ x_2^2 \end{pmatrix} \end{aligned}$$



איור 2.3 שימוש ב-SVM לצורך הפרדה לאחר ביצוע Kernel trick. המעגל באיור הימני ממופה למישור הפרדה לינארי במרחב מממד גבוה יותר, כפי שניתן לראות באיור השמאלי. ניתן לראות שאחרי המיפוי שנעשה בעזרת kernel trick, הנקודות אכן מופרדות בצורה לינארית.

2.1.2 Naive Bayes

סיווג בייסיאני הוא מודל המשתמש בחוק בייס על מנת לסווג אובייקט $x \in \mathbb{R}^n$ בעל n פיצ'רים לאחת מ- K קטגוריות אפשריות. יחד עם השימוש בחוק בייס, המודל מניח "נאיביות" – בהינתן סיווג של אובייקט מסוים, אין תלות בין הפיצ'רים השונים שלו.

נניח שיש מודל המקבל וקטור פיצ'רים בינאריים {כאב ראש, משתעל, חום גבוה}, ומסווג האם אדם בעל תכונות אלה חולה בשפעת או לא. באופן כללי ניתן לומר שיש תלות בין שיעול לבין חום גבוה, כלומר העובדה שיש לאדם חום מעלה את ההסתברות שהוא גם משתעל. למרות זאת, ניתן להניח באופן "נאיבי" שאם כבר יודעים שאדם חולה בשפעת, אז כבר אין יותר תלות בין היותו משתעל להיותו בעל חום. באופן פורמלי, אמנם סביר להניח שמתקיים $p(\text{חום}|\text{משתעל}) > p(\text{חום})$, אך ניתן להניח נאיביות ולקבל: $p(\text{שפעת}|\text{משתעל}) = p(\text{שפעת, חום}|\text{משתעל})$.

באופן כללי סיווג בייסיאני נאיבי מניח שבהינתן הסיווג של אובייקט מסוים, הפיצ'רים שלו בלתי תלויים. הנחה זו כמובן לא תמיד מדויקת, וממילא גם ערכי ההסתברויות הנובעים ממנה ומשמשים לסיווג אינם מדויקים, אך ההנחה מקלה

מאוד על חישוב ההסתברויות של הסיווג הבייסיאני, ובמקרים רבים תחת ההנחה זו התקבלו תוצאות סיווג. הסיבה להצלחת המודל נעוצה בכך שבבעיית סיווג העיקר הוא למצוא את הסיווג הסביר ביותר לאובייקט (שפעת או לא-שפעת לבדק בדוגמה), ולא דווקא לקבל הסתברות מדויקת לכל סיווג. במקרים רבים למרות שההסתברות הנובעת מההנחה הנאיבית אינה מדויקת עבור שני סיווגים אפשריים, היא בכל זאת שומרת על סדר ההסתברות שלהם.

נתבונן בוקטור פיצ'רים $x \in \mathbb{R}^n = (x_1, \dots, x_n)$, היכול להיות שייך לאחת מ-K קטגוריות $y = (y_1, \dots, y_K)$. ההתפלגות הפריורית $p(y_k)$ ידועה, ובנוסף ידועות ההתפלגויות המותנות של הפיצ'רים בהנתן הסיווג $p(x_i|y_k)$. בעזרת הנתונים האלה רוצים לסווג את x לאחת מהקטגוריות, כלומר למצוא את y_k שעבורו הביטוי $p(y_k|x)$ הוא מקסימלי. באופן פורמלי ניתן לנסח זאת כך:

$$y = \arg \max_k p(y_k|x), k = 1 \dots K$$

בשביל למצוא את y_k האופטימלי ניתן להיעזר בחוק בייס:

$$p(y_k|x) = \frac{p(y_k, x)}{p(x)}$$

המכנה לא תלוי ב- k , ולכן מספיק למצוא את y_k שעבור המונה מקסימלי. לפי כלל השרשרת מתקיים:

$$\begin{aligned} p(y_k, x) &= p(y_k, x_1, \dots, x_n) = p(x_1|y_k, x_2, \dots, x_n) \cdot p(y_k, x_2, \dots, x_n) \\ &= p(x_1|y_k, x_2, \dots, x_n) \cdot p(x_2|y_k, x_3, \dots, x_n) \cdot p(y_k, x_3, \dots, x_n) \\ &= \dots = p(x_1|y_k, x_2, \dots, x_n) \cdot p(x_2|y_k, x_3, \dots, x_n) \cdots p(x_{n-1}|y_k, x_n) \cdot p(x_n|y_k) p(y_k) \end{aligned}$$

כעת נשתמש בהנחת הנאיביות, לפי בהינתן הסיווג y_k , אין תלות בין הפיצ'רים. לפי הנחה זו נוכל לפשט את הביטוי:

$$\begin{aligned} &= p(x_1|y_k) \cdot p(x_2|y_k) \cdots p(x_{n-1}|y_k) \cdot p(x_n|y_k) p(y_k) \\ &= p(y_k) \prod_{i=1}^n p(x_i|y_k) \end{aligned}$$

בביטוי זה כל האיברים ידועים, ולכן כל שנותר זה רק להציב את הנתונים ולקבל את y_k עבורו ביטוי זה הכי גדול:

$$y = \arg \max_k p(y_k) \prod_{i=1}^n p(x_i|y_k)$$

בדוגמה שהובאה לעיל, הפיצ'רים קיבלו ערכים בידיים, ולכן היה ניתן לחשב את ההסתברות המותנית של כל פיצ'ר $p(x_i|y_k)$ על ידי ספירת כמות הפעמים שמופיע כל פיצ'ר באוכלוסייה הנדגמת ולחלק בגודל המדגם. עבור ערכים רציפים (כמו למשל מחיר מניה, גובה של אדם וכדו'), אין אפשרות לחשב כך את ההסתברות המותנית. במקרים כאלה יש להניח התפלגות מסוימת עבור המדגם, ולחשב את הפרמטרים של ההתפלגות שיטות שונות (למשל בעזרת נראות מרבית – MLE). עבור מדגם המתפלג נורמלית, ההסתברות המותנית היא גאוסיאן:

$$p(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}}$$

כאשר μ_y, σ_y^2 הם הפרמטרים של ההתפלגות, וכאמור הם משוערים בעזרת MLE או שיטת שערך אחרת. אם ההתפלגות היא לא נורמלית, ניתן להשתמש באלגוריתם [Kernel density estimation](#) עבור שערך ההתפלגות. גישה אחרת להתמודדות עם פיצ'רים יכולים לקבל ערכים רציפים היא לבצע דיסקרטיזציה לערכים אותם הפיצ'רים יכולים לקבל.

במקרה [המולטינומי](#), בו ההתפלגות היא רב ממדית ומציינת תוצאה של סדרה בלתי תלויה, יש לחשב את הנראות באופן המתאים להתפלגות מולטינומית. בכדי להבין את החישוב נביא קודם בדוגמה – נניח ורוצים לבנות מודל סיווג בייסיאני המזהה הודעות ספאם. נתונות 12 הודעות, מתוכן 8 אמיתיות ו-4 ספאם. כעת נניח וכל ההודעות מורכבות מאוסף של ארבע מילים, בהתפלגות הבאה:

Real (R) – {Dear, Friend, Lunch, Money} = {8, 5, 3, 1}.

Spam (S) – {Dear, Friend, Lunch, Money} = {2, 1, 0, 4}.

נחשב את הנראות – ההסתברות של כל מילה בהינתן הסיווג:

$$p(\text{Dear}|R) = \frac{8}{17}, p(\text{Friend}|R) = \frac{5}{17}, p(\text{Lunch}|R) = \frac{3}{17}, p(\text{Money}|R) = \frac{1}{17}$$

$$p(\text{Dear}|S) = \frac{2}{7}, p(\text{Friend}|S) = \frac{1}{7}, p(\text{Lunch}|S) = 0, p(\text{Money}|S) = \frac{4}{7}$$

כעת נבחן מה ההסתברות שהצירוף "Dear friend" הוא מהודעה אמיתית (הצירוף הוא למעשה התפלגות מולטינומית, כיוון שהוא מכיל שתי מילים שאין בין ההסתברויות שלהן קשר ישיר):

$$p(\text{Dear friend is R}) = p(R) \cdot p(\text{Dear}|R) \cdot p(\text{Friend}|R) = 0.67 \cdot 0.47 \cdot 0.29 = 0.09$$

$$p(\text{Dear friend is S}) = p(S) \cdot p(\text{Dear}|S) \cdot p(\text{Friend}|S) = 0.33 \cdot 0.29 \cdot 0.14 = 0.01$$

ממספרים אלה ניתן להסיק שהצירוף "Dear friend" אינו ספאם.

באופן כללי, עבור וקטור פיצ'רים $x \in \mathbb{R}^n = (x_1, \dots, x_n)$, הנראות מחושבת באופן הבא:

$$p(x|y_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p(y_{ki})^{x_i}$$

על הציר הלוגריתמי, בעזרת נוסחה זו ניתן לבנות מסווג לינארי:

$$p(y_k|x) = \frac{p(y_k, x)}{p(x)} \propto p(y_k) \cdot \prod_i p(y_{ki})^{x_i}$$

$$\rightarrow \log p(y_k|x) \propto \log p(y_k) \cdot \prod_i p(y_{ki})^{x_i} = \log p(y_k) + \sum_i x_i \cdot \log p(y_{ki}) \equiv b + w^T x$$

החיסרון בשימוש במסווג בייסיאני נאיבי בבעיות מולטינומיות נעוץ בכך שיש הרבה צירופים שלא מופיעים יחד בסט האימון, ולכן הנראות שלהם תמיד תהיה 0, מה שפוגם באמינות התוצאות.

מקרה דומה להתפלגות מולטינומית הוא מקרה בו הפיצ'רים הם משתני ברנולי, המקבלים ערכים בינאריים. במקרה זו הנראות הינה:

$$p(x|y_k) = \prod_{i=1}^n p_i^{x_i} (1 - p_i)^{1-x_i}$$

עבור דאטה לא מאוזן, ניתן להשתמש באלגוריתם שנקרא complement naive Bayes (CNB). לפי אלגוריתם זה, במקום לקחת את $\arg \max_k p(y_k) \prod_{i=1}^n p(x_i|y_k)$, לוקחים את המינימום של הפונקציה ההופכית:

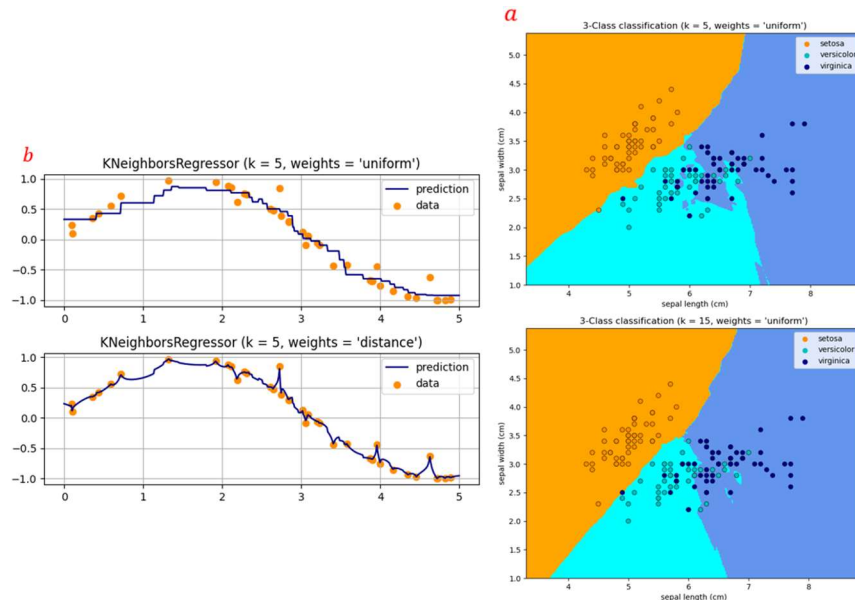
$$\arg \min_k p(y_k) \prod_{i=1}^n \frac{1}{p(x_i|y_k)}$$

שימוש באלגוריתם זה הוכח כיעיל במקרים בהם הדאטה אינו מאוזן והביצועים של מסווגים בייסיאנים אחרים (גאוסיאני או מולטינומי) היה לא מספיק טוב.

2.1.3 K-Nearest Neighbors (K-NN)

אלגוריתם השכן הקרוב הינו אלגוריתם של למידה מונחית, בו נתונות מספר דוגמאות ובנוסף ידוע ה-label של כל אחת מהן. אלגוריתם זה מתאים הן לבעיות סיווג (שיוך נקודה חדשה למחלקה מסוימת) והן לבעיות רגרסיה (נתינת ערך מאפיין לנקודה חדשה). האלגוריתם הינו מודל חסר פרמטרים, והוא מבצע סיווג לנתונים בעזרת הכרעת הרוב. עבור כל נקודה במדגם, המודל בוחן את ה-labels של K הנקודות הקרובות אליו ביותר, ומסווג את הנקודה לפי ה-label שקיבל את מרבית הקולות. מספר הנקודות הקרובות, K, הוא היפר-פרמטר שנקבע מראש.

אלגוריתם השכן הקרוב הוא אחד המודל הנפוצים והפשוטים ביותר בלמידת מכונה, וכאמור בנוסף לסיווג הוא מתאים גם לבעיות רגרסיה. המודל יפעל בצורה דומה בשני המקרים, כאשר ברגרסיה יתבצע שקלול של ממוצע בין השכנים הקרובים, ולא הכרעת הרוב, כלומר, התוצאה לא תהיה סיווג ל-label מסוים לפי הערך הנפוץ ביותר בקרב K השכנים הקרובים, אלא חישוב ממוצע של כל ה-labels השכנים. התוצאה המתקבלת היא ערך רציף, המייצג את הערכים בסביבת התצפית. ניתן להתחשב במרחק של כל שכן מהתצפית בצורה שווה (uniform), וניתן לתת משקל שונה לכל שכן בהתאם למרחק שלו מהנקודה אותה רוצים לחשב, כך שכלל ששכן מסוים קרוב יותר לנקודה אותה רוצים לחשב כך הוא יותר ישפיע עליה, ביחס של הופכי המרחק בין השכן לבין הנקודה (distance).



איור 2.4 (a) סיווג בעזרת אלגוריתם K-NN: מסווגים את המרחב לאזורים בהתאם ל-K השכנים הקרובים ביותר, כך שאם תבוא נקודה חדשה היא תהיה מסווגת בהתאם לצבע של האזור שלה, הנקבע כאמור לפי השכנים הקרובים ביותר. ניתן לראות שיש הבדל בין ערכי K שונים, וככל ש-K יותר גבוה ככה האזורים יותר חלקים ויש פחות מובלעות. (b) רגרסיה בעזרת אלגוריתם K-NN: קביעת ערך ה- γ בהתאם ל-K השכנים הקרובים ביותר. ניתן לתת משקלים שווים לכל השכנים, או לתת משקל ביחס למרחק של כל שכן מהנקודה אותה רוצים לחשב.

לעיתים נאמר על המודל שהוא "עצלן". הסיבה לכך היא שבשלב האימון לא מתבצע תהליך משמעותי, מלבד השמה של המשתנים וה-labels כאובייקטים של המחלקה, כלומר כל נקודה משויכת למחלקה מסוימת. עקב כך, כל מדגם האימון (או רובו) נדרש לצורך התחזית, מה שעשוי להפוך את המודל לאיטי כאשר יש הרבה דאטה. למרות זאת, המודל נחשב לאחד המודלים הקלאסיים הבולטים, בזכות היתרונות שלו. הוא פשוט וקל לפירוש, עובד היטב עם מספר רב של מחלקות, ומתאים לבעיות רגרסיה וסיווג. בנוסף הוא נחשב אמין במיוחד, כיוון שהוא לא מניח הנחות לגבי התפלגות הנתונים (כמו רגרסיה לינארית למשל).

מנגד, יש לו מספר חסרונות. עקב העובדה שהוא דורש את כל נתוני האימון בשביל התחזית, הוא עשוי להיות איטי כאשר מדובר על דאטה עשיר. מסיבה זו הוא גם אינו יעיל מבחינת זיכרון. מכיוון שהמודל דורש את כל נתוני האימון לצורך המבחן, כושר ההכללה שלו עשוי להיפגם (Generalization). ניקח לדוגמה מורה של כיתה בבית ספר, המנסה לסווג את התלמידים למספר קבוצות. אם יעשה זאת לפי צבע שיער ועיניים, לדוגמא, סביר להניח שלא יתקשה בכך; אם לעומת זאת הוא ינסה לסווג לפי צבע שיער, עיניים, חולצה, מכנסיים, נעליים, וכו' – סביר שיתקל בקושי. במצב כזה, כל תלמיד רחוק מרעהו באופן שווה כיוון שאין שני תלמידים שזהים לחלוטין בכל הפרמטרים, מה שמקשה על חישוב המרחק. בעיה זו מכונה קללת הממדיות (Course of dimensionality), ולכן מומלץ להיעזר באמצעים להורדת הממד (Dimensionality reduction).

קושי נוסף הקיים במודל הוא הצורך בבחירת ה-K הנכון, מטלה שעשויה להיות לא קלה לעיתים. בכל מימוש של אלגוריתם השכן הקרוב, K הינו היפר-פרמטר שצריך להיקבע מראש. היפר פרמטר זה קובע את מספר הנקודות אשר האלגוריתם יתחשב בהן בעת בחירת סיווג התצפית. בחירת היפר-פרמטר קטן מדי, לדוגמה $K = 1$, יכולה לגרום למצב בו המודל מותאם יתר על המידה לנתוני האימון, מה שמוביל לדיוק גבוה בנתוני האימון, ודיוק נמוך בנתוני המבחן. מן העבר השני, כאשר K גבוה מדי, למשל $K = 100$, נוצר המצב ההפוך – מודל שמתחשב יותר מדי בדאטה ולא מצליח למצוא הכללה נכונה לסיווג. מומלץ לבחור K אי-זוגי בגלל אופן הפעולה של האלגוריתם – הכרעת

הרוב. כאשר בוחרים K זוגי, עלולים להיתקל במצב של שוויון אשר עשוי להוביל לתוצאה מוטעית, ולכן כדי להימנע מתיקו כדאי לבחור K אי זוגי.

כמו אלגוריתמים רבים מבוססי מרחק, אלגוריתם השכן הקרוב רגיש לערכים קיצוניים (Outliers) ושימוש באלגוריתם ללא טיפול בערכים קיצוניים עשוי להוביל לתוצאות מוטות. מלבד זאת, חשוב לנרמל את הנתונים לפי שימוש במודל. הסיבה לכך היא שהאלגוריתם מבוסס מרחק; במצב זה, ייתכנו מרחקים בין תצפיות אשר עשויים להשפיע על החלטת המודל, למרות שמרחקים אלו הם חסרי משמעות לצורך הסיווג. דוגמה לכך היא משתנה שעושה שימוש ביחידות מידה שונות (מיליסקילומטרים). ההחלטה האם להשתמש בקילומטרים או במילים עלולה להטות את תוצאת המודל, למרות שבפועל לא השתנה דבר.

השיטה הנפוצה ביותר למדידת מרחק בין משתנים רציפים היא מרחק אוקלידי – עבור שתי נקודות במישור, המרחק ביניהם יחושב לפי הנוסחה: $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. במידה ומדובר במשתנים בדידים, כגון טקסט, ניתן להשתמש במטריקות אחרות כגון מרחק המינג, המודד את מספר השינויים הדרושים בכדי להפוך מחרוזת אחת למחרוזת שנייה, ובכך למדוד את הדמיון ביניהן.

לפני שימוש באלגוריתם השכן הקרוב, יש הכרח לוודא שהמחלקות מאוזנות. במידה ומספר דוגמאות האימון באחת המחלקות גבוה מאשר בשאר המחלקות, האלגוריתם ייטה לסווג למחלקה זאת. הסיבה לכך היא שבשל מספרן הגדול, מחלקה זו צפויה להיות נפוצה הרבה יותר בקרב K השכנים של כל תצפית. הדבר עשוי להביא לתוצאות מוטות, ולכן יש לוודא מראש שאין איזון בין המחלקות השונות.

2.1.4 Quadratic\Linear Discriminant Analysis (QDA\LDA)

Quadratic Discriminant Analysis הינו מודל נוסף לסיווג של דאטה לקבוצות שונות, המניח שבהינתן סיווג של אובייקט מסוים – מתקבלת התפלגות נורמלית, כלומר בהינתן $y_k, k \in \{1, \dots, K\}$ מתקיים:

$$x|y_k \sim N(\mu_k, \Sigma_k)$$

ובאופן מפורש, עבור $x \in \mathbb{R}^{n \times d}$ הפילוג המותנה הוא:

$$p(x|y = k; \mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_k|}} \exp\left(-\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k)\right)$$

בעזרת הנחה זו, ניתן למצוא מסווג אופטימלי עבור $y = \arg \max_k p(y_k|x)$. לפי חוק בייס:

$$p(y_k|x) = \frac{p(x|y = k)p(y)}{p(x)}$$

המכנה קבוע ביחס ל- y_k , ואת $p(y)$ ניתן לשערך בקלות על פי השכיחות של כל label במדגם – $p(y = k) = \frac{1_{y=k}}{n}$. את הביטוי הנוסף במונה – $p(x|y = k)$ – שכאמור מתפלג נורמלית, ניתן לשערך בעזרת הנראות מרבית (MLE). נסמן את הפרמטרים של המודל ב: $\theta = \{\mu_1, \Sigma_1, \dots, \mu_K, \Sigma_K\}$, ונקבל:

$$\theta_{MLE} = \arg \max_{\theta} p(x|y) = \arg \max_{\theta} \log p(x|y; \theta)$$

$$= \arg \max_{\theta} \log \sum_{i=1}^n p(x_i|y_i; \theta)$$

ניתן לפרק את הסכום לפי ה-label של כל דגימה:

$$= \arg \max_{\theta} \log \sum_{i \in y_i=1} p(x_i|y_i = 1; \theta) + \log \sum_{i \in y_i=2} p(x_i|y_i = 2; \theta) + \dots + \log \sum_{i \in y_i=K} p(x_i|y_i = K; \theta)$$

כעת בשביל לחשב פרמטרים עבור כל y_k מספיק להסתכל על הדגימות עבורן $y = k$, כלומר:

$$\theta_{kMLE} = \arg \max_{\theta_k} \log \sum_{i \in y_i=k} p(x_i|y_i = k; \theta_k)$$

על ידי גזירה והשוואה ל-0 ניתן לחשב את הפרמטרים האופטימליים:

$$\mu_k = \frac{\sum_{i \in y_i=k} x_i}{\sum_i \mathbb{1}_{y_i=k}}$$

$$\Sigma_k = \frac{\sum_{i \in y_i=k} (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_i \mathbb{1}_{y_i=k}}$$

ניתן לשים לב שהתוחלת μ_k היא למעשה ממוצע הדגימות עבורן $y = k$. בעזרת הפרמטרים המשוערים ניתן לבנות את המסווג:

$$y = \arg \max_k p(y_k | x; \mu_k, \Sigma_k) = \arg \max_k \log p(x | y = k) p(y)$$

$$= \arg \max_k -\frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + \log p(y)$$

עבור המקרה בו מטריצת ה-covariance היא אלכסונית, כלומר אין תלות בין משתנים שונים, מתקבל המסווג הבייסיאני הגאוסיאני (תוצאה זו הגיונית כיוון שהמסווג הבייסיאני מניח שבהינתן סיווג של אובייקט מסוים אין יותר תלות בין המשתנים).

עבור המקרה הבינארי, בו $y \in \{0, 1\}$, מתקבל סיווג בצורה של משוואה ריבועית:

$$y = 1 \Leftrightarrow -\frac{1}{2} \log |\Sigma_1| - \frac{1}{2} (x - \mu_1)^T \Sigma_1^{-1} (x - \mu_1) + \log p(y = 1) > -\frac{1}{2} \log |\Sigma_0| - \frac{1}{2} (x - \mu_0)^T \Sigma_0^{-1} (x - \mu_0) + \log p(y = 0)$$

נסמן:

$$a = \frac{1}{2} (\Sigma_1^{-1} - \Sigma_0^{-1})$$

$$b = \Sigma_1^{-1} \mu_1 - \Sigma_0^{-1} \mu_0$$

$$c = \frac{1}{2} (\mu_0^T \Sigma_0^{-1} \mu_0 - \mu_1^T \Sigma_1^{-1} \mu_1) + \log \frac{p(y = 1)}{p(y = 0)} - \log \frac{\sqrt{|\Sigma_1|}}{\sqrt{|\Sigma_0|}}$$

ונקבל:

$$y = 1 \Leftrightarrow x^T a x + b^T x + c > 0$$

וזוהו משטח הפרדה ריבועי.

Linear Discriminant Analysis הינו מקרה פרטי של Quadratic Discriminant Analysis, בו מניחים כי מטריצת ה-covariance זהה לכל ה-labels, כלומר $\Sigma_k = \Sigma$. במקרה זה מתקבל:

$$\log p(x | y = k) p(y) = -\frac{1}{2} (x - \mu_k)^T \Sigma^{-1} (x - \mu_k) + \log p(y)$$

הביטוי $(x - \mu_k)^T \Sigma^{-1} (x - \mu_k)$ נקרא מרחק מהלונביס, והוא מבטא את מידת הקשר בין x לבין μ_k תוך כדי התחשבות בשונות של כל משתנה. למעשה ניתן להסתכל על מסווג LDA כמסווג המשייך אובייקט ל-label עבורו המרחק על פי מטריקת מהלונביס הוא הכי קטן. על ידי גזירה והשוואה ל-0 מתקבל השערוך:

$$\mu_k = \frac{\sum_{i \in y_i=k} x_i}{\sum_i \mathbb{1}_{y_i=k}}$$

$$\Sigma_k = \frac{1}{n} \sum_i (x_i - \mu_k)(x_i - \mu_k)^T$$

והמסווג המתקבל הינו:

$$y = \arg \max_k p(y_k | x; \mu_k, \Sigma) p(y) = \arg \max_k -\frac{1}{2} (x - \mu_k)^T \Sigma^{-1} (x - \mu_k) + \log p(y = k)$$

$$= \arg \max_k -x^T \Sigma^{-1} \mu_k + \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log p(y = k)$$

ניתן לסמן:

$$a = \Sigma^{-1} \mu_k$$

$$b = \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log p(y = k)$$

ומתקבל מסווג לינארי (ומכאן השם של האלגוריתם):

$$y = \arg \max_k ax^T + b$$

מסווג זה מחלק כל שני אזורים בעזרת מישור לינארי, כאשר בסך הכל יש K קווי הפרדה. עבור המקרה הבינארי מתקיים:

$$a = \Sigma^{-1}(\mu_1 - \mu_0)$$

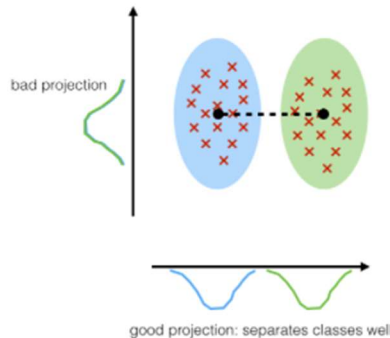
$$b = \frac{1}{2}(\mu_0^T \Sigma^{-1} \mu_0 - \mu_1^T \Sigma^{-1} \mu_1) + \log \frac{p(y = 1)}{p(y = 0)}$$

והסיווג הינו:

$$y = 1 \Leftrightarrow a^T x + b > 0$$

אלגוריתם LDA פשוט יותר מאלגוריתם QDA כיוון שיש פחות פרמטרים לשערך, אך יש לו שני חסרונות עיקריים – הוא לא גמיש אלא לינארי, ובנוסף הוא מניח שמטריצת ה-covariance זהה לכל ה-labels, מה שיכול לגרום לשגיאות בסיווג. כדי להתמודד עם הבעיה השנייה ניתן להשתמש באלגוריתמים המנסים למצוא את מטריצת ה-covariance שתביא לסיווג הטוב ביותר האפשרי (למשל Oracle Shrinkage Approximating ו-Ledoit-Wolf estimator).

באופן גרפי ניתן להסתכל על אלגוריתם LDA כמציאת כיוון ההפרדה בו יש את השונות הגדולה ביותר בין שתי התפלגויות נורמליות, ובנוסף יש בו את ההפרדה המקסימלית בין הקבוצות השונות. לאחר מציאת הקו האופטימלי, ניתן לחשב את ההתפלגויות של הקבוצות השונות כהתפלגויות נורמליות על הישר המאונך לקו ההפרדה:



איור 2.5 אלגוריתם LDA באופן גרפי: מציאת הכיוון של ההתפלגויות והטלת המידע על הציר האנכי לכיוון ההפרדה.

2.2 Unsupervised Learning Algorithms

2.2.1 K-means

אלגוריתם K-means הינו אלגוריתם של למידה לא מונחית, בו מתבצעת תחזית על נתונים כאשר ה-label אינו נתון. אלגוריתם זה מתאים לבעיות של חלוקה לאשכולות (Clustering), ובנוסף יכול לשמש בשלב הצגת וניקוי הנתונים (EDA). עבור כל נקודה במדגם, המודל ממזער את סכום ריבוע המרחקים (WCSS) מכל מרכז אשכול (סנטרואיד - centroid), ולאחר תהליך של התכנסות – נקבעים האשכולות והסנטרואידים הסופיים. מספר האשכולות הנדרש הוא היפר-פרמטר שנקבע מראש. כמו כל האלגוריתם השייכים ללמידה הבלתי-מונחית, ב-K-means לא מתבצע אימון, ולמעשה התחזית מתבצעת על כל הדאטה הנתון.

סנטרואיד הוא מונח מתחום הגיאומטריה, והוא מתאר את הממוצע האריתמטי של כל הנקודות שמתפרסות על פני צורה כלשהי. באופן אינטואיטיבי ניתן לחשוב על סנטרואיד כנקודת איזון של צורה גיאומטרית כלשהי, כך שאם ננסה להניח צורה, משולש לדוגמה, באופן מאוזן, הסנטרואיד הוא הנקודה שבה המשולש יתאזן ולא ייפול לאחד הצדדים.

בפועל, סביר שהצורות איתן מתמודדים במציאות יותר מורכבות ממשולש. במצב כזה, הסנטרואיד יהיה הנקודה בה סכום המרחקים של כל נקודה באשכול מהסנטרואיד יהיה מינימלי. כלומר, המודל ימקם את מרכזו של כל אשכול כך שסכום המרחקים של כל הנקודות מהסנטרואיד יהיה נמוך ככל האפשר. למעשה, זוהי ההגדרה הבסיסית של K-means: אלגוריתם מבוסס סנטרואידים הממזער את סכום ריבוע המרחק של כל הנקודות באשכול. מדד זה נקרא WCSS, והוא מדד משמעותי ביותר בקרב אלגוריתמים שמבצעים חלוקה לאשכולות, K-means בפרט. הסיבה לחזקה במשוואה היא שאנו רוצים להגביר את ההשפעה של המרחק, מעין "עונש" לתצפיות רחוקות מהמרכז.

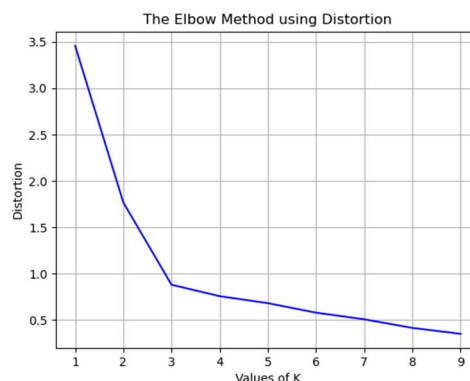
מדד WCSS הוא אחד הדרכים המקובלות ביותר להעריך את תוצאות החלוקה לאשכולות ב-K-means. היתרון של מדד זה הוא האפשרות לראות באופן כמותי את מידת ההצלחה של המודל, כלומר לקבל מספר ממשי שמכמת את הצלחת המודל. מנגד, WCSS הוא מספר ללא תחום מסוים והוא דורש פרשנות, כיוון שהערך והמשמעות שלו משתנים ממודל למודל. ערך מסוים יכול להיחשב תוצאה טובה במקרה מסוים, ובמקרה אחר זאת עשויה להיחשב תוצאה רעה מאוד. ניתן להשוות WCSS בין מודלים אך ורק כאשר יש להם את אותו מספר אשכולות ואותו מספר תצפיות. באופן פורמלי, ערך זה מחושב באופן הבא:

$$WCSS = \sum_{j=1}^k \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2$$

כאשר K הוא מספר האשכולות, ו-n הוא מספר הנקודות במדגם.

ישנו trade-off בין השאיפה למזער את מדד ה-WCSS ובין מספר האשכולות הרצוי: ככל שמספר האשכולות גדול יותר, כך ה-WCSS יקטן. הדבר מתיישב עם ההיגיון – פיזור סנטרואידים רבים (כלומר, חלוקה ליותר אשכולות) על פני הנתונים יוביל לכך שבהכרח סכום המרחקים של התצפיות מהסנטרואידים יקטן או לא ישתנה. כיוון שתצפית משויכת לסנטרואיד הקרוב אליה ביותר, אם התווסף סנטרואיד שקרוב לנקודה מסוימת – ה-WCSS קטן. ואם הסנטרואיד רחוק מכל שאר הנקודות במדגם יותר מהסנטרואידים הקיימים – חלוקת התצפיות לאשכולות לא תשתנה, וערך ה-WCSS לא ישתנה.

לכן מצד אחד, נרצה לבחור K גדול שימזער את ה-WCSS; מצד שני, הסיבה שהשתמשנו ב-K-means מלכתחילה היא בכדי לפשט את הנתונים למספר סביר של אשכולות, כזה שיאפשר לנו לערוך אנליזה נוחה. שיטת המרפק (Elbow method) היא טכניקה שמשמשת לפתרון סוגייה זו. הרעיון הוא לבחור את ה-K הקטן ביותר שממנו השיפור במדד ה-WCSS הוא מתון במידה סבירה. שיטה זו היא היוריסטית ואין דרך חד משמעית לקבוע שה-K הנבחר הוא האופטימלי. בדרך זו ניתן לשכנע מדוע ה-K שנבחר הוא הנכון, אך ההחלטה הסופית נתונה לשיקול דעתו של המשתמש.

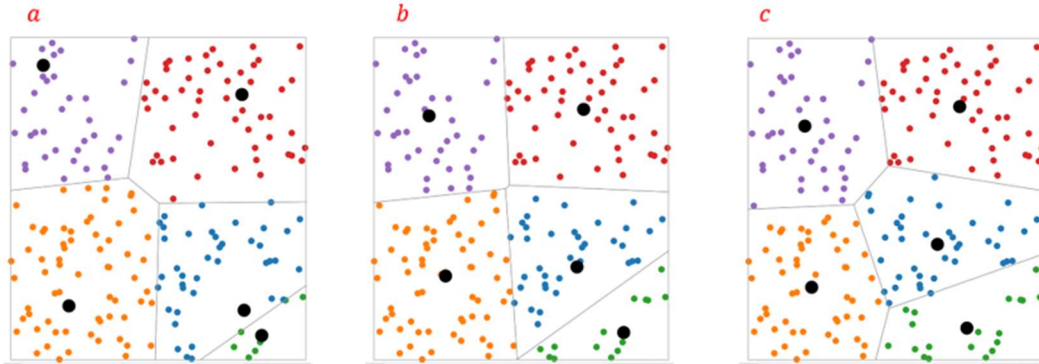


אירור 2.6 Elbow method – שיטה היוריסטית למציאת מספר האשכולות האופטימלי. בדוגמה זו ניתן לראות שבמעבר של K מ-2 ל-3 יש ירידה משמעותית בערך של ה-WCSS. המעבר מ-3 ל-4 לעומת זאת מוביל לשינוי זניח ב-WCSS (וכך גם במעברים הבאים). לכן ניתן להסיק שבמקרה כזה בחירה של K = 3 הינה בחירה טובה.

כאמור, האלגוריתם מחלק את הנתונים לאשכולות בדרך שממזערת את סך ריבועי המרחקים של כל תצפית ממרכז האשכול. באופן פורמלי האלגוריתם מתבצע ב-4 שלבים:

א. **אתחול:** המודל מציב את הסנטרואידים באופן רנדומלי.

- ב. **שיור**: כל תצפית משויכת לסנטרואיד הקרוב אליה ביותר.
 ג. **עדכון**: הסנטרואיד מוזז שכך שה-WCSS של המודל ימוזער.
 ד. חזרה על שלבים ב, ג עד אשר הסנטרואידים לא זזים לאחר העדכון, כלומר יש התכנסות.



איור 2.7 K-means. (a) **אתחול** 6 סנטרואידים באופן רנדומלי. (b) **שיור** כל נקודה לסנטרואיד הקרוב ביותר אליה, **ועדכון** הסנטרואידים לפי מדד ה-WCSS. (c) **חזרה** על b עד להתכנסות.

K-means ידוע בכך שהוא אלגוריתם פשוט ומהיר. לרוב, הבחירה הראשונה בפתרון בעיות של חלוקה לאשכולות תהיה ב-K-means. עם זאת, לאלגוריתם ישנם גם חסרונות. ראשית, בחירת ה-K הנכון עשויה להוות אתגר במרבית המקרים. בנוסף, האלגוריתם רגיש מאוד לערכים קיצוניים (Outliers). אופן הפעולה של האלגוריתם מאפשר לו ליצור אשכולות רק בצורה של ספירות, והדבר אינו אופטימלי בחלק מן המקרים.

בעיה נוספת יכולה להתעורר בבחירת המיקום הראשוני של הסנטרואידים – כיוון שהבחירה היא רנדומלית, ניתן להיקלע להתכנסות במינימום מקומי שהוא אינו המינימום הגלובלי. כדי להתמודד עם בעיה זו ניתן להשתמש באלגוריתם K++. בשלב ראשון האלגוריתם בוחר למקם סנטרואיד אחד באופן רנדומלי. לכל תצפית, האלגוריתם מחשב את המרחק בין התצפית לסנטרואיד הקרוב אליה ביותר. לאחר מכן, תצפית רנדומלית נבחרת להיות הסנטרואיד החדש. התצפית נבחרת בהתאם להתפלגות משוקללת של המרחקים, כך שכל שתצפית יותר רחוקה – כך גובר הסיכוי שהיא תבחר. שני השלבים האחרונים נמשכים עד שנבחרו K סנטרואידים. כאשר כל הסנטרואידים מוקמו, מבצעים K-means עם דילוג על שלב האתחול (השלב בו ממקמים את הסנטרואידים). K++ מוביל להתכנסות מהירה יותר, ומוריד את הסיכוי להתכנס לאופטימום מקומי.

2.2.2 Mixture Models

אלגוריתם K-means מחלק n נקודות ל-K קבוצות על פי מרחק של כל נקודה ממרכז מסוים. בדומה ל-K-means גם אלגוריתם mixture model הוא אלגוריתם של clustering, אך במקום להסתכל על כל קבוצה של נקודות כשייכות למרכז מסוים, המודל משייך נקודות להתפלגויות שונות. המודל מניח שכל קבוצה היא למעשה דגימות של התפלגות מסוימת, וכל הדאטה הוא ערבוב דגימות ממספר התפלגויות. הקושי בשיטה זה הוא האתחול של כל קבוצה – כיצד ניתן לדעת על איזה דוגמאות לנסות ולמצוא התפלגות מסוימת? עקב בעיה זו, לעיתים משתמשים קודם באלגוריתם K-means על מנת לבצע חלוקה ראשונית לקבוצות, ולאחר מכן מנסים למצוא לכל קבוצה של נקודות התפלגות מסוימת.

ראשית נניח שיש k אשכולות, אזי נוכל לרשום את ההסתברות לכל אשכול:

$$p(y = i) = \alpha_i, i = 1, \dots, k$$

וכמובן לפי חוק ההסתברות השלמה מתקיים $\sum_i \alpha_i = 1$.

בנוסף נניח שכל אשכול מתפלג נורמלית עם פרמטרים $\theta_i = (\mu_i, \sigma_i)$, אזי נקודה השייכת לאשכול i מקיימת:

$$x|y = i \sim \mathcal{N}(\mu_i, \sigma_i), i = 1 \dots k$$

אם מגיעה נקודה חדשה ורוצים לשייך אותה לאחד האשכולות, אז צריך למעשה למצוא את האשכול i שעבורו הביטוי $p(y = i|x)$ הוא הכי גדול. לפי חוק בייס מתקיים:

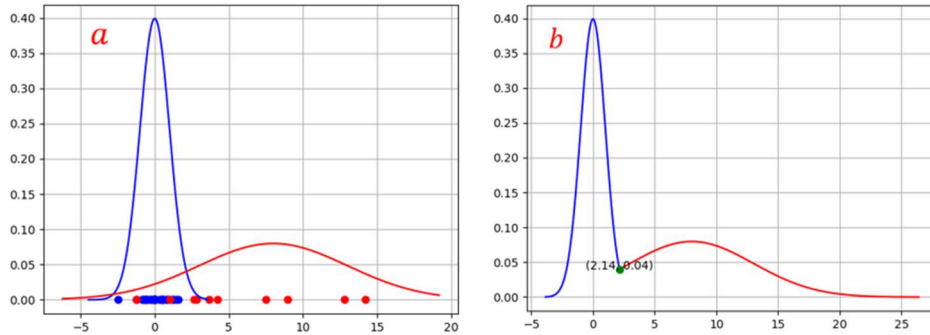
$$p(y = i|x) = \frac{p(y = i) \cdot p(x|y = i)}{p(x)}$$

המכנה למעשה נתון, כיוון שההתפלגות של כל אשכול ידועה ונותר לחשב את המכנה:

$$f(x) = f(x; \theta) = \sum_i p(y = i) f(x|y = i) = \sum_i \alpha_i \mathcal{N}(x; \mu_i, \sigma_i)$$

ובסך הכל:

$$p(y = i|x) = \frac{\alpha_i \cdot \mathcal{N}(x; \mu_i, \sigma_i)}{\sum_j \alpha_j \mathcal{N}(x; \mu_j, \sigma_j)}$$

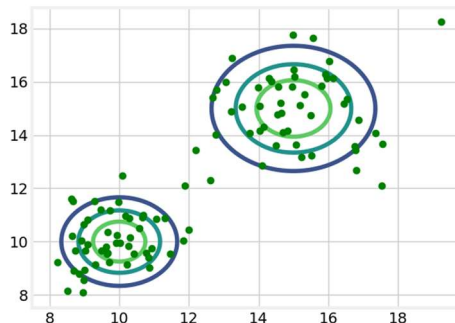


איור 2.8 (a) תערובת של שני גאוסיאנים בממד אחד: בשלב ראשון מחלקים את הנקודות לשני אשכולות ומתאימים לכל אשכול התפלגות מסוימת. במקרה זה אשכול אחד (מסומן בכחול) הותאם להתפלגות $\mathcal{N}(0,1)$, ואשכול אחד (מסומן באדום) הותאם להתפלגות $\mathcal{N}(8,5)$. (b) נקודה חדשה x תסווג לאשכול הכחול אם $x < 2.14$, כיוון שבתחום זה $\mathcal{N}(0,1) > \mathcal{N}(8,5)$. באופן דומה, הנקודה x תסווג לאשכול האדום אם $x > 2.14$, כיוון שבתחום זה $\mathcal{N}(0,1) < \mathcal{N}(8,5)$.

כאמור, כדי לשייך נקודה חדשה x לאחד מהאשכולות, יש לבדוק את ערך ההתפלגות בנקודה החדשה. ההתפלגות שעברה ההסתברות $p(x)$ היא הגדולה ביותר, היא זאת שאליה תהיה משויכת הנקודה. ההתפלגויות יכולות להיות בחד ממד, אך הן יכולות להיות גם בממד יותר גבוה. למשל אם מסתכלים על מישור, ניתן להתאים לכל אשכול התפלגות נורמלית דו-ממדית. במקרה ה- n ממדי, התפלגות נורמלית $X \sim \mathcal{N}(\mu, \Sigma)$ היא בעלת הצפיפות:

$$f_X(x_1, \dots, x_n) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

כאשר $|\Sigma|$ הוא הדטרמיננטה של מטריצת ה-covariance.



איור 2.9 תערובת של שני גאוסיאנים דו-ממדי: אשכול אחד מתאים לגאוסיאן עם וקטור תוחלות $\mu_1 = [10, 10]$ ומטריצת covariance: $\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$; והאשכול השני מתאים לגאוסיאן עם וקטור תוחלות $\mu_1 = [15, 15]$ ומטריצת covariance: $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

כיוון שהאלגוריתם mixture model מספק התפלגויות, ניתן להשתמש בו כמודל גנרטיבי, כלומר מודל שיועד לייצר דוגמאות חדשות. לאחר התאמת התפלגות לכל אשכול, ניתן לדגום מההתפלגויות השונות ובכך לקבל דוגמאות חדשות.

2.2.3 Expectation-maximization (EM)

אלגוריתם מקסום התוחלת הינו שיטה איטרטיבית למציאת הפרמטרים האופטימליים של התפלגויות שונות, במקרים בהם אין נוסחה סגורה למציאת הפרמטרים. נתבונן על מקרה של Mixture of Gaussians, ונניח שיש אשכול מסוים המתפלג נורמלית עם תוחלת ושונות $\theta = (\mu, \sigma)$, ומשויכות אליו n נקודות. כדי לחשב את ההתפלגות של אשכול זה ניתן להשתמש בלוג הנראות המרבית:

$$L(\theta|x_1, \dots, x_n) = \log \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i - \mu)^2}{2\sigma^2}} = \sum_{i=1}^n \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{(x_i - \mu)^2}{2\sigma^2}$$

כדי למצוא את הפרמטרים האופטימליים ניתן לגזור ולהשוות ל-0:

$$\frac{\partial L(\theta)}{\partial \mu} = \sum_{i=1}^n \frac{x_i - \mu}{\sigma^2} \rightarrow \mu_{MLE} = \frac{1}{n} \sum_{i=1}^n x_i$$

$$\frac{\partial L(\theta)}{\partial \sigma^2} = \frac{1}{2\sigma^2} \left(-n + \frac{1}{\sigma^2} \sum_{i=1}^n (x_i - \mu)^2 \right) \rightarrow \sigma_{MLE}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

כעת נניח ויש k אשכולות וכל אחד מתפלג נורמלית. כעת סט הפרמטרים אותם צריך להעריך הינו:

$$\theta = \{\mu_1, \dots, \mu_k, \sigma_1^2, \dots, \sigma_k^2, \alpha_1, \dots, \alpha_k\}$$

עבור מקרה זה, הלוג של פונקציית הנראות המרבית יהיה:

$$L(\theta|x_1, \dots, x_n) = \log \prod_{i=1}^n \sum_{j=1}^k \alpha_j \mathcal{N}(x_i, \mu_j, \sigma_j^2) = \sum_{i=1}^n \log \left(\sum_{j=1}^k \alpha_j \mathcal{N}(x_i, \mu_j, \sigma_j^2) \right)$$

אם נגזור ונשווה ל-0 נקבל בדומה למקרה הפשוט:

$$\sum_{i=1}^n \frac{1}{\sum_{j=1}^k \alpha_j \mathcal{N}(x_i, \mu_j, \sigma_j^2)} \alpha_j \mathcal{N}(x_i, \mu_j, \sigma_j^2) \frac{(x_i - \mu_j)}{\sigma_j^2} = 0$$

נוסחה זו אינה ניתנת לפתרון אנליטי, ולכן יש הכרח למצוא דרך אחרת בכדי לחשב את הפרמטרים האופטימליים של ההתפלגויות הרצויות. נתבונן בחלק מהביטוי שקיבלנו:

$$\frac{1}{\sum_{j=1}^k \alpha_j \mathcal{N}(x_i, \mu_j, \sigma_j^2)} \alpha_j \mathcal{N}(x_i, \mu_j, \sigma_j^2) = \frac{p(y_i = j) \cdot p(x_i | y = j)}{p(x_i)} = p(y_i = j | x_i) \equiv w_{ij}$$

קיבלנו למעשה את הפוסטרירור y_i (האשכול אליו רוצים לשייך את x_i), אך הוא לא נתון אלא הוא חבוי. כדי לחשב את המבוקש ננחש ערך התחלתי ל- θ ובעזרתו נחשב את y_i , ואז בהינתן y_i נבצע עדכון לפרמטרים – נבחן מהו סט הפרמטרים שמסביר בצורה הטובה ביותר את האשכולות שהתקבלו בחישוב ה- y_i . באופן פורמלי שני השלבים מנוסחים כך:

E-step – בהינתן אוסף נקודות x וערך עבור הפרמטר θ נחשב את האשכול המתאים לכל נקודה, כלומר כל נקודה x_i תותאם לאשכול מסוים y_i . עבור כל הנקודות y_i נחשב תוחלת ובעזרתה נגדיר את הפונקציה $Q(\theta, \theta_0)$, כאשר θ הוא פרמטר חדש ו- θ_0 הוא סט הפרמטרים הנוכחי:

$$Q(\theta, \theta_0) = \sum_{i=1}^n \sum_{j=1}^k p(y_i = j | x_i; \theta_0) \log p(y_i = j, x_i; \theta) = \sum_{i=1}^n \sum_{j=1}^k w_{ij} \log p(y_i = j, x_i; \theta)$$

$$\sum_{i=1}^n \mathbb{E}_{p(y_i | x_i; \theta_0)} \log p(y_i = j, x_i; \theta)$$

M-step – מחשבים את הפרמטר θ שיביא למקסימום את $Q(\theta, \theta_0)$ ואז מעדכנים את θ_0 ל- θ החדש:

$$\theta = \arg \max_{\theta} Q(\theta, \theta_0)$$

$$\theta_0 \leftarrow \theta$$

חוזרים על התהליך באופן איטרטיבי עד להתכנסות.

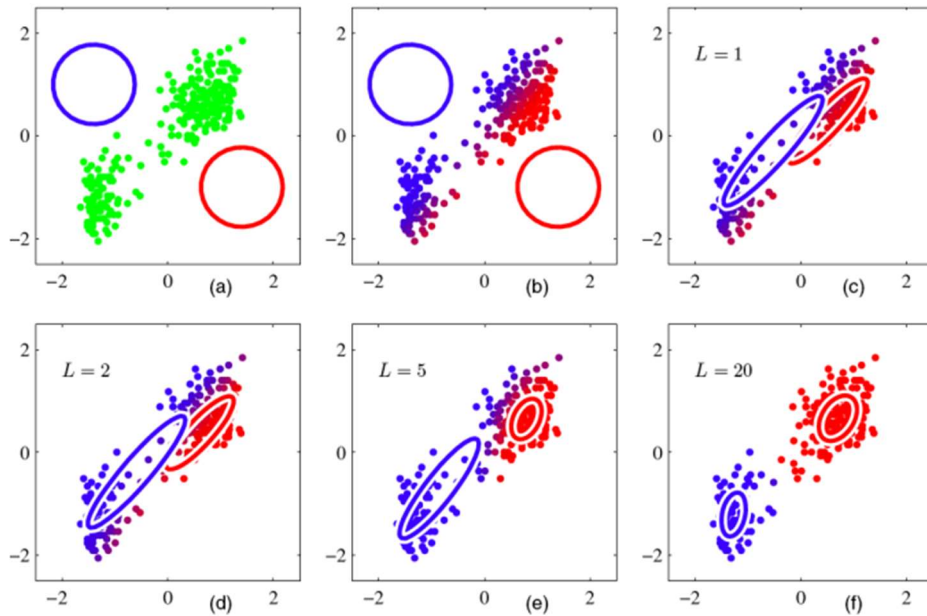
עבור Mixture of Gaussians נוכל לחשב באופן מפורש את הביטויים:

$$\begin{aligned}
 Q(\theta, \theta_0) &= \sum_{i=1}^n \sum_{j=1}^k w_{ij} \log p(y_i = j, x_i; \theta) \\
 &= \sum_{i=1}^n \sum_{j=1}^k w_{ij} \log p(y_i = j; \theta) + \sum_{i=1}^n \sum_{j=1}^k w_{ij} \log p(x_i | y_i = j; \theta) \\
 &= \sum_{i=1}^n \sum_{j=1}^k w_{ij} \log \alpha_j + \sum_{i=1}^n \sum_{j=1}^k w_{ij} \log \mathcal{N}(\mu_j, \sigma_j^2) \\
 &= \sum_{i=1}^n \sum_{j=1}^k w_{ij} \log \alpha_j - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^k w_{ij} \left(\log \sigma_j^2 + \frac{(x_i - \mu_j)^2}{\sigma_j^2} \right)
 \end{aligned}$$

וכעת ניתן לגזור ולמצוא אופטימום:

$$\begin{aligned}
 \hat{\alpha}_j &= \frac{1}{n} \sum_{i=1}^n w_{ij} \\
 \hat{\mu}_j &= \frac{\sum_{i=1}^n w_{ij} x_i}{\sum_{i=1}^n w_{ij}} \\
 \hat{\sigma}_j^2 &= \frac{\sum_{i=1}^n w_{ij} (x_i - \mu_j)^2}{\sum_{i=1}^n w_{ij}}
 \end{aligned}$$

עבור התפלגויות שונות שאינן בהכרח נורמליות יש לחזור לביטוי של $Q(\theta, \theta_0)$ ולבצע עבורו את האלגוריתם.



איור 2.10 20 איטרציות של אלגוריתם EM. מתחילים מניחוש אקראי של ההתפלגויות, ובכל איטרציה יש שיפור כך שההתפלגויות מייצגות בצורה יותר טובה את הדאטה המקורי.

נוכיח שהאלגוריתם משתפר בכל איטרציה, כלומר שעבור כל (θ, θ_0) מתקיים: $\log p(x; \theta) \geq \log p(x; \theta_0)$

$$\log p(x; \theta) = \sum_y p(y|x; \theta_0) \log p(x; \theta) = \sum_y p(y|x; \theta_0) \frac{\log p(x, y; \theta)}{\log p(y|x; \theta)}$$

$$\begin{aligned}
&= \sum_y p(y|x; \theta_0) (\log p(x, y; \theta) - \log p(y|x; \theta)) \\
&= \sum_y p(y|x; \theta_0) \log p(x, y; \theta) - p(y|x; \theta_0) \log p(y|x; \theta)
\end{aligned}$$

נשים לב שהאיבר הראשון הוא בדיוק $Q(\theta, \theta_0)$. האיבר השני לפי הגדרה הוא האנטרופיה של ההתפלגות $p(x|y; \theta_0)$:

$$H(\theta, \theta_0) = - \sum_y p(y|x; \theta_0) \log p(y|x; \theta_0)$$

כעת עבור שני ערכים שונים של θ מתקיים:

$$\begin{aligned}
\log p(x; \theta) - \log p(x; \theta_0) &= Q(\theta, \theta_0) + H(\theta, \theta_0) - Q(\theta_0, \theta_0) - H(\theta_0, \theta_0) \\
&= Q(\theta, \theta_0) - Q(\theta_0, \theta_0) + H(\theta, \theta_0) - H(\theta_0, \theta_0)
\end{aligned}$$

לפי [אי-שוויון גיבס](#) מתקיים $H(\theta, \theta_0) \geq H(\theta_0, \theta_0)$, לכן:

$$\log p(x; \theta) - \log p(x; \theta_0) \geq Q(\theta, \theta_0) - Q(\theta_0, \theta_0)$$

ולכן עבור כל עדכון של θ שמביא לאופטימום את $Q(\theta, \theta_0)$, הביטוי $Q(\theta, \theta_0) - Q(\theta_0, \theta_0)$ יהיה חיובי וממילא יהיה שיפור ב- $\log p(x; \theta)$.

2.2.4 Hierarchical Clustering

אלגוריתם נוסף של למידה לא מונחית עבור חלוקת n נקודות ל- K אשכולות נקרא Hierarchical Clustering, והוא מחולק לשתי שיטות שונות:

agglomerative clustering – בשלב הראשוני מגדירים כל נקודה כאשכול, ואז בכל פעם מאחדים שני אשכולות ובכך מורידים את מספר האשכולות ב-1, עד שמגיעים ל- K אשכולות. האיחוד בכל שלב נעשה על ידי מציאת שני האשכולות הקרובים ביותר זה לזה ואיחודם לאשכול אחד. ראשית יש לבחור מטריקה לחישוב מרחק בין שתי נקודות (למשל מרחק אוקלידי, מרחק מנהטן ועוד), ולאחר מכן לחשב מרחק בין האשכולות, כאשר יש מספר דרכים להגדיר את המרחק הזה, למשל:

complete-linkage clustering: $\max\{d(a, b) : a \in A, b \in B\}$.

single-linkage clustering: $\min\{d(a, b) : a \in A, b \in B\}$.

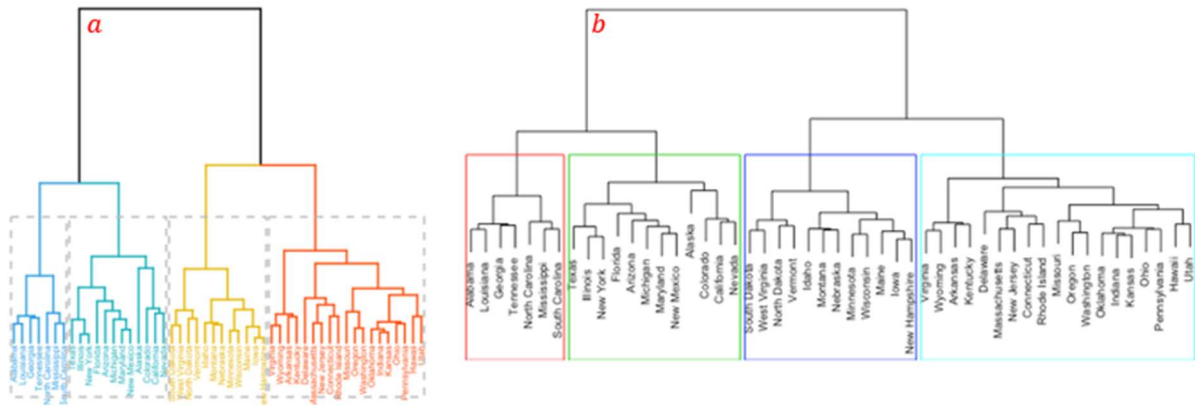
Unweighted average linkage clustering (UPGMA): $\frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$.

Centroid linkage clustering (UPGMC): $\|c_s - c_t\|$ where c_s, c_t are centroids of clusters s, t , respectively.

עם התקדמות התהליך יש פחות אשכולות, כאשר האשכולות כבר לא מכילים נקודה אחת בלבד אלא הם הולכים וגדלים. שיטה זו מכונה "bottom-up" כיוון שבהתחלה כל נקודה הינה אשכול עצמאי ובכל צעד של האלגוריתם מספר האשכולות קטן באחד. במילים אחרות, האלגוריתם בונה את האשכולות ממצב שבו אין למעשה חלוקה לאשכולות למצב שבו נוצרים אשכולות הולכים וגדלים.

divisive clustering – בשיטה זו מבצעים פעולה הפוכה – מסתכלים על כל הנקודות כאשכול אחד, ואז בכל שלב מבצעים חלוקה של אחד האשכולות לפי כלל חלוקה שנקבע מראש, עד שמגיעים ל- K אשכולות. כיוון שיש 2^n דרכים לחלק את המדגם, יש הכרח לנקוט בשיטות היוריסטיות כדי לקבוע את כלל החלוקה המתאים בכל שלב. שיטה מקובלת לביצוע החלוקה נקראת DIANA (DIvisive ANALysis Clustering), ולפיה בכל שלב בוחרים את האשכול בעל השונות הכי גדולה ומחלקים אותו לשניים. שיטה זו מכונה "top-down" כיוון שבהתחלה יש אשכול יחיד ובכל צעד של האלגוריתם מתווסף עוד אשכול.

את התצוגה של האלגוריתם ניתן להראות בצורה נוחה באמצעות dendrogram – דיאגרמה הבנויה כעץ המייצג קשרים בין קבוצות.



איור 2.11 תצוגה של Hierarchical Clustering בעזרת dendrogram (a) – divisive – התחלה מאשכול יחיד ופיצול עד שמגיעים למספר האשכולות הרצוי (במקרה זה $K = 4$). (b) – agglomerative – בהתחלה כל נקודה הינה אשכול, ובכל צעד מחברים שני אשכולות עד שמגיעים למספר האשכולות הרצוי.

2.2.5 Local Outlier Factor (LOF)

אלגוריתם Local Outlier Factor הינו אלגוריתם של למידה לא מונחית למציאת נקודות חריגות (Outliers). האלגוריתם מחשב לכל נקודה ערך הנקרא Local Outlier Factor (LOF), ועל פי ערך זה ניתן לקבוע עד כמה הנקודה היא חלק מקבוצה או לחילופין חריגה ויוצאת דופן.

בשלב ראשון בוחרים ערך k מסוים. עבור כל נקודה x_i , נסמן את k השכנים הקרובים ביותר שלה ב- $N_k(x_i)$. כעת נגדיר את k -distance של כל נקודה כמרחק שלה מהשכן הרחוק ביותר מבין השכנים ב- $N_k(x_i)$. אם למשל $k = 3$, אזי $N_k(x_i)$ הוא סט המכיל את שלושת השכנים הקרובים ביותר ל- x_i , וה- k -distance שלה הוא המרחק מהשכן השלישי הכי קרוב. חישוב המרחק בין שני שכנים נתון לבחירה – זה יכול להיות למשל מרחק אוקלידי, מרחק מנהטן ועוד. עבור בחירה של מרחק אוקלידי, ניתן להסתכל על k -distance כמעגל – הרדיוס של מעגל המינימלי המכיל את כל הנקודות השייכות ל- $N_k(x_i)$ הוא ה- k -distance.

לאחר חישוב ה- k -distance של כל נקודה, מחשבים לכל נקודה Local Reachability Density (LRD) באופן הבא:

$$LRD_k(x_i) = \frac{1}{\sum_{x_j \in N_k(x_i)} \frac{RD(x_i, x_j)}{k}}$$

כאשר $RD(x_i, x_j) = \max(k - \text{distance of } x_i, \text{distance}(x_i, x_j))$. הגודל LRD מחשב את ההופכי של ממוצע המרחקים בין x_i לבין k השכנים הקרובים אליו. ככל שנקודה יותר קרובה ל- k השכנים שלה כך ה-LRD שלה גדול יותר, ו-LRD קטן משמעותו שהנקודה יחסית רחוקה מאשכול הקרוב אליה.

בשלב האחרון בוחרים עבור כל נקודה x_i את היחס בין ה-LRD שלה וה-LRD של $N_k(x_i)$. היחס הזה הוא ה-LOF, והוא מחושב באופן הבא:

$$LOF_k(x_i) = \frac{\sum_{x_j \in N_k(x_i)} LRD(x_j)}{k} \times \frac{1}{LRD(x_i)}$$

הביטוי הראשון במכפלה הוא ממוצע ה-LRD של k השכנים של נקודה x_i , ולאחר חישוב הממוצע מחלקים אותו ב-LRD של הנקודה x_i עצמה. אם הערכים קרובים, אז ה-LOF יהיה שווה בקירוב ל-1, ואם הנקודה x_i באמת לא שייכת לאשכול של נקודות, אז ה-LRD שלה יהיה נמוך משמעותית מהממוצע של ה-LRD של השכנים שלה, וממילא ה-LOF שלה יהיה גבוה. אם עבור נקודה x_i מתקבל $LOF \approx 1$, אז סביר שהיא חלק מאשכול מסוים.

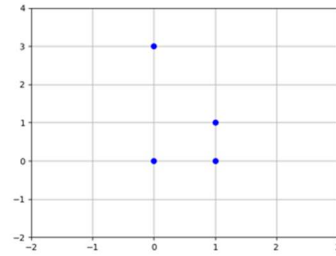
כדי להמחיש את התהליך נסתכל על האוסף הבא: $\{A = (0,0), B = (1,0), C = (1,1), D = (0,3)\}$, ונקבע $k = 2$. נחשב את ה- k -distance של כל נקודה במונחים של מרחק מנהטן:

$$k(A) = \text{distance}(A, C) = 2$$

$$k(B) = \text{distance}(B, A) = 1$$

$$k(C) = \text{distance}(C, A) = 2$$

$$k(D) = \text{distance}(D, C) = 3$$



נחשב את ה-LRD:

$$LRD_2(A) = \frac{1}{\frac{RD(A, B) + RD(A, C)}{k}} = \frac{2}{1 + 2} = 0.667$$

$$LRD_2(B) = \frac{1}{\frac{RD(B, A) + RD(B, C)}{k}} = \frac{2}{2 + 2} = 0.5$$

$$LRD_2(C) = \frac{1}{\frac{RD(C, B) + RD(C, A)}{k}} = \frac{2}{1 + 2} = 0.667$$

$$LRD_2(D) = \frac{1}{\frac{RD(D, A) + RD(D, C)}{k}} = \frac{2}{3 + 3} = 0.334$$

ולבסוף נחשב את ה-LOF:

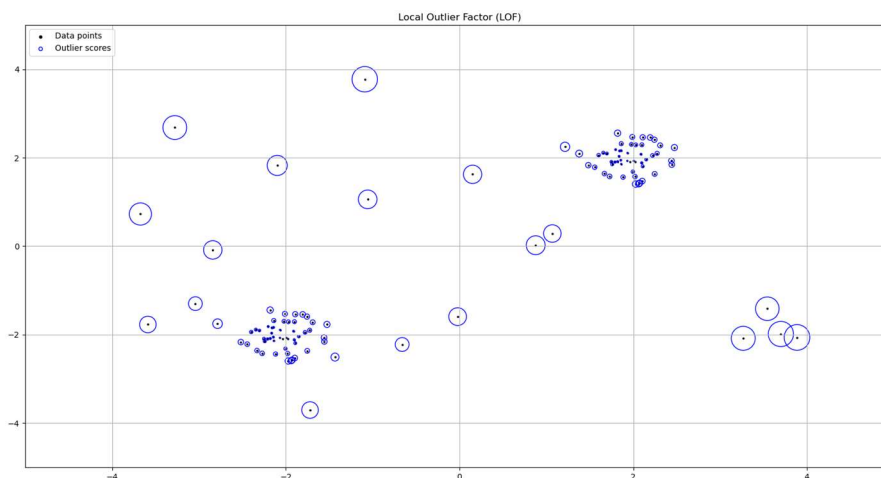
$$LOF_2(A) = \frac{LRD_2(B) + LRD_2(C)}{k} \times \frac{1}{LRD_2(A)} = 0.87$$

$$LOF_2(B) = \frac{LRD_2(A) + LRD_2(C)}{k} \times \frac{1}{LRD_2(B)} = 1.334$$

$$LOF_2(C) = \frac{LRD_2(B) + LRD_2(A)}{k} \times \frac{1}{LRD_2(C)} = 0.87$$

$$LOF_2(D) = \frac{LRD_2(A) + LRD_2(C)}{k} \times \frac{1}{LRD_2(D)} = 2$$

כיוון ש- $LOF_2(D) \gg 1$ באופן יחסי לשאר הנקודות, נסיק כי נקודה D היא outlier.



איור 2.12 Local Outlier Factor (LOF) – מציאת נקודות חריגות על ידי השוואת ערך ה-LRD של כל נקודה לממוצע ה-LRD של k השכנים שלה. ככל שה-LOF גדול יותר (העיגול הכחול), ככה הנקודה יותר רחוקה מאשכול של נקודות.

יש שני אתגרים מרכזיים בשימוש באלגוריתם זה – ראשית יש לבחור k מתאים, כאשר k יחסית קטן יהיה טוב עבור נקודות רועשות, אך יכול להיות בעייתי במקרים בהם יש הרבה מאוד נקודות הצמודות אחת לשנייה, ונקודה שמעט רחוקה מאוסף תזוזה כחריגה למרות שהיא באמת כן שייכת אליו. k גדול לעומת זאת יתגבר על בעיה זו, אך הוא לא יזהה נקודות חריגות שנמצאות בקירוב לאשכולות של נקודות. מלבד אתגר זה, יש צורך לתת פרשנות לתוצאות המתקבלות, ולהחליט על סף מסוים שך LOF, שהחל ממנו נקודה מסווגת כחריגה. LOF קטן מ-1 הוא בוודאי לא outlier אך עבור ערכי LOF גדולים מ-1 אין כלל חד משמעי עבור איזה ערך הנקודה היא outlier ועבור איזה ערך היא לא. כדי להתמודד עם אתגרים אלו הוצעו הרחבות לשיטה המקורית, כמו למשל שימוש בסטטיסטיקות שונות המורידות את התלות בבחירת הערך k (LoOP – Local Outlier Probability), או שיטות סטטיסטיות העוזרות לתת פרשנות לערכים המתקבלים (Interpreting and Unifying Outlier Scores).

2.3 Dimensionally Reduction

הורדת ממד (Dimensionality Reduction) הינה טרנספורמציה של דאטה מממד גבוה לממד נמוך, כאשר נרצה שהורדת הממד לא תשנה באופן מהותי את מאפייני הדאטה המקורי. הורדת הממד של דאטה נתון נדרשת משתי סיבות עיקריות; הראשונה טכנית וקשורה לסיבוכיות גבוהה במערכת מרובת ממדים, ואילו הסיבה השנייה יותר עקרונית ומהותית – הורדת הממד של הדאטה קשורה לניסיון להבין מהם המשתנים העיקריים ומהם המשתנים המשניים, הפחות חשובים להבנת הדאטה (אלו שפחות מאפיינים דוגמא נתונה ביחס לדוגמאות אחרות). לעיתים התחשבות במשתנים המשניים משפיעה לרעה על ביצועי המודל, למשל על ידי הוספת רעש ולא מידע. תופעה זו נקראת קללת הממדיות (curse of dimensionality). יתרון נוסף של הורדת ממד טמון בווזואליזציה של המידע, כך שניתן להציגו על ידי 2 או 3 ממדים עיקריים, בעזרת גרף דו-ממדי או תלת-ממדי בהתאמה.

דוגמא למערכת מרובת ממדים יכולה להיות מדידת רמות חלבונים (פרוטאינים) של גנים (genes) המבוטאים בתא חי, כאשר כל ממד, או מאפיין (פיצ'ר), מתאים לגן אחר. באופן כללי, ייתכן ונמדדים בכל ניסוי מאות תאים, כאשר לכל תא נמדדות רמות ביטוי של מאות או אלפי גנים. כמות עצומה זו של מידע בממד גבוה (אלפי תאים ואלפי גנים בכל תא) מאתגרת את המחקר – הן מבחינת זיהוי המאפיינים, או רמות הגנים המבוטאים, הרלוונטיים והמשפיעים ביותר, והן מבחינת ניסיון למדל את הדאטה בצורה כמה שיותר פשוטה. במחקר משנת 2007 נלקחו 105 דגימות של תאי סרטן שד, כאשר לכל דגימה (או דוגמה) נמדדו רמות התבטאות של 27,648 גנים שונים. כמובן שלנתח את המידע בצורה הגולמית זו משימה בלתי אפשרית, ויש הכרח לבצע עליו מניפולציה כלשהיא כדי שיהיה אפשר לעבוד איתו.

ישנן שיטות מרובות להורדת ממד לדאטה נתון, כאשר ניתן לסווג לשני חלקים עיקריים: בחירת מאפיינים (feature selection), והטלת מאפיינים (features projection). השיטה הראשונה היא ניסיון לבחור את המאפיינים (המשתנים) המתארים באופן מספק את המידע הנתון. השנייה, שבה עוסק פרק זה, נוקטת בגישה של הטלה, טרנספורמציה, של המאפיינים הקיימים לסט של מאפיינים חדשים. חשוב להדגיש שבשיטת בחירת המאפיינים אנו בעצם משמיטים מאפיינים פחות רלוונטיים. בניגוד לכך, בשיטה שנדון כעת, שיטת הטלת המאפיינים, כל מאפיין חדש הוא צירוף לינארי של כל האחרים, ולא רק של חלקם. כך, המאפיינים החדשים מכילים, או לוקחים בחשבון, כל אחד מהמאפיינים הנמדדים המקוריים, ללא השמטה.

ניתן לבצע הטלת מאפיינים באמצעות טרנספורמציות ליניאריות או לא-ליניאריות. בפרק זה נעסוק בטרנספורמציה ליניארית אחת, הנקראת ניתוח גורמים ראשיים (principal component analysis) ובשתי טרנספורמציות לא-ליניאריות (t-SNE, UMAP). נציין שקיימות עוד טרנספורמציות, ליניאריות ולא-ליניאריות, שאינן יוזכרו כאן.

2.3.1 Principal Components Analysis (PCA)

כפי שהוזכר לעיל, ניתוח גורמים ראשיים מבוסס על טרנספורמציה ליניארית של המאפיינים הקיימים. הגורם הראשי הראשון (first principal component, PCA_1) הינו הצירוף לינארי של המאפיינים הנתונים בעל השונות הגדולה ביותר. הגורם הראשי השני (second principle component, PCA_2) הוא גם צירוף לינארי של המאפיינים הנתונים, השונות שלו היא השנייה הגדולה ביותר, ובנוסף דורשים ש- PCA_2 יהיה אורתוגונלי ל- PCA_1 : $PCA_1 \perp PCA_2$. הגורם השלישי, הוא צירוף לינארי בעל השונות השלישית הגדולה ביותר, ומאונך לשני הגורמים הראשונים – $PCA_2 \perp PCA_3$ וגם $PCA_1 \perp PCA_3$, וכן הלאה כך שהגורם הראשי מסדר i , הוא בעל השונות ה- i -ית הגדולה ביותר תחת אילוץ של גורמים מאונכים – $PCA_i \perp PCA_j, \forall i < j$. הורדת הממד מתבצעת על ידי לקיחת מספר גורמים ראשיים ראשונים, והזנחת הקטנים ביותר.

לאחר שאפיינו את הגורמים הראשיים בהם אנו מעוניינים, עולה השאלה כיצד ניתן לבצע טרנספורמציה ליניארית שבעזרתה ניתן למצוא את הגורמים הראשיים האלו. נניח שבידינו דאטה $\hat{X} \in \mathbb{R}^{M \times N}$, כלומר נתונות M דוגמאות שונות, שכל אחת מהן היא בעלת N מאפיינים [למשל, עבור הדוגמא של תאי סרטן שד, נתון מידע מ- $M = 105$ תאים שונים, כאשר עבור כל תא נמדדו רמות ביטוי של $N = 27,648$ גנים שונים]. נסמן את מטריצת המאפיינים על

ידי $\hat{X} = \begin{bmatrix} \vec{X}_1 \\ \vdots \\ \vec{X}_M \end{bmatrix} = [\vec{X}^1, \dots, \vec{X}^N] \in \mathbb{R}^{M \times N}$, כאשר כל וקטור שורה, $\vec{X}_m, m \in \{1, \dots, M\}$, הינו נתוני המדידות של

המאפיינים השונים בדוגמא מספר m , ובהתאמה, וקטור עמודה $\vec{X}^n, n \in \{1, \dots, N\}$ (שימו לב לשינוי סימון, אינדקס עליון עבור וקטורי עמודה), הינו נתוני המדידות של מאפיין מסוים על כל הדוגמאות. נניח שממוצע המדידות עבור כל מאפיין הוא אפס, זאת אומרת שלכל מאפיין n -י מתקיים:

$$\text{mean}(\vec{X}^n) = \sum_{m=1}^M X_{m,n} = 0$$

מכיוון שכל עמודה של המטריצה מסמלת ערכים שלמאפיין מסוים במדידות שונות, סכום כל עמודה במטריצה \hat{X} הוא אפס. כעת, נרצה לבצע הטלה (טרנספורמציה) ליניארית, זאת אומרת נכפיל את מטריצה \hat{X} במטריצת משקלים \hat{W} :

$$\hat{T} = \hat{X} \cdot \hat{W}$$

אם נסמן את השורה ה- m -ית במטריצה \hat{T} על ידי \vec{T}_m , נקבל:

$$\vec{T}_m = \vec{X}_m \cdot \hat{W}$$

כאשר המטריצה $\hat{W} \in \mathbb{R}^{N \times K}$, כך ש- $\hat{T} \in \mathbb{R}^{M \times K}$. הטלה זו מביאה לכך שלאחר הטרנספורמציה נשארים רק K מאפיינים. כיוון שאנו מעוניינים בהורדת הממד, קרי הורדת מספר המאפיינים, נדרוש $K \leq N$.

את תהליך מציאת מטריצת המשקלים ניתן לנסח באופן פורמלי על ידי שלושה תנאים:

- (1) כל עמודה של מטריצת המשקלים הינה מנורמלת: $\|\hat{W}^k\|^2 = \sum_{m=1}^M (W_{m,k})^2 = 1$
- (2) השונות עבור המאפיין ה- k , המוגדרת על ידי $s_k^2 = (\vec{T}^k)^T \vec{T}^k = \sum_{m=1}^M (T_{mk})^2$, מקיימת: $s_k^2 > s_{k+1}^2$.
- (3) העמודות של \hat{W} אורתוגונליות זו לזו, זאת אומרת $\hat{W}^k \perp \hat{W}^{k'}$ לכל שתי עמודות k, k' .

נראה זאת באופן מפורש: נתחיל במציאת העמודה הראשונה \hat{W}^1 . נדרוש:

$$\hat{W}^1 = \underset{\|\hat{W}\|=1}{\operatorname{argmax}}(s_1^2)$$

זאת אומרת:

$$\begin{aligned} \hat{W}_1 = \underset{\|\hat{W}\|=1}{\operatorname{argmax}}(s_1^2) &= \underset{\|\hat{W}\|=1}{\operatorname{argmax}}((\vec{T}^1)^T \cdot \vec{T}^1) = \underset{\|\hat{W}\|=1}{\operatorname{argmax}}((\hat{X} \hat{W}^1)^T \cdot \hat{X} \hat{W}^1) \\ &= \underset{\|\hat{W}\|=1}{\operatorname{argmax}}((\hat{W}^1)^T (\hat{X})^T \cdot \hat{X} \hat{W}^1) \end{aligned}$$

ולכן העמודה הראשונה של מטריצת המשקלים \hat{W}^1 נתונה על ידי:

$$\hat{W}^1 = \underset{\|\hat{W}\|=1}{\operatorname{argmax}}((\hat{W}^1)^T \cdot \hat{S} \cdot \hat{W}^1)$$

כאשר מטריצה $\hat{S} \in \mathbb{R}^{(N \times N)}$ הינה מטריצת השונות המשותפת (covariance), המוגדרת על ידי $\hat{S} = (\hat{X})^T \cdot \hat{X}$. מטריצה זו, מסדר $N \times N$, מגדירה את השונות המשותפת בין שני מאפיינים, כאשר $S_{v_1, v_2} = \sum_{m=1}^M X_{v_1, m} X_{m, v_2}$. ניתן לשים לב כי מטריצה זו סימטרית וממשית (ולכן הרמיטית).

לפי משפט המינימום-מקסימום (קורנט-פישר-ויל): עבור \hat{S} מטריצה הרמיטית $(S_{ij} = S_{ji}^*)$, בעלת ערכים עצמיים $\lambda_1 \geq \dots \geq \lambda_K$, מתקיים:

$$\lambda_1 = \max_{\|\hat{W}\|=1} ((\hat{W}^1)^T \cdot \hat{S} \cdot \hat{W}^1)$$

כאשר \hat{W}^1 הינו הווקטור העצמי המתאים לערך העצמי המקסימלי של \hat{S} : λ_1 .

כעת, כדי למצוא את הווקטור העצמי הבא, \hat{W}^2 , והערך העצמי המתאים לו λ_2 , נגדיר מטריצה חדשה \tilde{X} :

$$\tilde{X} = \hat{X} - \hat{X}\hat{W}^1(\hat{W}^1)^T$$

$$\begin{aligned}\hat{W}^2 &= \operatorname{argmax}_{\|\hat{W}\|=1} (s_2^2) = \operatorname{argmax}_{\|\hat{W}\|=1} \left((\vec{T}^2)^T \cdot \vec{T}^2 \right) \\ &= \operatorname{argmax}_{\|\hat{W}\|=1} \left((\hat{W}^2)^T (\tilde{X} + \hat{X}\hat{W}^1(\hat{W}^1)^T)^T \cdot (\tilde{X} + \hat{X}\hat{W}^1(\hat{W}^1)^T) \hat{W}^2 \right) \\ &= \operatorname{argmax}_{\|\hat{W}\|=1} \left((\hat{W}^2)^T (\tilde{X})^T \cdot (\tilde{X}) \hat{W}^2 \right)\end{aligned}$$

כאשר \hat{W}^2 הינו הווקטור העצמי המתאים לערך העצמי המקסימלי של $(\tilde{X})^T$, ובעצם הוא הערך העצמי השני בגודלו עבור מטריצה $\hat{S} = \hat{X}^T \hat{X}$ (בחישוב השתמשנו בעובדה כי $(\hat{W}^1 \perp \hat{W}^2)$).

באופן כללי, כדי למצוא את \hat{W}^k והערך העצמי המתאים לו λ_k , נגדיר מטריצה חדשה \tilde{X} באופן הבא:

$$\tilde{X} = \hat{X} - \sum_{i=1}^{k-1} \hat{X}\hat{W}^i(\hat{W}^i)^T$$

$$\hat{W}^k = \operatorname{argmax}_{\|\hat{W}\|=1} \left((\hat{W}^k)^T (\tilde{X})^T \cdot (\tilde{X}) \hat{W}^k \right)$$

כך ש λ_k הינו הערך העצמי המקסימלי ה- k -י של מטריצת השונות המשותפת $\hat{S} = \hat{X}^T \hat{X}$.

ניתן גם, באופן פשוט יותר, להשתמש בשיטת פירוק לערכים סינגולריים, כאשר נמצא את הפירוק המתאים למטריצת השונות המשותפת:

$$\hat{S} = \hat{W} \cdot \hat{\Lambda} \cdot \hat{W}^T$$

כאשר $\hat{\Lambda}$ הינה מטריצה אלכסונית, ו- $\Lambda_{ii} = \lambda_i$ הינם הערכים העצמיים של \hat{S} המסודרים לפי גודלם מהגדול לקטן - $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_M$, ומטריצה \hat{W} מורכבת מווקטורי עמודה שהינם הווקטורים העצמיים המתאימים לערכים העצמיים. הווקטורים העצמיים בהגדרתם הינם אורתוגונליים זה לזה, וכיון ש- $\|\hat{W}^k\| = 1$ לכל k , הם בעצם אורתונורמליים.

לסיכום, על מנת למצוא את הגורמים הראשיים עבור המידע הנתון \hat{X} :

א. "מרכז" את הנתונים כך שהממוצע עבור כל מאפיין הוא אפס: $\hat{X}^m = \hat{X}^m - \operatorname{mean}_n(\hat{X}^m)$.

ב. מצא את מטריצת השונות המשותפת $\hat{S} = (\hat{X})^T \hat{X}$.

ג. מצא את $\hat{S} = \hat{W} \cdot \hat{\Lambda} \cdot \hat{W}^T$.

ד. חשב $\hat{T} = \hat{X} \cdot \hat{W}$.

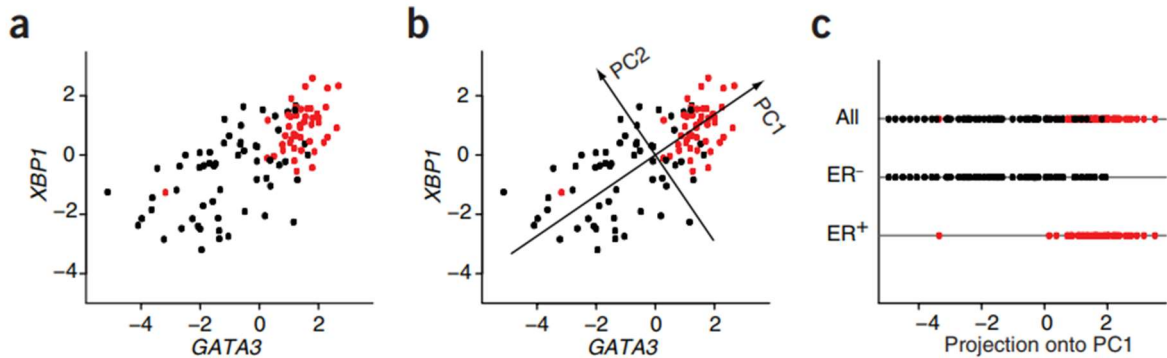
ה. הגורמים הראשיים נתונים על ידי וקטורי העמודה $\vec{W}^k \equiv PCA_k$, וההטלה של מדידה m למערכת

$$\vec{T}^m = \vec{X}^m \hat{W}$$

נציין שלשיטת הניתוח של גורמים ראשיים יש מספר מגבלות. ראשית, היא נותנת "משקל יתר" על מאפיינים שהשונות בהם גדולה, ללא קשר לחשיבותם, או ליחידות שבהן המאפיין נמדד (זאת אומרת לדוגמה שלגובה שנמדד בסנטימטרים יינתן "משקל" גבוה יותר מאשר גובה הנמדד במטרים). שנית, שיטת זו מניחה כי המדד החשוב הוא השונות המשותפת שהיא בעצם קורלציה לינארית בין שני משתנים, אולם ייתכן במערכות מסוימות שדווקא הקורלציה הלא-לינארית היא החשובה יותר. כמו כן, לעיתים "מרכז" המידע גורם לתוצאות לאבד ממשמעותן.

כדי להתגבר על המגבלות בשיטת ה-PCA שהצגנו לעיל פותחו שיטות נוספות או משלימות. לדוגמה, ניתן למזער את השפעת יחידות המידה על המאפיינים על ידי הפיכתם לחסרי יחידות. בנוסף, יש שיטות הלוקחות בחשבון קורלציות לא-לינאריות, לדוגמה שיטת kernel PCA, או שיטות להתמודדות עם בעיית המרכז על ידי דרישת משתנים חיוביים (NMF).

לצורך המחשה ניתן שתי דוגמאות. ראשית נחזור לדוגמה שהזכרנו בתחילת פרק זה – מחקר שפורסם בשנת 2007 ובו נלקחו 105 דגימות של תאי סרטן שד, כאשר לכל דגימה נמדדו רמות התבטאות של 27,648 גנים שונים. לשם הדגמה, נשתמש בניתוח שפורסם כשנה לאחר מכן (ב-2008) על ידי אחד מעורכי המחקר המקורי. שם, החוקר מציג רמות של שני חלבונים; האחד בשם GATA3, והשני בשם XBP1, כאשר הוא מסווג את דגימות תאי הסרטן לפי סוג קולטני האסטרון שלהם (+ או -). כעת, על ידי "סיבוב" מערכת הצירים – בעזרת טרנספורמציה ליניארית PCA כפי שהוסבר לעיל – נמצא כי ניתן לסווג, ללא איבוד מידע רב, את מצב קולטני האסטרון בתאי סרטן השד על ידי הגורם הראשי הראשון PCA₁, כפי שניתן לראות באיור. יש לשים לב שהגורם הראשי הראשון, PCA₁, מכיל מידע משני החלבונים.



איור 2.13 (a) רמות ביטוי של שני חלבונים GATA3 (ציר ה-X) ו-XBP1 (ציר ה-Y). קולטני אסטרון חיוביים או שלילים מסומנים באדום ושחור בהתאמה. (b) מציאת הגורמים הראשיים, וסיבוב מערכת הצירים. בהתאם לתיאוריה, ניתן להבחין כי השונות של המידע על גבי הציר החדש PCA₁ הינה מקסימלית. (c) הצגת תוצאות המדידה כפונקציה של PCA₁ בלבד. בגרף זה ניתן לראות בבירור כיצד הורדת הממד מסייעת למצוא הבחנה פשוטה (בממד אחד) בין קולטני האסטרון.

נביא בנוסף דוגמה חישובית מפורטת. נניח ונתון המערך הדו-ממדי הבא:

$$X = \begin{pmatrix} -0.5 & -0.4 \\ -0.4 & -0.1 \\ 0.1 & 0 \\ 0.3 & 0.3 \\ 0.5 & 0.2 \end{pmatrix}$$

מערך הנתונים מכיל 5 דוגמאות, ולכל דוגמה נמדדו שני מאפיינים. זאת אומרת $M = 5, N = 2$, כך ששורות המטריצה מציגות את המדידות השונות, והעמודות מייצגות את מאפיינים.

מערך זה כבר ממורכז, כלומר מתקיים עבור המאפיין הראשון:

$$\text{mean}_1(X^m) = \sum_{m=1}^5 X_{m1} = -0.5 - 0.4 + 0.1 + 0.3 + 0.5 = 0$$

ועבור המאפיין השני:

$$\text{mean}_2(X^m) = \sum_{m=1}^5 X_{m2} = -0.4 - 0.1 + 0 + 0.3 + 0.2 = 0$$

נחשב את מטריצת השונות המשותפת:

$$S = (\hat{X})^T \hat{X} = \begin{pmatrix} -0.5 & -0.4 & 0.1 & 0.3 & 0.5 \\ -0.4 & -0.1 & 0 & 0.3 & 0.2 \end{pmatrix} \begin{pmatrix} -0.5 & -0.4 \\ -0.4 & -0.1 \\ 0.1 & 0 \\ 0.3 & 0.3 \\ 0.5 & 0.2 \end{pmatrix}$$

$$= \begin{pmatrix} 0.5^2 + 0.4^2 + 0.1^2 + 0.3^2 + 0.5^2 & 0.5 \cdot 0.4 + 0.4 \cdot 0.1 + 0.1 \cdot 0 + 0.3^2 + 0.5 \cdot 0.2 \\ 0.5 \cdot 0.4 + 0.4 \cdot 0.1 + 0.1 \cdot 0 + 0.3^2 + 0.5 \cdot 0.2 & 0.4^2 + 0.1^2 + 0^2 + 0.3^2 + 0.2^2 \end{pmatrix}$$

$$= \begin{pmatrix} 0.76 & 0.43 \\ 0.43 & 0.3 \end{pmatrix}$$

על מנת ללכסן מטריצה זו, נפתור:

$$0 = |\hat{S} - \lambda I| = \begin{vmatrix} 0.76 - \lambda & 0.43 \\ 0.43 & 0.3 - \lambda \end{vmatrix} = (0.76 - \lambda)(0.3 - \lambda) - 0.43^2 \approx (\lambda - 1.02)(\lambda - 0.04)$$

כאשר לפולינום אופייני זה שתי פתרונות: $\lambda_1 \approx 1.02, \lambda_2 \approx 0.04$ (שים לב שבחרנו $\lambda_1 > \lambda_2$, התוצאות המובאות בחלק זה מקורבות ולכן הסימון \approx).

נמצא את הווקטור העצמי המתאים לערך העצמי הגדול מבין השניים λ_1 . וקטור זה, המסומן על ידי \hat{W}^1 , מקיים:

$$\hat{S}\hat{W}^1 = \lambda_1\hat{W}^1$$

כך ש:

$$0 = (\hat{S} - \lambda_1 I)\hat{W}^1 \approx \begin{pmatrix} -0.83 & 0.107 \\ 0.107 & -0.94 \end{pmatrix} \begin{pmatrix} W_{11} \\ W_{21} \end{pmatrix} \Rightarrow \hat{W}^1 \approx \begin{pmatrix} 0.86 \\ 0.51 \end{pmatrix}$$

הווקטור העצמי השני, \hat{W}^2 , המתאים לערך העצמי $\lambda_2 \approx 0.04$, מחושב באותו אופן, ומתקבל: $\hat{W}^2 \approx \begin{pmatrix} 0.51 \\ -0.86 \end{pmatrix}$.

כך שמטריצת המשקלים נתונה על ידי:

$$\hat{W} = (\hat{W}^1 \quad \hat{W}^2) = \begin{pmatrix} 0.86 & 0.51 \\ 0.51 & -0.86 \end{pmatrix}$$

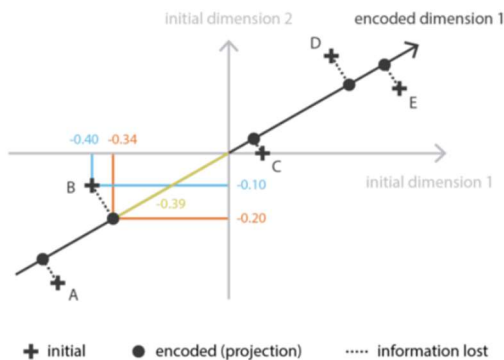
הטלת המדידות למערכת המאפיינים החדשה נתונה על ידי:

$$\hat{T} = \hat{X} \cdot \hat{W}$$

לכן, המדידות של הגורם הראשי הראשון, נתונות על ידי

$$\hat{T}^1 = \hat{X} \cdot \hat{W}^1 \approx \begin{pmatrix} -0.5 & -0.4 \\ -0.4 & -0.1 \\ 0.1 & 0 \\ 0.3 & 0.3 \\ 0.5 & 0.2 \end{pmatrix} \begin{pmatrix} 0.86 \\ 0.51 \end{pmatrix} = \begin{pmatrix} -0.5 \cdot 0.86 - 0.4 \cdot 0.51 \\ -0.4 \cdot 0.86 - 0.1 \cdot 0.51 \\ 0.1 \cdot 0.86 \\ 0.3 \cdot 0.86 + 0.51 \cdot 0.3 \\ 0.5 \cdot 0.86 + 0.2 \cdot 0.51 \end{pmatrix} \approx \begin{pmatrix} -0.63 \\ -0.39 \\ 0.09 \\ 0.41 \\ 0.53 \end{pmatrix}$$

נראה זאת באופן גרפי:



Point	Initial	Encoded
A	(-0.50, -0.40)	-0.63
B	(-0.40, -0.10)	-0.39
C	(0.10, 0.00)	0.09
D	(0.30, 0.30)	0.41
E	(0.50, 0.20)	0.53

איור 2.14 הורדת ממד של דאטה דו-ממדי לממד אחד.

נספח: משפט המינימום-מקסימום (קורנט-פישר-ויל):

עבור $\hat{S} \in \mathbb{R}^{M \times M}$ מטריצה הרמיטית ($S_{ij} = S_{ji}^*$) מסדר עם ערכים עצמיים $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_M$, מתקיים:

$$\begin{aligned} \lambda_m &= \min_U \left\{ \max_{\substack{\vec{x} \in U, \\ \|\vec{x}\|=1}} \{ \vec{x}^\dagger \hat{S} \vec{x} \} \mid \dim(U) = M - m + 1 \right\} \\ &= \min_U \left\{ \max_{\substack{\vec{x} \in U, \\ \vec{x}^\dagger \vec{x} = 1}} \left\{ \frac{\vec{x}^\dagger \hat{S} \vec{x}}{\vec{x}^\dagger \vec{x}} \right\} \mid \dim(U) = M - m + 1 \right\} \end{aligned}$$

הערך העצמי המקסימלי מקיים:

$$\lambda_1 = \max_{\|\hat{W}\|=1} ((\hat{W}^1)^T \cdot \hat{S} \cdot \hat{W}^1)$$

כאשר \hat{W}^1 , הינו הערך העצמי המתאים ל- λ_1 – ערך העצמי המקסימלי של \hat{S} .

2.3.2 t-distributed Stochastic Neighbors Embedding (t-SNE)

אלגוריתם הורדת הממד PCA פועל באופן לינארי, מה שמקל על תהליך החישוב שלו, אך מגביל את יכולות ההכללה שלו. אלגוריתם אחר, לא לינארי, נקראת t-SNE, והוא מנסה לקחת את הדאטה בממד גבוה ועל ידי מידע סטטיסטי למפות אותו למערכת דו-ממדית או תלת-ממדית. לשם כך, נשתמש באותו מערך נתונים, $\hat{X} \in \mathbb{R}^{M \times N}$, כאשר M הוא מספר הדוגמאות, ו- N הוא מספר המאפיינים (או המשתנים). חשוב לשים לב כי כל מדידה מיוצגת על ידי וקטור שורה \vec{X}_m . הרעיון הכללי של השיטה הוא למפות את סט המדידות באופן כזה שמדידות דומות יותר, קרי מדידות "קרובות" יותר במרחב ה- N ממדי, יוצגו על ידי נקודות קרובות יותר במרחב חדש K -ממדי, כאשר לרוב $K \leq 3$. נסמן את המרחב המקורי ה- X ואת המרחב החדש ב- \mathcal{Y} , כאשר בשני המרחבים המדידות מוצגות על ידי נקודות בגרף פיזור (scatter plot). המטריקה המשמשת למדידת דמיון (similarity) בין שתי נקודות במרחב המקורי X הינה הסתברותית. עבור שתי מדידות m_1, m_2 במרחב המקורי ה- N -ממדי, ההתפלגות הנורמלית המשותפת P_{m_1, m_2} הינה:

$$P_{m_1, m_2} = \frac{Z_1^{-1}}{2N} \exp\left(-\frac{\|\vec{X}_{m_1} - \vec{X}_{m_2}\|^2}{2\sigma_1^2}\right) + \frac{Z_2^{-1}}{2N} \exp\left(-\frac{\|\vec{X}_{m_1} - \vec{X}_{m_2}\|^2}{2\sigma_2^2}\right)$$

כאשר σ_i נקרא פרפלקסיות (perplexity) והוא פרמטר שנקבע מראש, ו- Z הינו קבוע הנורמליזציה, המוגדר על ידי $Z_i = \sum_{k \neq i} \exp\left(-\frac{\|\vec{X}_i - \vec{X}_k\|^2}{2\sigma_i^2}\right)$. עבור נקודות קרובות יותר, עבורן הביטוי $\|\vec{X}_{m_1} - \vec{X}_{m_2}\|$ קטן, ההסתברות שהנקודה \vec{X}_{m_1} שכנה של \vec{X}_{m_2} גדולה. לעומת זאת כאשר הנקודות רחוקות זו מזו, כלומר $\|\vec{X}_{m_1} - \vec{X}_{m_2}\|$ גדול, ההסתברות שהנקודה \vec{X}_{m_1} שכנה של \vec{X}_{m_2} קטנה מאוד עד אפסית.

כעת, כפי שהוזכר לעיל, נרצה למפות את סט המדידות: $\begin{bmatrix} \vec{X}_1 \\ \vdots \\ \vec{X}_M \end{bmatrix} \rightarrow \begin{bmatrix} \vec{Y}_1 \\ \vdots \\ \vec{Y}_M \end{bmatrix}$, כך שהממד של \vec{Y}_m הינו נמוך (2 או 3 ממדים). בנוסף, נדרוש שנקודות דומות ("שכנות") במרחב X , ישארו שכנות לאחר המיפוי למרחב \mathcal{Y} . מתברר שפונקציית ההסתברות המותנית, המתאימה לתיאור דמיון בין נקודות שכנות במרחב החדש \mathcal{Y} , הינה התפלגות t, הנקראת גם התפלגות סטודנט עם דרגת חופש אחת (נדון ברעיון לבחור בפונקציית ההסתברות אלו בהמשך). כך, נכמת את הדמיון בין m_1 לבין m_2 , על ידי ההסתברות המשותפת Q_{m_1, m_2} המוגדרת באופן הבא:

$$Q_{m_1, m_2} = 3^{-1} \frac{1}{1 + \|\vec{Y}_{m_1} - \vec{Y}_{m_2}\|^2}$$

כאשר $3 = \sum_{k \neq j} \left(1 + \|\vec{Y}_k - \vec{Y}_j\|^2\right)^{-1}$ הינו קבוע נורמליזציה.

המיפוי בין מרחב המקורי X לבין המרחב החדש \mathcal{Y} הוא מיטבי אם הוא "משמר" את השכנות של נקודות (מדידות) קרובות. לשם כך נגדיר את פונקציית המחיר על ידי Kullback-Leibler divergence, הבוחן מרחק בין שתי התפלגויות:

$$C = \mathcal{D}_{KL}(P|Q) \equiv \sum_{m_1} \sum_{m_2} P_{m_1, m_2} \log\left(\frac{P_{m_1, m_2}}{Q_{m_1, m_2}}\right)$$

נרצה למצוא את הווקטור $\begin{bmatrix} \vec{Y}_1 \\ \vdots \\ \vec{Y}_M \end{bmatrix}$ עבורו פונקציית המחיר מינימלית, ולשם כך נשתמש בגרדיאנט לפי \vec{Y}_{m_i} :

$$\frac{\delta C}{\delta \vec{Y}_{m_i}} = \frac{\delta}{\delta \vec{Y}_{m_i}} \left[\sum_{m_1} P_{m_1, m_i} \log \left(\frac{P_{m_1, m_i}}{Q_{m_1, m_i}} \right) + \sum_{m_2} P_{m_1, m_2} \log \left(\frac{P_{m_i, m_2}}{Q_{m_i, m_2}} \right) \right]$$

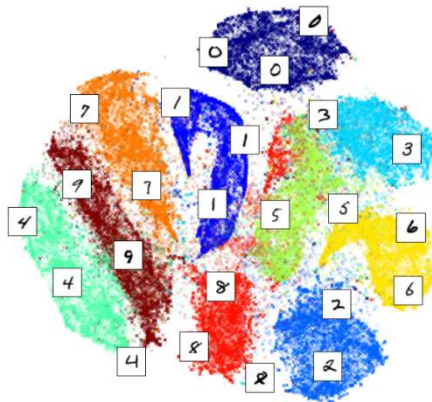
$$= 4 \sum_{m_1} (P_{m_1, m_i} - Q_{m_1, m_i}) \left(1 + \|\vec{Y}_{m_1} - \vec{Y}_{m_i}\|^2 \right)^{-1} (\vec{Y}_{m_1} - \vec{Y}_{m_i})$$

חישוב המינימום באופן אנליטי לא תמיד אפשרי או לא תמיד יעיל, ולכן מקובל להשתמש בשיטת gradient descent, שהינה שיטה איטרטיבית למציאת המינימום של פונקציה (פירוט על שיטה זו ווריאציות שונות שלה מופיע בחלק 4.3.5). עבור הורדת הממד, חישוב המינימום בעזרת שיטה זו יעשה באופן הבא:

- א. אתחול: * נתון $X \in \mathbb{R}^{M \times N}$.
* פרמטר לפונקציית הדמיון: בחירת השונות σ^2 .
* בחירת פרמטרים לאופטימיזציה: קצב הלמידה η , מומנטום $\alpha(t)$.
- ב. חשב את P_{m_1, m_2} .
- ג. אתחל את המיפוי $\mathcal{Y}^{(0)} = \{\vec{Y}_1, \vec{Y}_2, \dots, \vec{Y}_M\} \sim N(0, s\hat{I}_M)$ [ז"א בחר את הערכים ההתחלתיים לפי התפלגות גאוסיאנית עם ממוצע 0 וסטיית תקן s (s נבחר להיות קטן, נניח $s = 10^{-4}$). \hat{I}_M מטריצת יחידה].
- ד. עבור איטרציה t :
* חשב את Q_{m_1, m_2} .
* חשב את הגרדיאנט של פונקציית המחיר $\frac{\delta C}{\delta \mathcal{Y}}$.
* עדכן: $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)[\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}]$

נעיר כי בשפות תכנות רבות, האלגוריתם עצמו כבר מוגדר על ידי פונקציות מובנות, ויש רק להגדיר את הפרמטרים הדרושים.

במאמר המקורי שהציג את השיטה הובאה דוגמה של שימוש באלגוריתם עבור הטלה של הספרות 0 עד 9, המיוצגות על ידי תמונות בממד גבוה $\mathbb{R}^{28 \times 28}$, למרחב דו ממדי. בדוגמה זו נלקחו 6,000 תמונות של ספרות ומיפו אותן למרחב דו-ממדי. במרחב זה ניתן לראות בבירור כיצד כל תמונה מופתה לאזור אחר, כיוון שבפועל נוצרו עשרה אשכולות שונים, המובחנים בצורה ברורה אחד מהשני. בייצוג הדו-ממדי אין משמעות לצירים, כיוון שבאלגוריתם זה יש חשיבות רק למרחק היחסי בין הנקודות.



איור 2.15 הצגה (וויזואליזציה) דו-ממדית של מערך נתונים עבור כתב-יד של ספרות (MNIST) על ידי שיטת t-SNE. כל דוגמא \vec{X}_m מאופיינת על ידי $28 \times 28 = 784$ ערכים (פיקסלים בגוויי אפור) ומסווגת להיות ספרה בין 0 ל-9. באיור מוצגות 6000 נקודות (מדידות) כאלו, כאשר צבעים שונים מייצגים ספרות שונות. מלבד ההבחנה בין הספרות, ניתן לראות שספרות דומות קרובות זו לזו גם במרחב החדש (למשל הספרה 1 קרובה לספרה 7, שבתורה קרובה לספרה 9).

כאמור, פונקציית הדמיון בין שתי נקודות במרחב המקורי הינה הפילוג הנורמלי המשותף של שתי הנקודות, ואילו במרחב החדש פונקציית הדמיון הינה התפלגות t . שתי הערות חשובות על בחירות אלו:

א. סימטריה:

פונקציית הדמיון הגאומטרית בין שתי נקודות במרחב \mathcal{X} הינה פונקציה סימטרית, כלומר $P_{m_1, m_2} = P_{m_2, m_1}$. אולם ניתן להגדיר גם פונקציית דמיון א-סימטרית, המבוססת על התפלגות מותנת (במקום התפלגות משותפת). הפונקציה המותנת נתונה על ידי:

$$P_{m_1|m_2} = Z_2^{-1} \exp\left(-\frac{\|\vec{X}_{m_1} - \vec{X}_{m_2}\|^2}{2\sigma_2^2}\right)$$

כך ש:

$$P_{m_1, m_2} = \frac{P_{m_1|m_2} + P_{m_2|m_1}}{2N}$$

ב. בחירת פונקציית הדמיון במקום פונקציית t :

באלגוריתם שתואר, פונקציית הדמיון בין שתי נקודות במרחב ה- \mathcal{Y} נתונה על ידי התפלגות t . ניתן להגדיר גם פונקציה אחרת, למשל את פונקציית דמיון גאומטרית עבור שתי מדידות במרחב \mathcal{Y} . שיטה זו נקראת SNE, והגרדיאנט של פונקציית המחיר במקרה זה נתונה על ידי:

$$\frac{\delta \mathcal{C}}{\delta \vec{Y}_{m_i}} = 4 \sum_{m_1} (P_{m_1, m_i} - Q_{m_1, m_i}) (\vec{Y}_{m_1} - \vec{Y}_{m_i})$$

ג. אולם, פונקציית דמיון גאומטרית במרחב \mathcal{Y} יכולה לגרום לכך שנקודות לא מאוד קרובות במרחב \mathcal{X} , ימופו לנקודות קרובות במרחב \mathcal{Y} , כיוון שהגאומטריה בעצם גורם לאטרקטור (משיכה) יחסית חזק בין שתי נקודות, גם במקרים בהם הנקודות אינן מאוד קרובות. לעומת זאת, כאשר פונקציית הדמיון הינה התפלגות סטודנט t , שהינה התפלגות עם זנב כבד יותר, שתי נקודות שאינן מאוד קרובות ימופו בצורה ראויה למרחב \mathcal{Y} כך שאינן "נמשכות" זו לזו. שיטה אחרת, הנקראת UNI-SNE, מציעה להשתמש בהתפלגות אחידה, אך גם לה חסרון דומה ל-SNE, כאשר שתי נקודות לא מאוד דומות זו לזו, אינן "דוחות" אחת את השנייה. באופן אינטואיטיבי, ניתן לחשוב על גרדיאנט פונקציית המחיר כשדה כוח, ועל פונקציית המחיר בתור פוטנציאל, כך שהכוח הפועל הוא בעצם כוח קפיץ.

לשיטת t-SNE יש שלוש מגבלות עיקריות

- הורדת ממד: השיטה משמשת לוויזואליזציה של מידע מממד גבוה בדו-ממד או תלת-ממד. אולם, באופן עקרוני, ייתכן ונרצה להוריד את הממד לא לשם הצגתו, אלא לצרכים אחרים, כאשר הממד החדש הינו גדול מ-3. ייתכן ובממד גבוה פונקציית התפלגות סטודנט t עם דרגת חופש אחת, אשר לה משקל גבוה יחסית במרחקים גבוהים, לא תשמר את המבנה של המידע המקורי. לכן, כאשר נרצה להוריד לממד גבוה מ-3, פונקציית התפלגות t עם יותר מדרגת חופש אחת מתאימות יותר.
- קללת הממדיות: t-SNE מבוססת על מאפיינים מקומיים בין נקודות. השיטה, המבוססת על מטריקת מרחק אוקלידית, וכך מניחה לינאריות מקומית על גבי היריעה המתמטית בה מתקיימות הנקודות. אולם, במערכת נתונים בו הממד הפנימי גבוה, שיטת t-SNE עלולה להיכשל כיוון שהנחת הלינאריות לא מתקיימת. למרות שישנן מספר שיטות למזער תופעה זו, עדיין, בהגדרה, כאשר הממד הפנימי גבוה, לא ניתן להוריד ממד כך שמבנה המידע ישמר באופן מלא.
- פונקציית מחיר לא קמורה: הרבה שיטות למידה מבוססות על פונקציית הפסד קמורה, כך שתיאורטית מציאת אופטימיזציה (יחידה) לפונקציה זו אפשרית תמיד. אולם, בשיטת t-SNE, פונקציית המחיר אינה קמורה, והפתרון המתקבל על ידי האופטימיזציה משתנה בהתאם לפרמטרים הנבחרים.

2.4 Ensemble Learning

2.4.1 Introduction to Ensemble Learning

נניח ויש בידינו אוסף נתונים מסוים, ורוצים לבנות מודל המנתח את הנתונים האלו שמתבסס על אלגוריתם מסוים. כמעט תמיד, המודל לא יהיה מדויק במאה אחוז, והוא יהיה בעל שונות או בעל הטיה. ניתן להשתמש במכלול (Ensemble) של מודלים שונים המבוססים על אותו אלגוריתם רצוי, ובכך לקבל מודל משוקלל בעל שונות/הטיה נמוכים יותר מאשר מודל שמתבסס על אותו אלגוריתם אך נבנה באופן פשוט.

בכדי להבין את יותר טוב את החשיבות של שימוש ב-ensembles, יש להרחיב על ה-Trade off בין שונות המודל להטיה שבו. מודל אופטימלי יתאפיין בשונות נמוכה ובהטיה נמוכה. כלומר, השוני בין התחזיות לא יהיה מהותי, ובממוצע התחזית תהיה קרובה מאוד לערך האמיתי. מודל כזה יהיה מודל אמין, ונוכל לבסס עליו את צעדנו. למרבה הצער, מודל שכזה לרוב אינו אפשרי. סוג אחר של מודל יהיה המודל הגרוע, ההפוך למודל האופטימלי. זהו מודל עם שונות גבוהה והטיה גדולה. מודל שכזה יציג טווח רחב של תחזיות על אותם נתונים, ובממוצע יהיה רחוק מאוד מהערך האמיתי. מודל זה כלל אינו שימושי.

בפועל, המודלים במציאות ינועו לאורך שני קצוות: מודלים עם שונות גבוהה והטיה נמוכה, ומודלים עם שונות נמוכה והטיה גבוהה. הזיהוי של המיקום שלנו לאורך ציר זה קריטי, כיוון שהוא מאפשר לנו לבחור את דרך ההתמודדות הטובה ביותר. יש מספר משפחות של model ensembles, ושני העיקריים שבהם נקראים Bagging and Boosting. כאשר ניתקל במודלים עם שונות גבוהה, כלומר מודל הסובל מ-Overfitting, לרוב נרצה להשתמש באנסמבל מסוג Bagging על-מנת להוריד את השונות במודל הסופי. אלגוריתם מסוג Boosting יטפל במקרה השני, בו ההטיה גבוהה והשונות נמוכה.

2.4.2 Bootstrap aggregating (Bagging)

Bagging היא משפחת אלגוריתמים אשר פועלת כ-ensemble, כלומר – מספר אלגוריתמים שפועלים ביחד, על-מנת להגיע לתוצאה משופרת. כאמור, אלגוריתמים מסוג bagging נועדו להגדיל את יציבות המודל והעלאת הדיוק שלו, זאת תוך הורדת השונות והימנעות מ-overfitting. Bagging מורכב ממספר רב של אלגוריתמים, המכונים "לומדים חלשים" (Weak learners), כאשר כל אחד מהם מבצע למידה ותחזית על חלק מן הנתונים, מתוך מטרה להגיע לתוצאה איכותית. Bagging נחשבת שיטה נפוצה ופשוטה יחסית לשיפור ביצועים, אם כי היא עשויה להיות יקרה מבחינה חישובית.

מודל "פשוט" יקבל את הנתונים, יתאמן עליהם ויבצע תחזית על נתונים חדשים. זהו תהליך הלמידה והמבחן אשר ידוע לנו ממודלים כגון עץ החלטה (Decision Tree), רגרסיה לינארית וכו'. כפי שכבר הוסבר, מצב כזה עשוי להוביל להתאמת יתר של המודל לנתוני האימון, דבר שעשוי להוביל למודל בעל שונות גבוהה. בכדי להתמודד עם בעיה זו, אלגוריתמים מסוג Bagging מפרקים את התהליך לשני שלבים: Bootstrapping and Aggregating.

בשלב ה-Bootstrapping, יוצרים מהנתונים המקוריים קבוצות חדשות, כאשר כל קבוצה נוצרת על ידי דגימה (עם חזרות) של איברים מהקבוצה המקורית, באופן כזה שגודל כל קבוצה חדשה הוא בגודל של הדאטה המקורי. בשלב השני, ה-Aggregating, הקבוצות החדשות נכנסות לקלט "לומדים חלשים", אלגוריתמים פשוטים יותר, אשר עובדים במקביל על תחזית, כלומר, יוצרים מודל נפרד לכל קבוצה של נתונים. בשלב הסופי, יתבצע איחוד של כל המודלים על מנת ליצור מודל משוקלל בעל שונות קטנה יותר מאשר מודל המסתמך על הדאטה המקורי כפי שהוא.

אופן חיבור המודלים המתבצע ב-Bagging מבוסס על אותו רעיון של K-NN, אלגוריתם השכן הקרוב. כאמור לעיל (בפרק 2.1.3), באלגוריתם השכן הקרוב כל "שכן" העיד על התווית שלו, ולאחר הכרעת הרוב נקבעה התווית של התצפית החדשה. במקרה שבו נספור את תדירות כל התוויות השכנות, התווית הנבחרת תהיה של התצפית הנפוצה ביותר; נעשה זאת כאשר K-NN יעבוד כמסווג. במקרים בהם K-NN יעבוד כרגרסיה, יתבצע ממוצע של כל התוויות השכנות, וזאת גם תהיה התחזית. למעשה, אלגוריתמים מסוג Bagging מסוגלים לעבוד הן כרגרסיה והן כמסווג, וכאמור, הם ישתמשו באותו רעיון כמו ב-K-NN. כאשר bagging יעבוד כמסווג, כל weak learner יבצע תחזית, והתווית השכיחה ביותר תהיה התוצאה של האנסמבל. כאשר bagging יעבוד כרגרסיה, כל מודל יבצע תחזית, אבל התוצאה של האנסמבל תהיה הממוצע של כל המודלים.

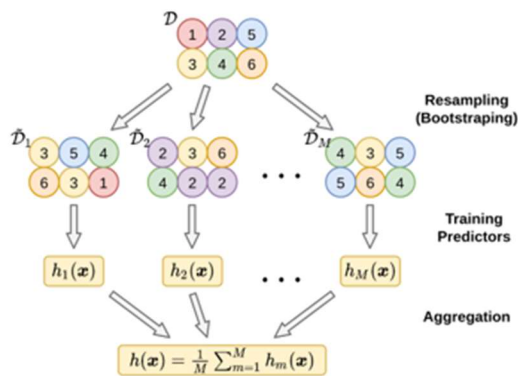
באופן פורמלי, עבור דאטה $X \in \mathbb{R}^{n \times d}$, ניצור M קבוצות חדשות באותו גודל של הדאטה המקורי – $\{X_m \in \mathbb{R}^{n \times d}\}_{m=1}^M$, ועבור כל קבוצה X_m נבנה מודל $c_m(x)$. עבור בעיות סיווג ההחלטה תתקבל על פי הצבעת הרוב:

$$C(x) = \text{majority}(\{c_1(x), \dots, c_M(x)\})$$

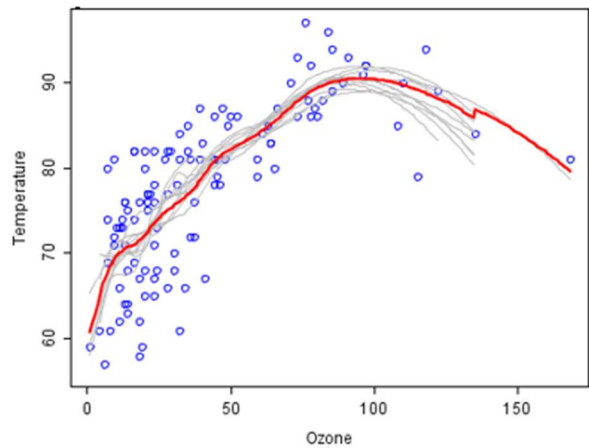
ועבור בעיות רגרסיה ההחלטה תתבצע בעזרת מיצוץ כל המודלים:

$$C(x) = \frac{1}{M} \sum_{m=1}^M c_m(x)$$

a



b



איור 2.16 a) אלגוריתם Bagging: בשלב ראשון יוצרים הרבה מחלקות שונות שנוצרות מהדאטה המקורי (Bootstrapping), לאחר מכן בונים מודל המתאים לכל מחלקה (Aggregating), ולבסוף יוצרים מודל יחיד המבוסס על כל המודלים הקודמים. b) דוגמה לבניית מודל רגרסיה בעזרת אלגוריתם Bagging. ניתן לראות שהמודל המשוקלל הוא בעל שונות קטנה יותר מכל שאר המודלים.

בין אם משתמשים בהכרעת הרוב ובין אם משתמשים במיצוע של המודלים, המודל המשוקלל שנוצר הופך להיות חלק יותר ובעל פחות שיפועים חדים, מה שמקטין את ה-overfitting, וממילא מפחית את השונות. ניתן להבין זאת על ידי דוגמה פשוטה – נניח ויש התפלגות נורמלית $\mathcal{N}(\mu, \sigma^2)$, אז השונות של n דגימות בלתי תלויות הינה $\frac{\sigma^2}{n}$. כעת נניח ומבצעים m ניסויים שבכל אחד מהם דוגמים n נקודות, ובין כל שתי קבוצות יש קורלציה ρ . השונות הממוצעת של הניסויים הינה:

$$\text{Var}\left(\frac{1}{m} \sum_{i=1}^m \text{single cycle}\right) = \frac{1}{m} (1 - \rho) \sigma^2 + \rho \sigma^2$$

אם נבצע הרבה מאוד ניסויים, כלומר ניקח m גדול מאוד, נקבל:

$$\lim_{m \rightarrow \infty} \frac{1}{m} (1 - \rho) \sigma^2 + \rho \sigma^2 = \rho \sigma^2$$

ובסך הכל השונות הסופית הינה בקירוב $\rho \sigma^2$, וביטוי זה לרוב קטן מאשר השונות של מודל המבוסס על הדאטה המקורי ללא שימוש ב-ensembles. ניתן לשים לב שככל שהקורלציה בין הקבוצות קטנה, כך השונות של המודל המשוקלל גם כן קטנה יותר.

מודל נפוץ מאוד מסוג Bagging הוא Random Forest. אלגוריתם זה משלב בין עצי החלטה לבין הרעיון הבסיסי של Bagging, כאשר הוא מפצל את הנתונים ואת המשתנים לעצי החלטה רבים, וכל אחד מהם מקבל חלק מסוים מן השלם. העצים הם בעלי שונות גבוהה, כלומר – כל אחד מהם הוא overfitting בפני עצמו, אך עם זאת הקורלציה ביניהם נמוכה, מה שמקל על הורדת השונות והימנעות מ-overfitting. לבסוף, השקלול של כל המודלים ביחד מצליח לייצר מודל בעל שונות נמוכה, ומוביל לתוצאות טובות.

ל-bagging יתרונות רבים. הוא מוריד את השונות, והוא גם חסין לערכים קיצוניים (Outliers). יכולת העבודה שלו במקביל עשויה לאפשר לו להגיע לתוצאות באופן מהיר יותר.

עם זאת, ל-bagging יש גם חסרונות. הוא אינו מוריד את ההטיה, ולכן עשוי לא להתאים במקרים רבים. במודלים של בינה מלאכותית יש חשיבות רבה ליכולת הפרשנות של המודל; לרוב, יידרש הסבר פחות טכני של תוצאות המודל למקבלי ההחלטות או לצרכנים. הם עשויים לא לקבל כלל החלטות של מודל שיראה כ"קופסא שחורה". יש קושי רב לתת פרשנות להחלטות של מודלים מבוססי bagging, והדבר מקשה על השימוש בו. מעבר לכך, bagging עשוי להיות יקר מבחינה חישובית. עקב כך, הוא שימושי מאוד במקרים בהן שיפור זעיר עשוי להוביל להצלחה, אך לרוב תינתן עדיפות למודלים פשוטים יותר של ensembles.

2.4.3 Boosting

כאמור, המושג boosting מתייחס למשפחת אלגוריתמים המשתמשים באוסף של מודלים "חלשים" על מנת ליצור מודל אחד "חזק", כאשר מודלים אלו מתמקדים בניסיון להפחית את ההטיה שיש למודל. מבחינה אינטואיטיבית,

מודל חלש הוא כזה שתוצאותיו מעט טובות יותר מניחוש אקראי בעוד שאחד חזק מתקרב לביצועים אופטימליים. בניגוד לטכניקות ensemble אחרות שפועלות במקביל, העקרון המנחה כאן הוא לשרשר את המודלים באופן כזה שכל מודל שמתווסף יטפל בשגיאות שקודמיו פספסו. היופי נעוץ בכך ש-boosting מוכיח כי למידה חלשה בהכרח מצביעה על קיום של שיטת למידה חזקה, לרוב, מודלים מבוססי boosting מתמקדים בבעיות סיווג בינארי.

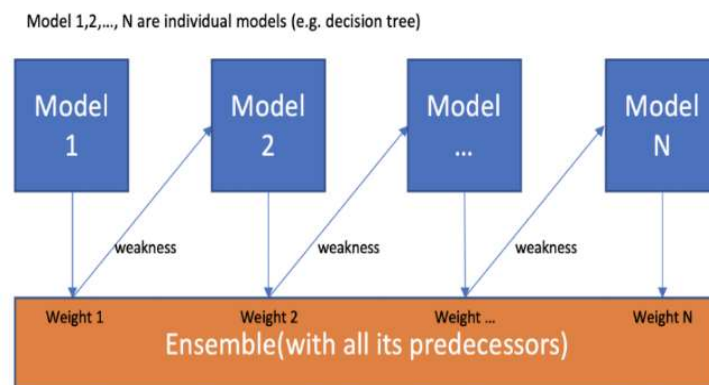
באופן פורמלי, המושגים "לומד חלש" ו-"לומד חזק" עבור בעיית סיווג בינארי מוגדרים כך: אלגוריתם נקרא לומד חזק אם לכל $\epsilon, \delta > 0$, האלגוריתם מסוגל (עבור אוסף נתונים גדול מספיק) לבנות מסווג $c(x)$ שמקיים $p(c(x) \neq y) < \epsilon$ בהסתברות גדולה מ- $1 - \delta$. לומד חלש הינו אלגוריתם שלכל $\delta > 0$ קיים $\gamma > 0$ כך שעבור אוסף נתוני מספיק גדול, האלגוריתם מסוגל לבנות מסווג שמקיים $p(c(x) \neq y) < \frac{1}{2} - \gamma$ בהסתברות גדולה מ- $1 - \delta$. כאמור, המטרה של boosting הינה לקחת אוסף של מודלים חלשים ובעזרתם ליצור מודל חזק, כאשר הצליחו להוכיח שניתן להפוך כל לומד חלש ללומד חזק על ידי בניית קומביניציה לינארית של מסווגים אשר נוצרו בעזרת הלומד החלש.

נמחיש את הרעיון של boosting בעזרת דוגמה: נניח ויש בידינו אוסף נתונים X , המחולק באופן אקראי לשלוש קבוצות שוות (כל אחת מכילה שליש מהנתונים) x_1, x_2, x_3 . כעת בונים מודל לצורך סיווג בינארי, המסומן ב- h_1 . נמצא כי h_1 מתאים בצורה טובה רק לקבוצות x_1, x_2 אך מסווג בצורה לא טובה את פריטי הקבוצה x_3 . כיוון ש- x_3 מכיל שליש מסך הנתונים, שגיאת הסיווג גדולה ו- $c_1(x)$ הוא מודל חלש, ונרצה לשפר אותו. בכדי לעשות זאת ניקח רק חלק מהנתונים, $X' \in X$, ונדאג לכך ש- X' יכיל הרבה מאיברי x_3 . כעת נבנה מודל נוסף $c_2(x)$ על בסיס X' , מתוך כוונה שמודל זה יתמקד גם בקבוצה x_3 ויסווג את איבריה בצורה טובה. כעת נניח שמודל זה אכן מסווג בצורה נאותה את איברי x_3 , אך הפעם המודל שוגה בצורה גסה בסיווג איברי x_2 . עקב השגיאה בסיווג x_2 המודל השני גם הוא מודל חלש, אך כעת יש בידינו שני מודלים חלשים שהחולשה בכל אחד נובעת מקבוצת איברים אחד של אוסף הנתונים המקורי X . אם נמצא דרך הולמת לחבר את שני המסווגים, נוכל ליצור מודל בעל פוטנציאל להצליח לסווג את X כמו שצריך.

באופן כללי, אם רוצים לאמן מודל $C(x)$ בעזרת אלגוריתם L על אוסף הנתונים \mathcal{D} , יש לבצע את השלבים הבאים:

1. אתחול הנתונים: $\mathcal{D}_1 = \mathcal{D}$.
2. עבור $t = 1, \dots, T$:
 - אימון מודל חלש על \mathcal{D}_t : $c_t(x) = L(\mathcal{D}_t)$.
 - חישוב שגיאת המסווג: $\epsilon_t = P_{x \sim \mathcal{D}_t}(c_t(x) \neq f(x))$.
 - התאמת הנתונים עבור האיטרציה הבאה: $\mathcal{D}_{t+1} = \text{Adjust Distribution}(\mathcal{D}_t, \epsilon_t)$.
3. איחוד המודלים החלשים: $C(x) = \text{combine outputs}\{c_1(x), \dots, c_T(x)\}$.

יש כל מיני שיטות כיצד לבצע את השלבים השונים באלגוריתם boosting, ונפרט את השיטות המרכזיות.



איור 2.17 – סכמה כללית של boosting. המודלים (במקרה זה מדובר בעץ החלטה רדוד, אך זה תקף לכל מודל חלש) מחוברים אחד לשני באופן שכל אחד לומד מהתפלגות המשוקללת בהתאם לשגיאות של המודלים הקודמים.

Adaptive-Boosting (AdaBoost)

Adaboost היא אחת הטכניקות הראשונות של boosting, ועל אף שקיימות טכניקות boosting נוספות, Adaboost היא בין הפופולריות ביותר בתחום (אם כי יש לה מספר לא מבוטל של וריאנטים). העוצמה הגלומה בטכניקה זו נובעת מכך שגם בהינתן מספר פיצ'רים רב, האלגוריתם מצליח להיפגע פחות מ"קללת הממדיות" ולשמור על יכולות ניבוי טובות, בניגוד לאלגוריתמים אחרים של סיווג, כמו למשל SVM או אפילו רשתות נוירונים.

כזכור, תחת ההנחה שקיים אלגוריתם לומד חלש $c(x)$, המטרה היא למצוא דרך להפוך אותו למודל חזק $C(x)$. באופן אינטואיטיבי היה ניתן לחשוב שאפשר פשוט לאמן מספר מודלים על תת קבוצות של הדאטה המקורי (עם אפשרות לחפיפות בין תת-קבוצות), להשתמש ב-majority vote, ובכך לשרשר את ההיפותזות של כל המודלים לפלט אחד. גישה זו כמובן נאיבית ופשטנית, ואינה לוקחת בחשבון מקרה בו מרבית המודלים שוגים. גישה טובה יותר תהיה לבנות מודל על בסיס חלק מהדאטה, לבחון את מידת ההצלחה של המודל על יתר הדאטה, ולפי ההצלחה שלו במשימה זו לתת משקל ל-vote של המודל. ניתן להוסיף תחכום לרעיון זה, כך שבכל שלב יינתן יותר דגש איברים בדאטה שהמודלים הקודמים שגו בסיווג שלהם, ובכך בכל שלב בו מאמנים מודל נוסף תהיה הצלחה יותר גדולה מאשר המודל הקודם. חלק זה הינו החלק האדפטיבי (Adaptive) באלגוריתם, על שמו נקרא האלגוריתם Adaboost.

כעת, נסביר כיצד ניתן להרכיב מסווג חזק באמצעות אוסף של מסווגים חלשים עבור אוסף נתונים $X \in \mathbb{R}^N$.

1. ראשית יש לאתחל משקולות באופן אחיד עבור כל אחת מ- N הדוגמאות בסט הנתונים $w_i^{t=0} = \frac{1}{N}$.

2. לאחר מכן יש לבצע איטרציות באופן הבא:

בניית מסווג אופטימלי $c_t(x)$ ביחס לאוסף הנתונים המשוקלל.

חישוב שגיאת הסיווג של $c_t(x)$: $\epsilon_t = \sum_i w_i^t \{c_t(x) \neq y_i\}$.

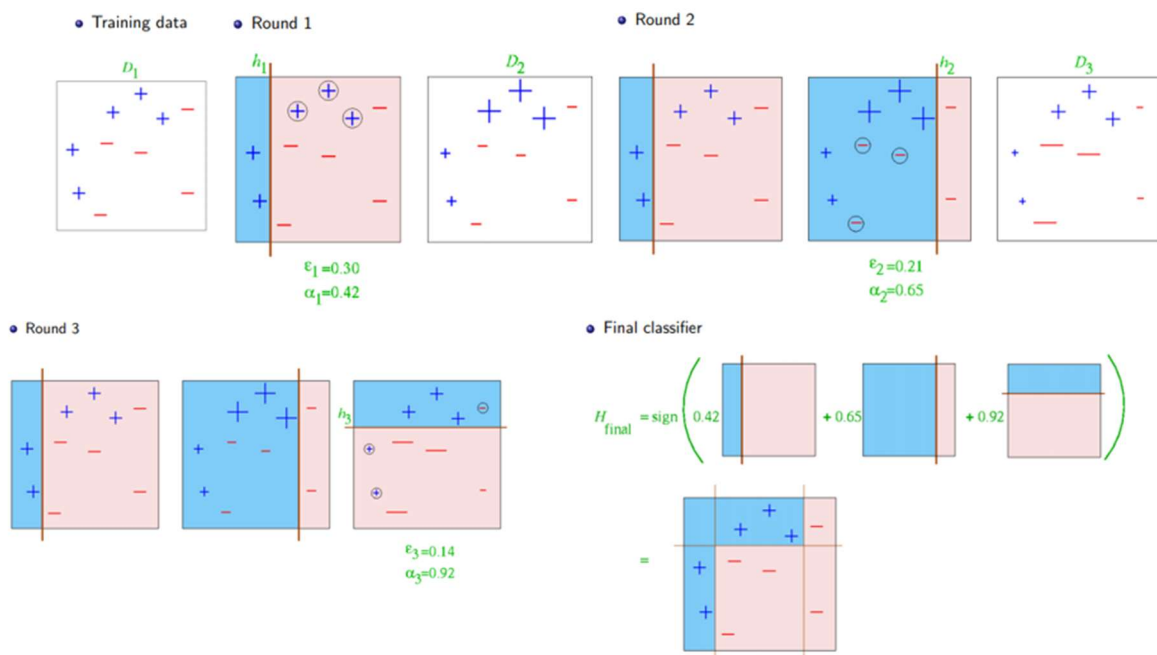
חישוב משקל עבור מסווג זה: $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon}{\epsilon} \right)$.

עדכון המשקלים: $w_i^{t+1} = w_i^t \exp(-\alpha_t y_i c_t(x_i))$.

נרמול המשקלים לפי הסכום שלהם: $N_{t+1} = \sum_i w_i^t \rightarrow w_i^{t+1} = \frac{w_i^t}{N_{t+1}}$.

3. חישוב המסווג המשוקלל, שהינו קומבינציה לינארית של המסווגים החלשים:

$$C(x) = \text{sign} \left(\sum_t \alpha_t c_t(x) \right)$$



איור 2.18 – דוגמה לשימוש ב-AdaBoost עבור מודל סיווג בינארי.

2. References

SVM:

https://commons.wikimedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png

<https://svm.michalhaltuf.cz/support-vector-machines/>

<https://medium.com/analytics-vidhya/how-to-classify-non-linear-data-to-linear-data-bb2df1a6b781>

https://xavierbourretsicotte.github.io/Kernel_feature_map.html

Naïve Bayes:

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

https://scikit-learn.org/stable/modules/naive_bayes.html

K-NN:

<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

EM:

https://www.cs.toronto.edu/~urtasun/courses/CSC411_Fall16/13_mog.pdf

https://stephens999.github.io/fiveMinuteStats/intro_to_em.html

Hierarchical Clustering:

<https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/>

LOF:

<https://towardsdatascience.com/local-outlier-factor-lof-algorithm-for-outlier-identification-8efb887d9843>

PCA:

Saal, L.H. et al. (2007). Proc. Natl. Acad. Sci. USA 104, 7564–7569.

t-SNE:

<https://nlml.github.io/in-raw-numpy/in-raw-numpy-t-sne/>