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 שייווצר 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 המאפיינים 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:

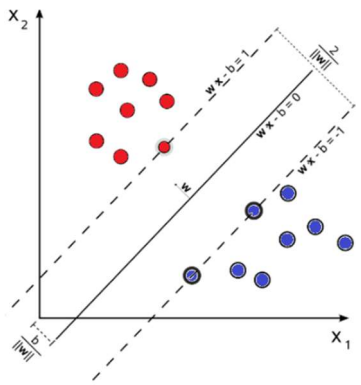
$$w^* = argmin_w \left(\frac{1}{2}||w||^2\right), s.t\ \forall i\ y_i(w \cdot x_i) \geq 1$$

כלומר, רוצים למצוא את וקטור המשקולות w^* המייצר שוליים $margin = \frac{1}{2}||w||^2$, 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 מצדדיו 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 המקווקוים 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 דוגמא 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)

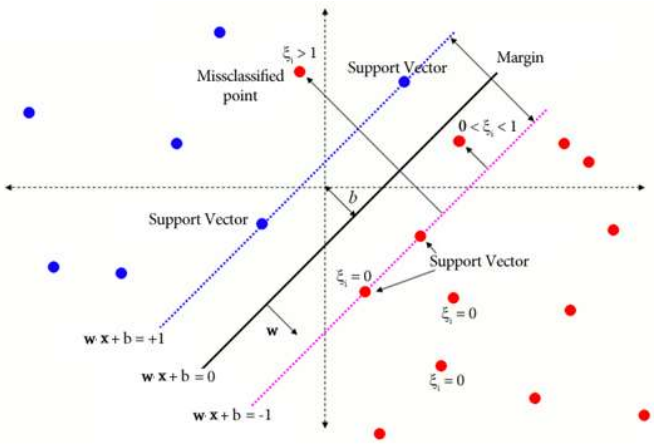
הפרדה מושלמת באמצעות מישור לינארי לעיתים קרובות איננה אפשרית, ולכן ניתן להרחיב את המודל כך שיאפשר לנקודות מסוימות לא להיות ב"צד" המתאים להן. הרחבה זו, היוצרת "הפרדה רכה", מאפשרת לטפל בבעיות שבהן אין הפרדה לינארית בין הקבוצות, כמו למשל שיש נקודות חריגות. משמעות ההרחבה היא שכל וקטור מפר לפחות אחד מהאילוצים,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

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

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

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 בגלל הפשטות 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 עבורו הפרדה 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 $\psi: \chi \to F$ 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):

$$K(\overrightarrow{x_i}, \overrightarrow{x_j}) = \psi(\overrightarrow{x_i})^T \psi(\overrightarrow{x_j})$$

דוגמאות של פונקציות קרנל נפוצות:

קרנל לינארי:

$$K(\overrightarrow{x_i}, \overrightarrow{x_j}) = \overrightarrow{x_i} \cdot \overrightarrow{x_j}$$

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:

$$K(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i} \cdot \overrightarrow{x_j} + c)^d$$

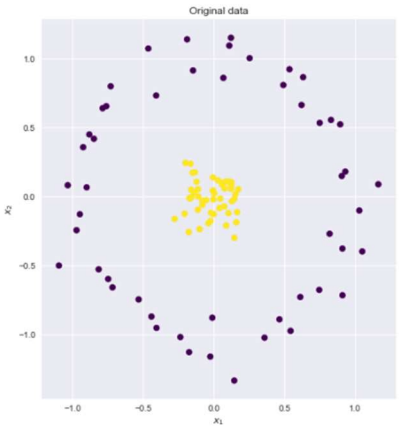
העתקה מהמרחב המקורי למרחב שמהווה פולינום ממעלה $d \geq 2$. $c \geq 0$ הוא פרמטר חופשי המשפיע על היחס בין סדר גבוה לעומת סדר נמוך בפולינום. כאשר $c = 0$, הקרנל נקרא הומוגני.

קרנל גאוסיאני:

$$K(\overrightarrow{x_i}, \overrightarrow{x_j}) = \exp\left(-\gamma\left(\overrightarrow{x_i} - \overrightarrow{x_j}\right)^2\right), \gamma > 0$$

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 יתר 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:

$$x_1, x_2 :\rightarrow z_1, z_2, z_3$$

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

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.

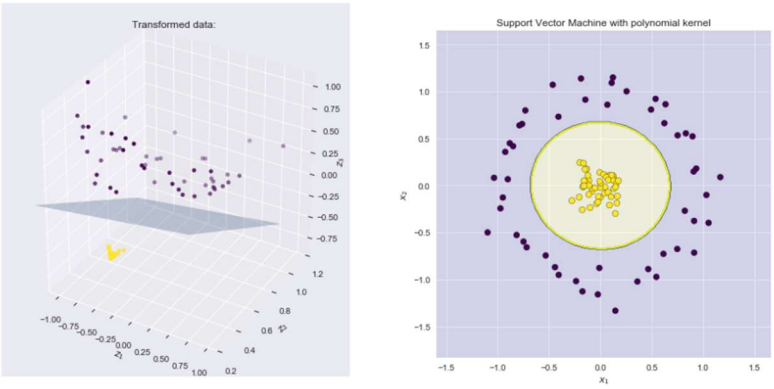
מתברר שמרחב התכונות שלעיל תואם את השימוש בקרנל פולינומי ידוע: $K(x, x') = (x^Tx')^d$. נבחר $d = 2$, נסמן $\boldsymbol{x} = (x_1, x_2)^T$ ונקבל:

$$K\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} x_1' \\ x_2' \end{pmatrix}\right) = (x_1x_2' + x_2x_2')^2 =$$

$$2x_1x_1'x_2x_2' + (x_1x_1')^2 + (x_2x_2')^2 = \left(\sqrt{2}x_1x_2x_1^2x_2^2\right)\begin{pmatrix} \sqrt{2}x_1'x'_2 \\ x_1'^2 \\ x_2'^2 \end{pmatrix}$$

$$K\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} x_1' \\ x_2' \end{pmatrix}\right) = \varphi(\boldsymbol{x})^T\varphi(\boldsymbol{x}')$$

$$\varphi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} \sqrt{2}x_1x_2 \\ x_1^2 \\ x_2^2 \end{pmatrix}$$



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 לינאריות.

2.1.2 Naive Bayes

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

נניח שיש מודל המקבל וקטור מאפיינים 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 $p(\text{חום}|\text{משתעל}) > p(\text{משתעל})$, אך ניתן להניח נאיביות ולקבל: $p(\text{חום}|\text{משתעל}, \text{שפעת}) = p(\text{משתעל}|\text{שפעת})$.

באופן כללי סיווג בייסיאני נאיבי מניח שבהינתן הסיווג של אובייקט מסוים, המאפיינים 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 לנבדק בדוגמה), ולאו 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 מאפיינים $x \in \mathbb{R}^n = (x_1, \dots x_n)$, היכול להיות שייך לאחת מ-K קטגוריות $y = (y_1, \dots y_k)$. ההתפלגות הפריורות $p(y_k)$ ידועה, ובנוסף ידועות ההתפלגויות המותנות של המאפיינים 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:

$$p(y_k, x) = p(y_k, x_1 \dots, x_n) = p(x_1|y_k, x_2 \dots, x_n) \cdot p(y_k, x_2 \dots, x_n)$$

$$= p(x_1|y_k, x_2 \dots, x_n) \cdot p(x_2|y_k, x_3 \dots, x_n) \cdot p(y_k, x_3 \dots, x_n)$$

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

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

$$= p(x_1|y_k) \cdot p(x_2|y_k) \cdots p(x_{n-1}|y_k) \cdot p(x_n|y_k)p(y_k)$$

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

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, המאפיינים קיבלו ערכים בדידים, ולכן היה ניתן לחשב את ההסתברות המותנית של כל מאפיין $p(x_i|y_k)$ על ידי ספירת כמות הפעמים שמופיע כל מאפיין 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 משוערכים בעזרת MLE או שיטת שערוך אחרת. אם ההתפלגות היא לא נורמלית, ניתן להשתמש באלגוריתם Kernel density estimation עבור שערוך ההתפלגות. גישה אחרת להתמודדות עם מאפיינים היכולים לקבל ערכים רציפים היא לבצע דיסקרטיזציה לערכים אותם המאפיינים 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:

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

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

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):

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

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

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

באופן כללי, עבור וקטור מאפיינים $x \in \mathbb{R}^n = (x_1, \dots x_n)$, 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 המאפיינים הם משתני ברנולי, המקבלים ערכים בינאריים. במקרה זו הנראות הינה:

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

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:

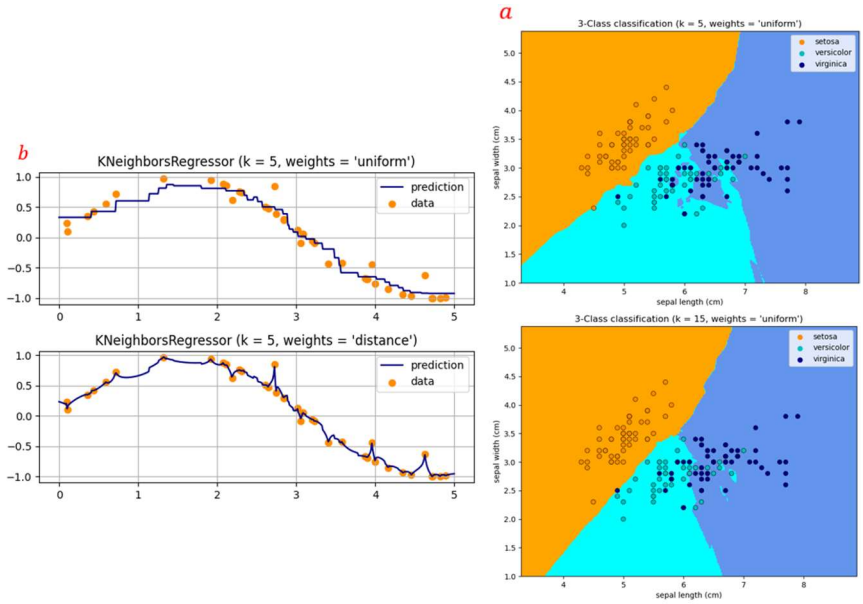
$$\arg\min_k p(y_k) \prod_{i=1}^{n} \frac{1}{p(x_i|y_k)}$$

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 שככל 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-y 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 מידי, 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 מידי, 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 (מיילים\קילומטרים). ההחלטה האם להשתמש בקילומטרים או במיילים 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 שאכן יש 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 – $p(y = k) = \frac{\mathbb{1}_{y=k}}{n}$.
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:

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

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

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

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 המשוערכים 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:

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

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:

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

וזהו 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:

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

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

והמסווג המתקבל הינו:

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

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

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:

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

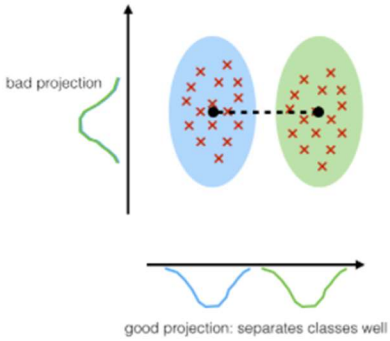
$$b = \frac{1}{2}(\mu_0^T\Sigma^{-1}\mu_0 - \mu_1^T\Sigma^{-1}\mu_1) + \log\frac{p(y=1)}{p(y=0)}$$

והסיווג הינו:

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

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.

2.1.5 Decision Trees

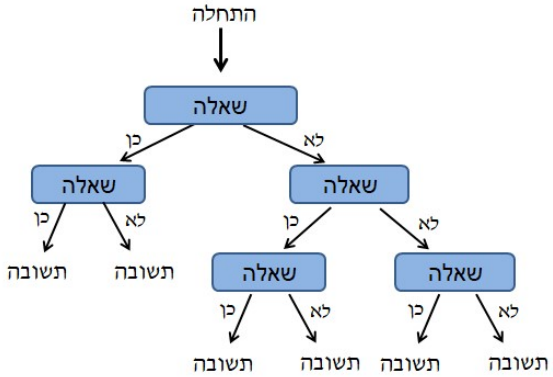
1. הקדמה

עץ החלטה הינו אלגוריתם לומד היכול לשמש הן לבעיות סיווג והן לבעיות רגרסיה. באופן כללי, עצי החלטה לוקחים את המשתנה שברצוננו להסביר (משתנה המטרה/החיזוי), ומחלקים את המרחב שלו לקבוצות (segments). לכל קבוצה של סט האימון מחשבים "ממוצע" (Mean) או "שכיח" (Mode), וכאשר מתקבלת תצפית חדשה משייכים אותה לקבוצה בעלת הממוצע או השכיח הדומה ביותר. מאחר וכללי הסיווג לקבוצות השונות יכולים להצטייר בצורת עץ, אלגוריתם זה נקרא "עץ החלטה".

הגישות השונות בעצי החלטה פשוטות ואינטואיטיביות להבנה, אולם הן לא מצליחות להתחרות במדדי הדיוק של מודלים אחרים של Supervised Learning. לכן, בפרקים הבאים נציג שיטות ensembles, בהן מתבצעת בנייה של כמה עצי-החלטה במקביל, אשר משולבים בסופו של דבר למודל יחיד. ניתן להראות ששימוש במספר גדול של עצים

יכול לשפר דרמטית את מדדי הדיוק של המודל. עם זאת, ככל שמשתמשים ביותר עצים כך יכולת הפרשנות של המודל נהיית מורכבת יותר ופחות אינטואיטיבית לצופה שאינו מעורב בבניית המודל (למשל גורם עסקי בארגון שאנחנו מעוניינים להסביר לו את תוצאות המודל).

ראשית נתבונן במבנה בסיסי של עץ ונגדיר עבורו את המושגים הרלוונטיים:



איור 2.6 דיאגרמה של עץ החלטה.

על מנת להבין את מבנה העץ ולייצר שפה משותפת נציג את השמות המקובלים בעבודה עם עצים:

- Root (שורש) – נקודת הכניסה לעץ (חלקו העליון ביותר של העץ).
- Node (צומת) – נקודת ההחלטה/פיצול של העץ – השאלות.
- Leaves (עלים) – הקצוות של העצים – התשובות. נקראים גם terminal nodes.
- Branch (ענף) – חלק מתוך העץ המלא (תת-עץ)
- Depth (עומק) – מספר ה- nodes במסלול הארוך ביותר בעץ.

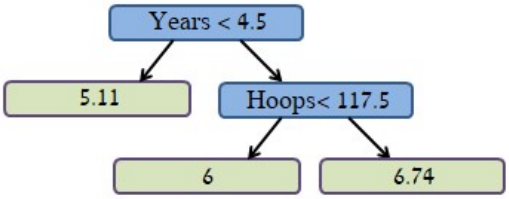
יסודות של עצי החלטה

עצי רגרסיה:

כאמור, עצי החלטה יכולים לשמש הן עבור סיווג והן עבור רגרסיה, ונתחיל עם דוגמה פשוטה לבניית עץ עבור בעיית רגרסיה – חיזוי שכר (באלפי שקלים) שחקני כדורסל באמצעות עצי החלטה. נרצה לחזות את השכר של שחקן בהתבסס על הנתונים הבאים:

- שנים - מספר העונות שאותו ששחקן שיחק בליגת העל.
- סלים - מספר הסלים שהוא קלע בשנה הקודמת.

תחילה נסיר תצפיות ללא ערכים עבור המשתנה המוסבר שלנו, הלא הוא "שכר" ובנוסף נבצע על אותו משתנה Log-transform בכדי שהוא יהיה ככל הניתן בקירוב להתפלגות נורמלית (עקומת גאוס/פעמון).



איור 2.7 דוגמה של עץ החלטה עבור בעיית רגרסיה של עץ החלטה.

כאמור, האיור מתאר עץ המנסה לחזות את Log השכר (באלפי שקלים) של שחקן בהינתן מספר עונות הותק שלו וכמות הסלים שקלע בעונה הקודמת. ניתן לראות שחלקו העליון של העץ (ה-Root) מתחלק ל-2 ענפים. הענף השמאלי מתייחס לשחקנים בעלי ותק של יותר מ4.5 עונות ($years < 4.5$), והענף הימני מתייחס לשחקנים פחות ותיקים. לעץ יש 2 צמתים (=שאלות/Internal nodes) ו-3 עלים (=תשובות/Terminal nodes). המספר בכל עלה שבתחתית העץ הוא הממוצע של כל התצפיות שסווגו לעלה הזה. לאחר שבניית העץ הסתיימה, כל תצפית חדשה

שתסווג לעלה מסוים תקבל את הערך הממוצע של התצפיות שעל בסיסם נבנה העץ. בהמשך הפרק יוסבר כיצד לבנות את העץ וכיצד לקבוע את כללי הפיצול בהתאם לדאטה הקיים.

בסופו של דבר העץ מחלק את השחקנים לשלוש קבוצות:

- שחקנים ששיחקו 4 עונות או פחות:

$$R_1 = \{X|Years < 4.5, Hoops\} = 5.11$$

- שחקנים ששיחקו 5 עונות או יותר וקלעו פחות מ- 118 סלים בעונה שחלפה:

$$R_2 = \{X|Years > 4.5, Hoops < 117.5\} = 6$$

- שחקנים ששיחקו 5 עונות או יותר ושקלעו יותר מ- 118 סלים בעונה שחלפה:

$$R_3 = \{X|Years > 4.5, Hoops > 117.5\} = 6.74$$

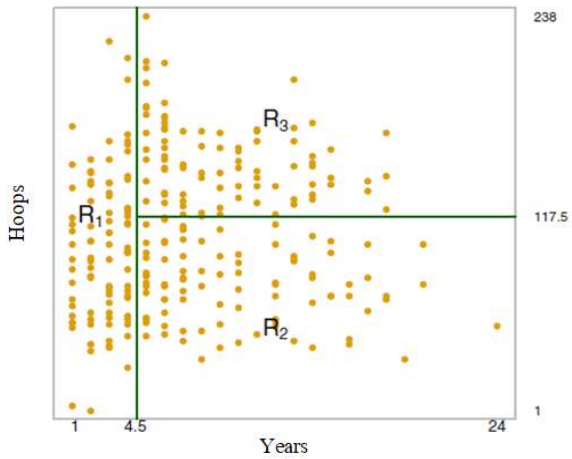
בהתאם לאנלוגיה של העץ, הקבוצות R_1, R_2, R_3 מכונות בשם Terminal nodes, או "עלים" של העץ.

אפשר לפרש את עץ הרגרסיה שבדוגמה הצורה נוספת: Years הוא הפרמטר המרכזי ביותר בקביעה מה יהיה גובה השכר, ושחקנים עם פחות שנות ניסיון ירוויחו פחות משחקנים מנוסים יותר. עבור שחקן יחסית חדש (עד 5 שנים בליגה) הפרמטר של כמות הסלים אותה הוא קלע בעונה הקודמת מתברר כפחות משפיע על השכר. כלומר, כל עוד לשחקן אין 5 שנות ניסיון, כמות הסלים היא יחסית שולית ביחס לחוסר הניסיון עבור קביעת שכרו. לעומת זאת, בקרב שחקנים עם 4.5 שנות ניסיון או יותר, כמות הסלים שהם קלעו בשנה הקודמת כן משפיעה על השכר, ושחקנים שקלעו הרבה סלים בשנה הקודמת כנראה ירוויחו יותר כסף מאשר שחקנים עם אותו ניסיון אך קלעו פחות סלים.

עץ הרגרסיה שהובא בדוגמא מפשט קצת "יתר על המידה" את הקשר "האמיתי" בין years, Hoops ו- Salary, והחיזוי של העץ לא מספיק טוב ביחס למודלים אחרים של רגרסיה, כמו למשל רגרסיה לינארית שתוסבר בפרק 3. עם זאת, מודל זה פשוט יותר להבנה ופרשנות וקל יחסית לייצוג באמצעים גרפיים.

2. חיזוי באמצעות ריבוד (Stratification) של מרחב המשתנים:

עד כה הוסבר באופן אינטואיטיבי מהו עץ החלטה וכיצד הוא פועל. האתגר המרכזי הוא לבנות את העץ בצורה טובה כך שהחיזוי שלו אכן יהיה תואם למציאות. בכדי להבין כיצד בונים עץ בצורה יעילה, נציג את הדוגמה הקודמת באופן נוסף:



איור 2.8 דוגמה של עץ החלטה עבור בעיית רגרסיה של עץ החלטה.

באיור זה ניתן לראות שהנתונים נפרסו על פני מרחב דו ממדי, וחולקו בעזרת קווי החלטה בהתאם ל- Internal nodes. כעת נגדיר את תהליך החלוקה והחיזוי בשני שלבים:

1. מחלקים את מרחב הערכים האפשריים של המשתנה אותו רוצים לחזות ל-J אזורים נפרדים $R_1, \dots, R_J$ כתלות בפרמטרים השונים $X_1, \dots, X_n$.
2. לאחר החלוקה, כל תצפית שנופלת באזור R_i תקבל את ערך הממוצע של הנקודות מסט האימון שמרכיבות את הקבוצה R_i.

באופן תיאורטי, לכל אזור יכולה להיות כל צורה שהיא. אולם לטובת הפשטות, אנו בוחרים לחלק את המשתנה ל-"אזורים מרובעים". המטרה היא למצוא את האזורים שמביאים למינימום את סכום ריבועי ההפרשים בין התוויות של הנתונים בסט האימון לבין הערכים של כל אזור. מדד זה נקרא residual sum of squares (RSS), שמוגדר כך:

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2$$

כאשר $\hat{y}_{R_j}$ הוא ממוצע המשתנים של תצפיות האימון באזור j, ו-$y_i - \hat{y}_{R_j}$ הוא המרחק בין כל תצפית לבין הערך הממוצע באזור זה. כאמור, המטרה של מדד זה היא למצוא את מקבץ התצפיות בכל אזור, כך שריבוע המרחק בין הערך הממוצע לבין כל תצפית יהיה מינימלי. כיוון שבדיקת כל החלוקות השונות האפשריות אינה ריאלית מבחינה חישובית, לרוב משתמשים באלגוריתם חמדן (greedy) אשר עובד "מלמעלה למטה". גישה זו ביחס לעץ החלטה נקראת "פיצול בינארי רקורסיבי" (recursive binary splitting), והיא נקראת "מלמעלה למטה" כיוון שהיא מתחילה מראש העץ (root), היכן שכל התצפיות עדיין שייכות לקבוצה אחת גדולה. לאחר סימון נקודת ההתחלה, האלגוריתם מפצל את מרחב הערכים של המשתנה אותו רוצים לחזות, כאשר כל פיצול מסומן באמצעות שני ענפים חדשים בהמשך העץ.

האלגוריתם כאמור הוא חמדן, וזה מתבטא בכך שבכל שלב בתהליך בניית העץ בוחרים לבצע את הפיצול הטוב ביותר עבור השלב הנוכחי, מבלי להתחשב כמה שלבים קדימה. בגישה זו יתכן ובשלב הנוכחי יש פיצול יותר יעיל לטווח ארוך, אך הוא לא יבחר אם הוא לא הפיצול האופטימלי בשלב זה. ניתן להמחיש את דרך הפעולה של אלגוריתם חמדן לעמידה בפקק תנועה, ומתוך ניסיון לעקוף את הפקק נבחרת הפניה הראשונה שנראית פחות פקוקה, מבלי להתחשב בפקק שעשוי לבוא בהמשך. כמובן שפניה זו לא בהכרח תוביל לדרך יותר מהירה, ויתכן שבראייה יותר ארוכת טווח דווקא היה מוטב להימנע מפנייה זו כיוון שהיא מובילה לפקק גדול יותר בהמשך.

כדי לבצע את אותו "פיצול בינארי רקורסיבי", יש לבצע את התהליך האיטרטיבי הבא:

עבור כל פרמטר אפשרי X_j וקו חלוקה s, נקבל שתי קבוצות:

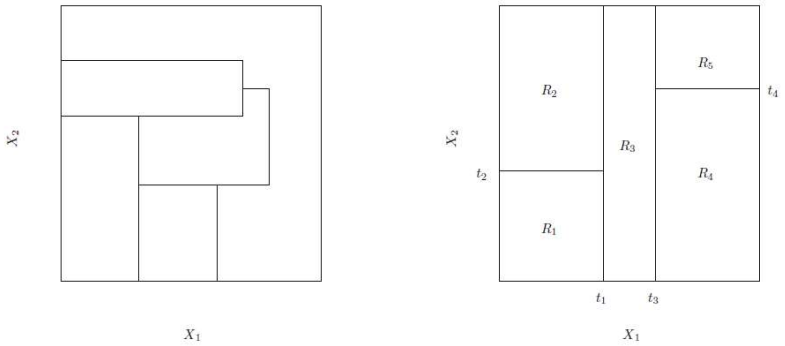
$$R_1(j,s) = \{X | X_j < s\}$$
$$R_2(j,s) = \{X | X_j \geq s\}$$

עבור שתי קבוצות אלה נחפש את הערכים של j ו-s שמביאים למינימום את המשוואה הבאה:

$$\sum_{i:x_i \in R_1(j,s)} \left(y_i - \hat{y}_{R_1}\right)^2 + \sum_{i:x_i \in R_2(j,s)} \left(y_i - \hat{y}_{R_2}\right)^2$$

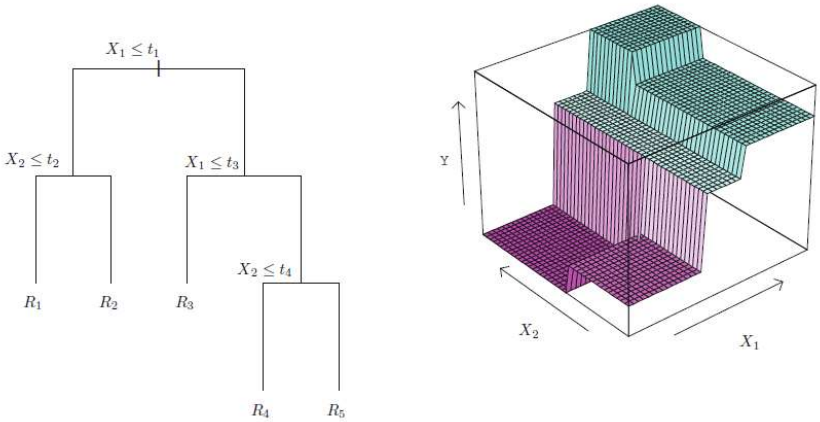
מציאת ערכי j, s שמביאים למינימום את המשוואה יכולה להתבצע די בקלות כשכמות הפרמטרים לא גדולה מידי, אך זה תהליך שעלול להיות די סבוך כשיש הרבה פרמטרים.

למעשה תהליך זה מייצר ענף אחד שיוצא מה-root, ולו 2 עלים. כעת מבצעים את התהליך שוב, מתוך מטרה למצוא את הפרמטר הבא שמביא למינימום את RSS בכל קבוצה, ובכך לקבל 3 קבוצות. ובאופן דומה, נרצה לפצל את אחת מאותן 3 קבוצות שכבר יש לנו כדי לצמצם עוד את ה- RSS. התהליך הזה נמשך איטרטיבית עד שמגיעים "לקריטריון עצירה" מסוים, כמו למשל מספר פיצולים, מספר מקסימלי של תצפיות בכל אזור וכו'. אחרי שהתבצעה החלוקה לקבוצות $R_1, \dots R_j$, ניתן להפוך את הקבוצות לעץ, ולהשתמש בו בכדי לחזות נקודות חדשות



איור 2.9 חלוקה של משתנים לאזורים.

באיור המצורף ניתן לראות שתי חלוקות – החלוקה הימנית הינה חלוקה דו-ממדית של שני משתנים באמצעות פיצול בינארי רקורסיבי. החלוקה השמאלית היא גם חלוקה דו-ממדית של שני משתנים, אך היא לא יכלה להיווצר באמצעות פיצול בינארי רקורסיבי, כיוון שהיא מורכבת מידי ואינה תואמת את הגישה החמדנית שרוצה "חלוקה פשוטה ומהירה".



איור 2.10 תיאור אזורי חלוקה כעץ בינארי. מצד שמאל מוצג העץ התואם לחלוקה מהאיור הקודם, ומצד ימין ישנה פרספקטיבה רחבה כיצד חיזויים מתבצעים באמצעות העץ.

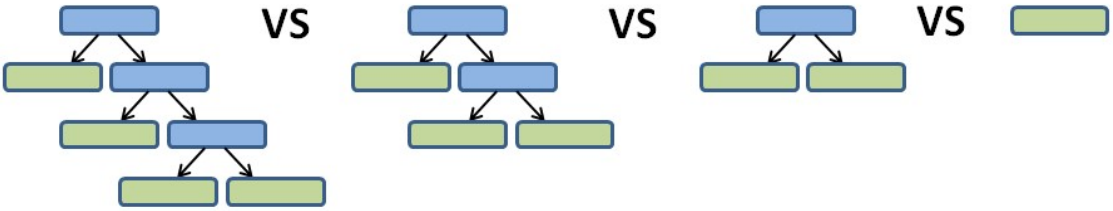
3. גיזום (pruning):

התהליך שתואר משקף את התהליך הקלאסי של יצירת עץ רגרסיה, והוא מסוגל לנפק חיזויים טובים (מבחינת מדד RSS ושונות נמוכה). עם זאת העץ עשוי לבצע התאמת-יתר (Over-fitting) על הנתונים, מה שיוביל לביצועים לא טובים עבור דאטה חדש. בעיה זו עלולה לנבוע מכך שהעץ שנבנה בתהליך האימון עשוי להיות "מורכב מידי" ומותאם יתר על המידה לנתוני האימון, מה שפוגע ביכולת ההכללה שלו. לעומת עץ מורכב, עץ דליל יותר בעל פחות פיצולים (כלומר פחות קבוצות $R_1, \dots, R_J$) ייתן חיזויים בעלי הטיה (Bias) גדולה יותר בהשוואה לאלטרנטיבה הראשונה, אך כנראה יוביל לשונות קטנה יותר בין סט האימון לבין סט המבחן, ובנוסף מודל כזה יהיה קל יותר לפרשנות.

גישה פשטנית בכדי להתמודד עם בעיה זו היא לקבוע רף כלשהו, כך שבכל פיצול הירידה ב-RSS תעלה על רף זה. כלומר, פיצול שמוביל לירידה שולית יחסית ב-RSS לא יתבצע. באופן הזה העצים שיתקבלו יהיו קטנים יותר ויש פחות חשש ל-over-fitting. החיסרון בגישה זו נעוץ בכך שהיא קצרת-ראייה. יתכן מצב בו יש פיצול בעל תרומה קטנה יחסית אך הוא יכול להוביל לפיצול אחר בעל תרומה גדולה, כזה שיביא לירידה גדולה ב-RSS בהמשך. שיטה זו תדלג על אותו פיצול בעל ערך קטן, מאחר והוא לא עובר את הרף שהוגדר.

גישה אחרת, שמתברר והיא טובה יותר, הולכת בכיוון ההפוך. בשלב הראשוני יבנה עץ גדול מאוד (T_0), ולאחר מכן נגזום ממנו כל מיני פיצולים בכדי להימנע מ-over-fitting. את מקומם של הענפים שהסרנו יתפוס עלה בודד שמקבל ערך ממוצע המתחשב במספר גדול יותר של תצפיות. כמובן שצעד זה יגרום לירידה בביצועים על פני סט האימון, אך השאיפה היא שזה ישתלם בבדיקת השגיאה בסט המבחן.

השאלה הגדולה היא כמובן מהי הדרך הטובה ביותר לגזום את העץ. אינטואיטיבית, המטרה שלנו היא לבחור subtree מתוך העץ המקורי שמוביל לשגיאה הקטנה ביותר. אומדן זה יכול להתבצע על ידי בדיקת השגיאה של העץ החדש ביחס לסט המבחן. כיוון שאי אפשר לבחון את כל תתי העצים האפשריים, נהוג להשתמש בגישה הנקראת Cost complexity pruning (ידועה גם בשם weakest link pruning). בגישה זו, במקום להתחשב בכל subtree אפשרי, אנו מסתכלים על רצף מסוים של subtrees שהם למעשה חלקים שונים המרכיבים את T_0 – העץ המקורי שנבנה בהתחלה:

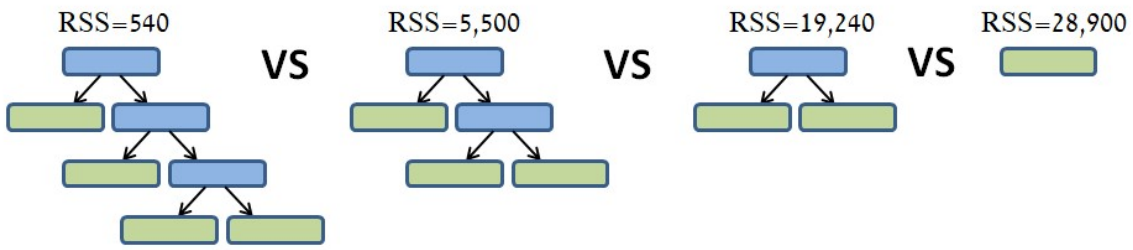


איור 2.11 חיפוש תת עץ אופטימלי ביחס ל-T_0. מתחילים מ-T_0 (העץ השמאלי) וכל פעם מורידים פיצול יחיד עד שמגיעים ל-root.

כעת מחשבים עבור כל אחד מרצף העצים שהתקבל את ה-RSS שלו. לפני שנסביר באיזה עץ לבחור, נסכם את השלבים שבוצעו עד כה:

א. בניית עץ רגרסיה גדול מכל הדאטה (ולא רק מסט האימון) בגישת ה- recursive binary splitting (הגישה החמדנית שתוארה לעיל). העץ ימשיך להבנות עד קריטריון עצירה מסוים (כמו למשל – הגעה למינימום של תצפיות). עץ הזה יסומן על ידי T_0.
ב. כל עלה בעץ מקבל את הערך הממוצע של תצפיות הפרמטר שבאותו עלה, ועבור כל עלה פיצול מחושב ה-RSS.
ג. סכימת כל ערכי ה-RSS שקיבלנו בכל העלים. הסכום שהתקבל הוא ה-RSS של כל העץ T_0.
ד. כעת מבצעים את שלבים א-ג ל-subtree הבא ברצף. זהו עץ-משנה שכמעט זהה לעץ המקורי, למעט שני עלים שנגזמו מהעץ הקודם. כך חוזרים על התהליך עבור כל ה-subtrees , עד ל-subtree האחרון שמורכב רק מה-root של העץ המקורי.

התוצר של השלב האחרון הוא RSS לכל subtree ברצף העצים. ניתן להבחין כי בכל פעם שענף מסוים הוסר, ה-RSS שהתקבל נהיה גדול יותר לעומת ה-subtree מהשלב הקודם. תוצאה זו הגיונית, שהרי כל פיצול במקור נועד להקטין את השגיאה באמצעות הקטנת ה-RSS, ואילו גיזום עושה את ההיפך – הוא נועד למנוע over-fitting, גם במחיר של שגיאה גדולה יותר.



איור 2.12 ביצוע גיזום וחישוב RSS לכל תת עץ (subtree) שהתקבל.

ה. כעת יש לבחור את אחד מה-subtrees, וכאמור זה נעשה בגישת Cost complexity pruning. גישה זו משקללת עבור כל תת עץ את מדד ה-RSS שלו יחד עם מספר העלים בעץ. באופן פורמלי:

$$Tree\ score = RSS + \alpha T$$

כאשר T הוא מספר העלים בעץ (**T**erminal nodes) ו-α הוא פרמטר רגולריזציה הקובע את היחס בין מספר העלים לבין מדד ה-RSS. פרמטר זה מחושב באמצעות Cross-validation המבצע trade-off בין הרצון להגדיל את העץ ובכך להקטין את ה-RSS לבין הרצון להימנע עד כמה שניתן מ-over-fitting. המחובר αT מכונה Tree Complexity Penalty, וכאמור הוא נועד "לפצות" על ההפרש במספר העלים שבין ה-subtrees השונים שנבחנים.

משלב ד' כבר קיבלנו RSS לכל subtree, וכעת נשאר רק לחשב לכל אחד מהם את ה- Tree score.

נשים לב כי:

- כאשר $\alpha = 0$, העץ המקורי (T_0) יהיה בהכרח בעל ה-Tree score הנמוך ביותר, כיוון שכאשר $\alpha = 0$ אזי כל הרכיב αT בנוסחה שווה ל- 0. במקרה זה ה-Tree score יהיה זהה לגמרי למדד ה-RSS:

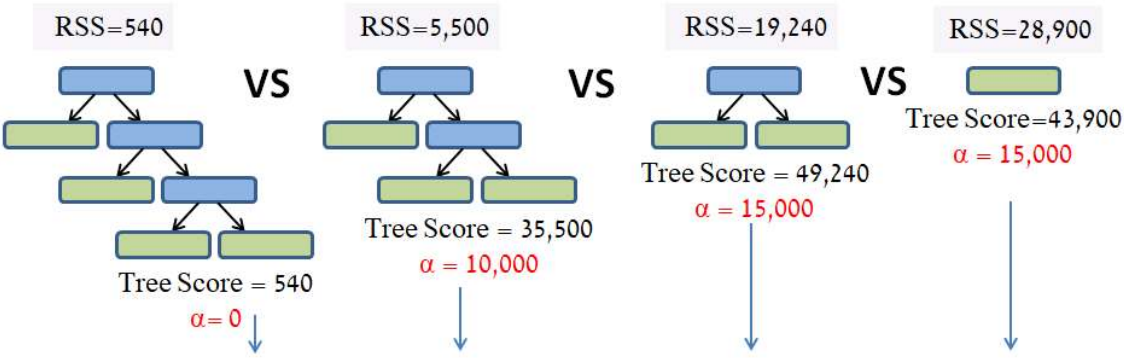
$$Tree\ score = RSS + \mathbf{0} \cdot T = RSS$$

ממילא שאר שלבי החישוב מיותרים, כי אנחנו כבר יודעים שהעץ T_0 הוא בעל ה-RSS הנמוך ביותר מכל ה-subtrees, וכן בעל ה-Tree Score הנמוך ביותר. **לכן, נרצה לקבוע ש-$\boldsymbol{\alpha > 0}$**. נתבונן מה קורה עבור ערכי α שונים:

ו. עבור $\alpha = 10{,}000$ הציון של T_0 יהיה 40,540. ולכן שווה לנו לגזום 2 עלים ולקבל עבור אותו α ציון נמוך יותר מ- 40,540 (העץ השני משמאל עם ציון של 35,500). ושוב, אם נשאר ב- $\alpha = 10{,}000$ ובמקביל נגזום 2 עלים נוספים (אנחנו כעת בעץ השלישי משמאל), נקבל ציון של 39,240 – שזה עדיין גבוה יותר מ-35,500 ולכן זה לא טוב לנו.

עבור $\alpha = 15{,}000$ ניתן לראות שהציון המתקבל יהיה נמוך יותר מה-subtree הקודם. כעת נשים לב שבמעבר ל-subtree האחרון (ה- root) לא משנה אם α יגדל או יישאר אותו דבר, בכל מקרה הציון של ה-root יהיה נמוך יותר מה-subtree הקודם.

ז. למעשה חזרנו על אותם צעדים עד שאין יותר מה לגזום. בסופו של תהליך ערכים שונים של α יתנו רצף של עצים, מעץ מלא ועד לעלה בודד. יוצא מכך שלכל ערך של α קיים subtree מתאים $T \subset T_0$ כך שה- Tree score המתקבל יהיה קטן ככל האפשר.



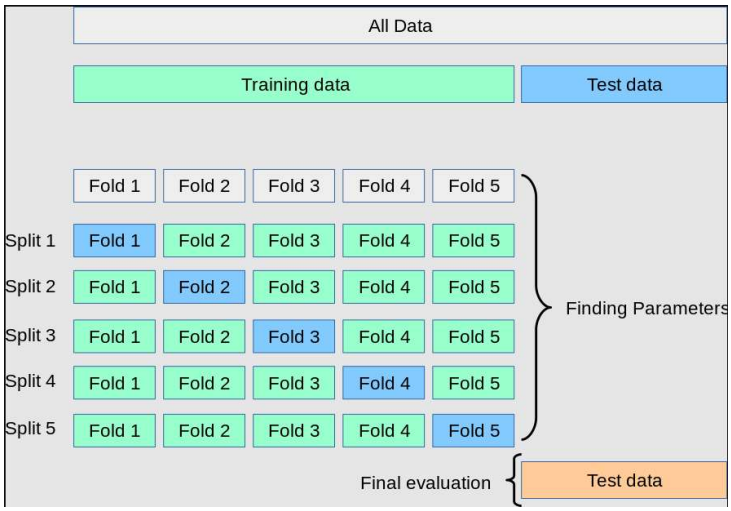
RSS		540	5,500	19,240	28,900
כמות העלים - T		4	3	2	1
Tree Score when	α=0	540	5,500	19,240	28,900
	α=10,000	~~40,540~~	35,500	39,240	38,900
	α=15,000	60,540	~~50,500~~	49,240	43,900
	α=22,000	88,540	71,500	~~63,240~~	50,900

איור 2.13 חישוב ערך α מותאם לכל עץ משנה כך שיתקבל Tree score קטן ככל האפשר.

ח. כעת, נבחר ב-subtree עם ה-score הנמוך ביותר – כלומר העץ השני משמאל עם ציון של 35,500. במקביל יש לזכור מהם ערכי ה-α של ה-subtrees השונים שהתקבלו (בסעיף הקודם), כיוון שהם יהיו שימושיים לחישוב ערך α אופטימלי:

- ערכי ה-α חשובים, כיוון שלכל α עשוי להתקבל עץ אחר בעל ציון מינימלי. עד לשלב זה העץ נבנה ביחס **לכל הדאטה** (בלי חלוקה ל- Train ו-Test), אך המטרה היא לבחון מהו ערך ה-α שייתן את התוצאה האופטימלית בעזרת העץ הגזום עבור דאטה חדש.

ט. כדי לבחור את ערך ה-α האופטימלי ניתן להשתמש ב-Cross-Validation. לשם כך, יש לקחת את הדאטה המלא (ממנו נבנה העץ הראשון – T_0), ולחלק אותו באופן הבא:



איור 2.14 cross-validation עם $K_{fold} = 5$.

- ב-cross-validation המודל מתאמן תחילה לפי split 1 (איור 2.14 לעיל) על התצפיות שבחלקים הירוקים, ובוחן את החיזויים על סט התצפיות הכחול.
- התהליך הזה נעשה K פעמים, כאשר בכל איטרציה האימון נעשה את התצפיות שבחלקים הירוקים, ובחינת החיזויים (וחישוב ה-RSS) נעשה על התצפיות שבחלק הכחול.

י. בכל אחד מה-splits ב-cross-validation המודל משתמש רק ב-Training data כדי לבנות עץ מלא (נסמן אותו הפעם ב-T_1) ורצף (sequence) חדש של subtrees שמביאים למינימום את ה-Tree Score (אותו סדר פעולות כמו בסעיפים א'-ד'), בעזרת אותם ערכי α שהתקבלו בסעיף ז'.

יא. כעת יש לחשב את ה-RSS לכל subtree באמצעות שימוש ב-Test data בלבד, ולזכור מיהו ה-Subtree שקיבל את ה-RSS הנמוך ביותר. נניח שבאיטרציה הראשונה העץ שקיבל את ה- RSS הנמוך ביותר ב-Testing data הוא דווקא העץ שבו $\alpha = 10{,}000$.

יב. את התהליך של סעיפים י-יא יש לבצע K פעמים – פעם אחת עבור כל split, כאשר בכל איטרציה האימון מתבצע על התצפיות שבחלקים הירוקים, ובחינת החיזויים (וחישוב ה-RSS) מתבצע על התצפיות שבחלק הכחול.

יג. בסופו של התהליך, כל איטרציה תניב subtrees בעלי RSS מסוים, וכן ערכי α שנקבעו כבר מראש בסעיף ז'. לאחר K האיטרציות יש לבדוק מיהו העץ עם ה-RSS המינימלי מבין כל האיטרציות, ומהו ערך ה-α של העץ הזה, **וזה יהיה הערך הסופי של α.**

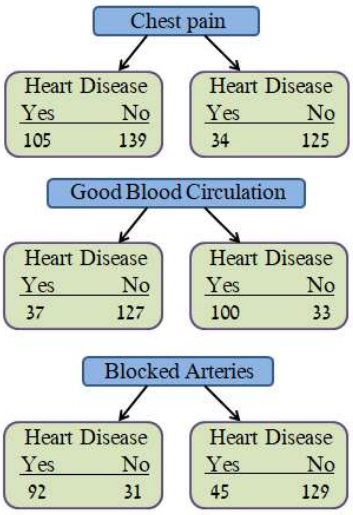
יד. לבסוף, יש לחזור לעץ המקורי T_0 וה-subtrees שנבנו מה-data set המלא (שמכיל גם את ה-Training וגם את ה-Test), ולבחור את העץ שתואם לערך ה-α הנבחר. ה-subtree הזה יהיה "העץ הגזום הסופי" שיבחר.

לסיכום:

- אחרי כל החישובים מצאנו שהעץ T_0 הוא בעל ה-RSS הנמוך ביותר, אך יתכן והוא יסבול מ-Overfit. כדי לפצות על כך, נוסף רכיב שנועד להתחשב גם ביתר העצים שהינם בעלי RSS גבוה יותר, ו-"מעניש" את העץ המקורי (T_0) עקב ריבוי העלים שבו.
- באופן הזה התקבל ציון המאפשר להשוות בין העצים ולבחור את העץ הטוב ביותר. העץ הזה מהווה מעין איזון בין הרצון ל-RSS מינימלי לבין שונות נמוכה בין ה-Train ל-Test .
- בכדי למצוא את ה-α האופטימלי, שמצד אחד נותן עץ בעל RSS אופטימלי ומצד שני נמנע כמה שניתן מ-Overfitting ניתן להיעזר ב-cross-validation.

4. עץ סיווג

עץ סיווג די דומה לעץ רגרסיה, רק שהמטרה היא שונה – במקום לקבל תשובה כמותית כמו ברגרסיה, עץ סיווג ייתן תווית (label) לתצפית המבוקשת. כאמור לעיל, עץ רגרסיה מספק חיזוי לתצפית מסוימת בהתאם לערך הממוצע של תצפיות האימון ששייכות לאותו terminal node. בעץ סיווג לעומת זאת, כל תצפית תשויך לקבוצה (class) בעל תווית משותפת. למשל, נניח ומעוניינים לסווג מטופל מסוים האם יש לו מחלת לב או לא. אנו יכולים לבנות עץ החלטה על בסיס מאפיינים של חולים שאובחנו בעבר ואנו יודעים להגיד מי מהם באמת חולה לב ומי לא, ועל בסיס העץ הזה להחליט עבור כל מטופל חדש האם הוא דומה במאפיינים שלו למטופלים שאובחנו בעבר כחולי לב או לא. כך שהתשובה שהעץ נותן היא לא "ערך ממוצע" כמו שראינו בעצי רגרסיה, אלא פשוט החלטה - "כן חולה לב" או "לא חולה לב". מלבד החיזוי של התווית, עץ סיווג מספק גם יחסים בין הקבוצות השונות בקרב תצפיות האימון שנופלים באותו אזור. נתבונן בדוגמה שתמחיש את העניין:



איור 2.15 השפעת פרמטרים שונים על הסיכוי לחלות במחלת לב.

באיור לעיל ניתן לראות שלושה פרמטרים (שבמקרה הזה הם סימפטומים של מטופל) בעזרתם מנסים לסווג האם למטופל יש מחלת לב או לא. כפי שניתן לראות אף אחד מהמשתנים אינו יכול לענות על שאלה זו בפני עצמו, כיוון שבאף אחד מהעלים אין אחידות בתצפיות. משתנים כאלה, אשר אינם יכולים בפני עצמם לספק סיווג מושלם, נקראים משתנים לא הומוגניים (impure - לא טהורים). כיוון שברוב המקרים כל המשתנים אינם הומוגניים, יש למצוא דרך כיצד לבחור באחד מהמשתנים להיות המשתנה שבראש העץ (Root node). כלומר, יש לייצר מדד הבוחן ומשווה את רמת ה-impurity של כל משתנה. ישנם מספר מדדים, ונתמקד בשניים מהם – Gini index ו-Entropy.

א. מדד Gini index:

נסמן את ההסתברות לשיוך תווית מסוימת לקבוצה j ב-p_j, ונגדיר:

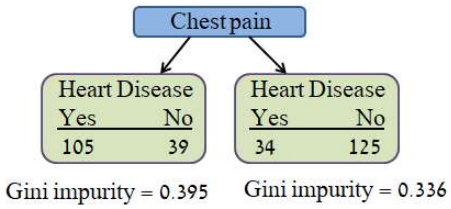
$$Gini = 1 - \sum_{j=1}^{J} p_j^2$$

באופן אינטואיטיבי, מדד Gini מייצג את הסיכוי לקבלת סיווג שגוי עבור בחירה רנדומלית של נקודה מהדאטה בהתאם לפרופורציות של כל class בדאטה. נדגים זאת על אחד הפרמטרים שבדוגמה הקודמת – Chest pain. עבור הענף הימני מתקיים:

$$Gini = 1 - \left(\frac{34}{34+125}\right)^2 - \left(\frac{125}{34+125}\right)^2 = 0.336$$

ועבור הענף השמאלי מתקיים:

$$Gini = 1 - \left(\frac{39}{39+105}\right)^2 - \left(\frac{105}{139+105}\right)^2 = 0.395$$



איור 2.16 חישוב מדד Gini עבור הפרמטר Chest pain.

אחרי שחישבנו את ה- Gini Impurity לשני העלים, נחשב את מדד Gini הכולל של כל המשתנה Chest Pain. בשביל חישוב זה יש לאזן בין מספר התצפיות שבכל עלה, באופן הבא:

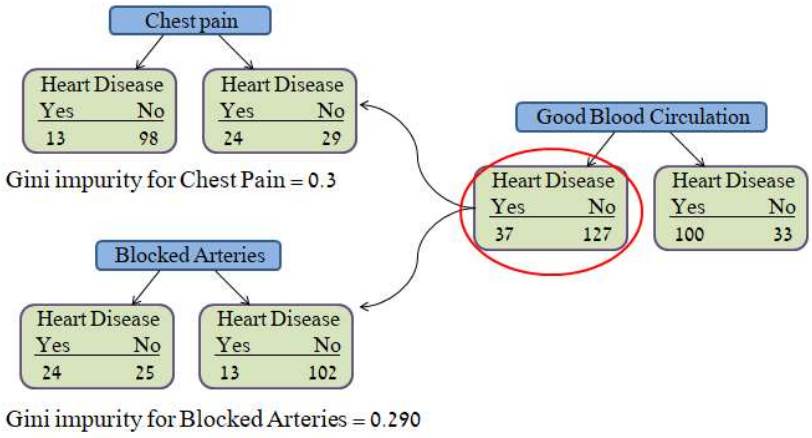
$$\text{Gini impurity for chest pain} = \left(\frac{144}{144+159}\right) \times 0.395 + \left(\frac{159}{144+159}\right) \times 0.336 = 0.364$$

באופן דומה ניתן לחשב את מדד Gini גם עבור יתר המשתנים ונקבל:

$$\text{Gini impurity for good blood circulation} = 0.36$$

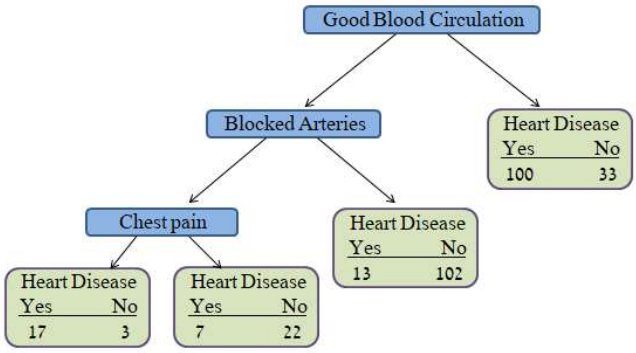
$$\text{Gini impurity for blocked arteries} = 0.381$$

למשתנה Good blood Circulation יש את הציון הכי נמוך, מה שאומר שהוא מסווג הכי טוב את המטופלים עם ובלי מחלת לב, ולכן נשתמש בו כמשתנה המסווג הראשון בראש העץ (ה- root). בכדי לקבוע את הפיצול הבא, יש להתבונן כיצד שאר הפרמטרים מסווגים את התצפיות של ה-root. נניח למשל ומתקיים:



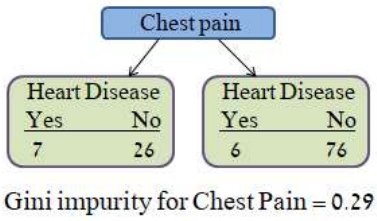
איור 2.17 חישוב מדד Gini עבור שאר הפרמטרים לאחר קביעת ה-root.

כפי שניתן לראות, למשתנה Blocked arteries יש ציון Gini נמוך יותר, ולכן הוא זה שנבחר להיות הפיצול הבא. לבסוף נבחן כיצד הפרמטר האחרון מסביר את התצפיות של הפיצול שלפניו, ונקבל את העץ הבא:



איור 2.18 חישוב מדד Gini עבור פיצול לפי הפרמטר Chest pain ביחס לשאר העץ.

נשים לב שניתן לפצל גם את העלה 13/102 לפי הפרמטר Chest pain, באופן הבא (המספרים כמובן תלויים בתצפיות האמיתיות):

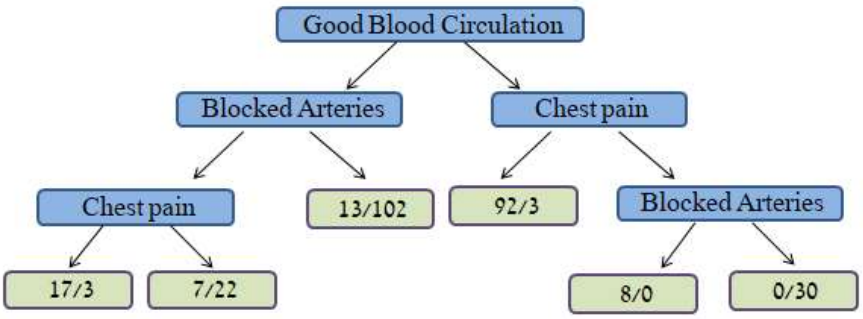


איור 2.19 חישוב מדד Gini עבור פיצול לפי הפרמטר Chest pain ביחס לעלה 13/102.

מדד Gini שהתקבל הינו 0.29, בעוד שבלי הפיצול המדד של העלה היה 0.2, ולכן במקרה הזה עדיף להשאיר אותו כפי שהיה לפני ניסיון הפיצול. כעת באופן דומה נבנה גם את הענף הימני של העץ:

1. נחשב את מדדי Gini.
2. אם לענף הקיים יש ציון Gini נמוך יותר, אז אין טעם לפצל עוד, והוא הופך להיות terminal node.
3. אם פיצול הענף הקיים מביא לשיפור, בוחרים את המשתנה המפצל בעל ציון Gini הנמוך ביותר.

עץ טיפוסי לאחר סיום התהליך נראה כך:

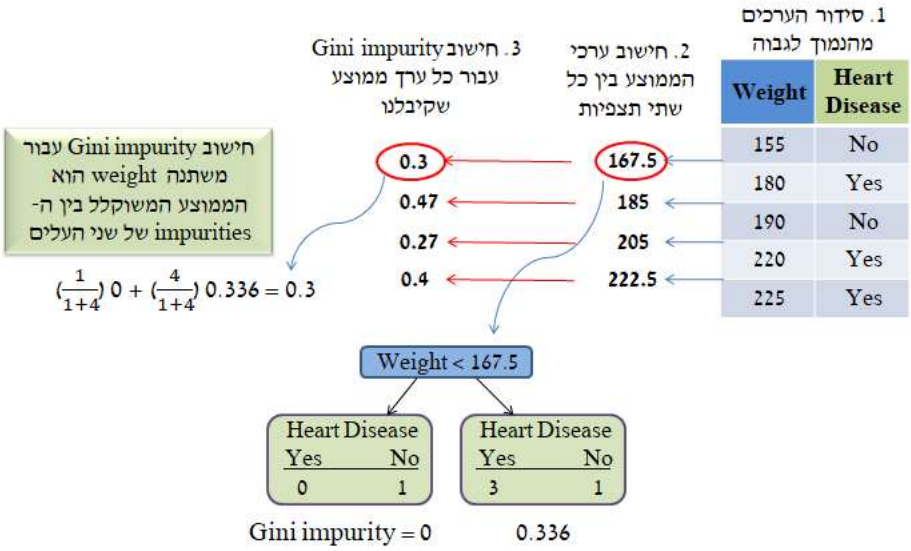


איור 2.20 חישוב מדד Gini עבור פיצול לפי הפרמטר Chest pain ביחס לעלה 13/102.

מדד Gini שהתקבל הינו 0.29, בעוד שבלי הפיצול המדד של העלה היה

התהליך שתואר מתאים למצבים בהם הפרמטרים מתפצלים באופן בינארי, כלומר הפיצול של כל פרמטר נקבע על ידי שאלה שעליה יש תשובה של כן או לא. במקרים בהם ישנם משתנים רציפים, הפיצול דורש כמה שלבים מקדימים:

1. סידור ערכי הפרמטרים מהערך הנמוך ביותר לערך הגבוה ביותר
2. חישוב הערך הממוצע בין כל 2 תצפיות.
3. לחישוב Gini impurity עבור כל ערך ממוצע שהתקבל בשלב הקודם.



איור 2.21 פיצול פרמטרים רציפים לפי מדד Gini.

אנחנו מקבלים את הGini הנמוך ביותר מתי שאנחנו קובעים Threshold של weight<205.

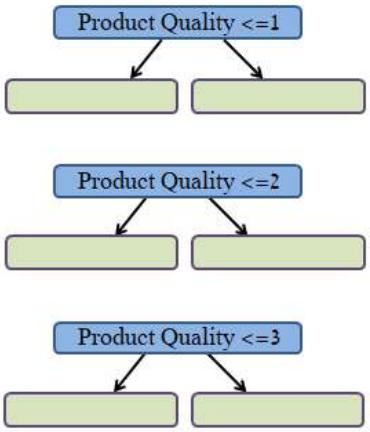
אז זהו ה-cutoff וערך הGini שנשתמש בהם כשאנחנו משווים את המשתנה weight ליתר המשתנים.

עד עכשיו דיברנו על איך מחלקים משתנה רציף ואיך מחלקים משתנה בינארי (שאלות כן/לא). כעת נדבר איך מחלקים משתנים קטגוריאליים. למשל: משתנה מדורג שמקבל ערכים מ-1 עד 4. למשל: טיב מוצר מ-1-4. או משתנה שמכיל מספר קטגוריות. למשל: צבע מועדף (מכיל ערכים: אדום, ירוק, כחול וכו').

משתנה מדורג מאוד דומה למשתנה רציף, חוץ מזה שאנחנו צריכים לחשב בו את הציון עבור כל חלוקה אפשרית.

Product Quality	Satisfied
1	No
1	Yes
3	No
1	Yes
וכו'...	וכו'...

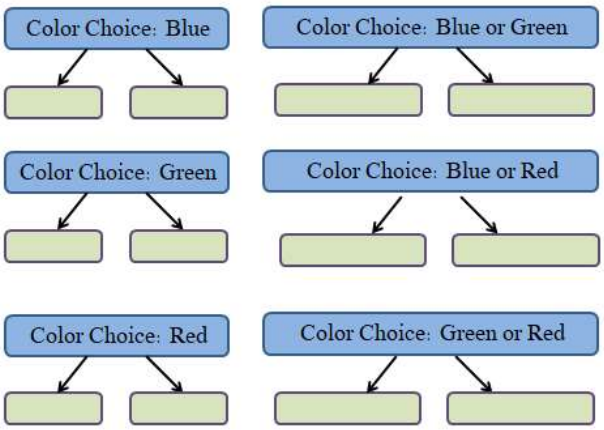
שימו לב: אנחנו לא צריכים לחשב את ה-impurity score ל- prod. quality<=4 בגלל שזה יכלול בעצם את כולם



איור 2.22 פיצול פרמטרים קטגוריאליים לפי מדד Gini.

כשיש משתנה קטגוריאלי עם כמה קטגוריות, אפשר לחשב את ציון הGini עבור כל קטגוריה, כמו גם עבור כל קומבינציה אפשרית:

Color Choice	Satisfied
Blue	No
Green	Yes
Red	No
Green	Yes
וכו'...	וכו'...



איור 2.23 פיצול פרמטרים קטגוריאליים לפי מדד Gini.

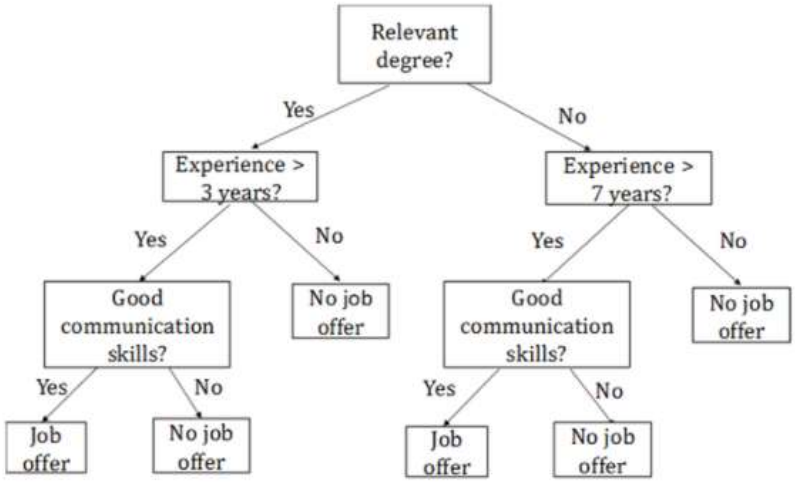
ב. מדד Entropy & Information Gain:

מדד נוסף לפיצול צמתי העץ מתבסס על האנטרופיה של העלים, בעזרתה ניתן לבחון את ה-information gain המקסימלי מכל פיצול. אנטרופיה באה למדוד את השגיאה של התפלגות המשתנה הנבחן מול משתנה המטרה. נניח וישנן n תוצאות אפשריות, כל אחת מהן בעלת הסתברות p_i, אז האנטרופיה מוגדרת באופן הבא:

$$H(x) = -\sum_{i=1}^{n} p_i \log p_i$$

בדומה למדד Gini, הפיצול האופטימלי נבחר על ידי המשתנה בעל מדד האנטרופיה הנמוך ביותר. אם כל התצפיות בעלה מסוים משויכות לאותו class, אזי מדד האנטרופיה יהיה 0. מאידך, כאשר בעלה מסוים יש התפלגות שווה בין 2 class-ים של המשתנה המוסבר, מדד האנטרופיה יהיה 1 (שזה הערך המקסימלי שמדד אנטרופיה יכול לקבל).

לעיל פירטנו שלב אחרי שלב את חישוב מדד Gini ל- Use-Case של חולי לב. כעת ניקח דוגמא אחרת פשוטה יותר הנוגעת לקבלת מועמד לתפקיד מסוים, כאשר מטרת העץ היא להחזיר "Yes" אם רוצים לקבל את המועמד, אחרת יוחזר "No".



איור 2.23 עץ סיווג עבור קבלת מועמד לתפקיד מסוים.

הדוגמה מתארת את אחד המאפיינים העיקריים של עצי החלטה – ההחלטה מתקבלת על ידי הסתכלות היררכית על הפרמטרים השונים, כאשר בכל פעם מתמקדים בפרמטר אחד ולא על כולם בבת אחת. תחילה, נלקח בחשבון המאפיין החשוב ביותר (תואר רלוונטי במקרה שלפנינו), לאחר מכן נלקחים שנות הניסיון, וכך הלאה.

נניח שהסיכוי בקרב כלל העובדים להתקבל לעבודה הוא 20%, והסיכוי לא להתקבל הוא 80%. במקרה זה האנטרופיה תחושב באופן הבא (הלוגריתם יכול להיות מחושב בכל בסיס, פה נלקח הבסיס הטבעי):

$$H = -0.2\ln 0.2 - 0.8\ln 0.8 = 0.5004$$

כעת נניח בנוסף ש-30% מהמועמדים בעלי תואר רלוונטי מקבלים הצעת עבודה ואילו מתוך אלה שאינם בעלי תואר רלוונטי רק 10% מקבלים הצעת עבודה. האנטרופיה עבור מועמד בעל תואר הינה:

$$H = -0.3\ln 0.3 - 0.7\ln 0.7 = 0.61$$

ועבור מועמד ללא תואר רלוונטי לתחום:

$$H = -0.1\ln 0.1 - 0.9\ln 0.9 = 0.32$$

נניח והמועמדים מתפלגים באופן שווה בין בעלי תואר לכאלה שאינם בעלי תואר, כלומר ל-50% מהמועמדים יש תואר רלוונטי ול-50% אין, הרי שתוחלת האנטרופיה במקרה זה הינה:

$$\mathbb{E}_{H(x)} = 0.5 \times 0.61 + 0.5 \times 0.32 = 0.46$$

לאחר כל החישובים, נוכל לבחון את רמת ה-impurity שמתקבלת מהידיעה האם למועמד מסוים יש תואר רלוונטי או לא. כלומר, כמה הידיעה שלמועמד מסוים יש תואר רלוונטי מפחיתה מחוסר הוודאות שלו לקבל את המשרה. אם אי הוודאות נמדדת באמצעות מדד ה-Entropy, הרי שהרווח מהמידע הוא:

$$\text{Information gain} = 0.5004 - 0.4680 = 0.0324$$

הן עבור מועמדים בעלי תואר והן עבור כאלה ללא תואר, המשתנה שממקסם את הרווח מהמידע הצפוי (ירידה באנטרופיה הצפוי) הוא מספר שנות הניסיון. כאשר למועמד יש תואר רלוונטי, הסף עבור "שנות ניסיון" הממקסם את הרווח מהמידע הצפוי הוא 3 שנים. עבור הענף התואם למועמד שאין לו תואר רלוונטי, הסף של שנות ניסיון הממקסם את הרווח מהמידע הצפוי הוא 7 שנים. לפיכך, שני הענפים הבאים הם: "ניסיון > 7" ו- "ניסיון ≥ 7". באותו האופן בונים את יתר העץ.

Misclassification rate

לאחר בניית עץ הסיווג, יש לבדוק את רמת הדיוק שלו על דאטה חדש. בעץ רגרסיה זה נעשה בעזרת מדד RSS, ובבעיות סיווג מקובל למדוד את ה-Misclassification rate. מדד זה בא לכמת את היחס בין כמות התצפיות שהמודל סיווג באופן שגוי לכמות הכוללת של התצפיות. במקרה זה פונקציית המחיר תהיה:

$$\mathcal{L}(\tilde{y}, y) = I\{\tilde{y} \neq y\}$$

פונקציית מחיר זו נקראת zero-one loss, כאשר תחת פונקציה זו חיזויים נכונים יקבלו ציון 0 ושגיאות יקבלו ציון 1, ללא תלות בגודל השגיאה. פונקציית ה misclassification rate נראית כך:

$$R(h) = \mathbb{E}[I\{h(x) \neq y\}]$$

סיכום

עץ החלטה (Decision Tree) הינו אלגוריתם לסיווג או לחיזוי ערכו של משתנה, כאשר המאפיינים מסודרים לפי סדר החשיבות. לצורך סיווג, קיימים שני מדדים אלטרנטיביים לאי וודאות: מדד אנטרופיה (Entropy) ומדד ג'יני (Gini). כאשר ערכו של משתנה מסוים נחזה, אי הוודאות נמדדת באמצעות RSS. חשיבותו של מאפיין הינו הרווח מהמידע הצפוי שלו (Expected Information Gain). הרווח מהמידע הצפוי נמדד על ידי הירידה באי הוודאות הצפויה אשר תתרחש כאשר יתקבל מידע אודות המאפיין.

במקרה של חלוקת **משתנה קטגוריאלי**, המידע המתקבל הינו על פי רוב אודות קטגוריה (Label) של המשתנה למשל: צבע מועדף (מכיל ערכים: אדום, ירוק, כחול וכו'). במקרה של **משתנה רציף** יש לקבוע ערך סף (Threshold) אחד (או יותר) המגדיר שני טווחים (או יותר) עבור ערכי המשתנה. ערכי סף אלו נקבעים באופן שממקסם את ה-information gain הצפוי.

אלגוריתם עץ ההחלטה קובע תחילה את צומת השורש (Root Node) האופטימלי של העץ באמצעות קריטריון "מקסום הרווח מהמידע" שהוגדר לעיל. לאחר מכן הוא ממשיך לעשות אותו הדבר עבור הצמתים העוקבים. הקצוות של הענפים הסופיים של העץ מכונים צמתי עלים (Leaf Nodes או Terminal Nodes).

במקרים בהם עץ ההחלטה משמש לסיווג, צמתי העלים כוללים בתוכם את ההסתברויות של כל אחת מהקטגוריות להיות הקטגוריה הנכונה. כאשר עץ ההחלטה משמש לחיזוי ערך נומרי לעומת זאת, אז צמתי העלים מספקים את ערך התוחלת של היעד. הגיאומטריה של העץ נקבעת באמצעות סט האימון (Training Set), אך הסטטיסטיקה שעוסקת ברמת הדיוק של העץ צריכה כמו תמיד בלמידת מכונה לבוא מתוך סט הבדיקה (Test Set) ולא רק מתוך סט האימון.

אחרית דבר:

מדדי RSS ומדד misclassification rate שהוצגו בפרק זה הינם רק חלק מהמדדים המקובלים. לכל מדד יתרונות וחסרונות, והמדד הרלוונטי יבחר בהתאם לסוג הנתונים והבעיה העסקית. נדון מעט בחוזקות ובחולשות של עצי החלטה:

<u>יתרונות:</u>

- בהשוואה לאלגוריתמים אחרים, עצי החלטה דורשים פחות השקעה בתהליך הכנת הנתונים (pre-processing).
- עץ החלטה לא דורש נרמול של הדאטה.
- ערכים חסרים בדאטה לא משפיעים על תהליך בניית העץ.
- עץ החלטה תואם לאופן שבו מרבית בני האדם חושבים על בעיה מסוימת והוא פשוט להסבר למי שאינם מומחים.
- אין שום דרישה שהקשר בין המשתנה המוסבר והמשתנים המסבירים יהיה לינארי.
- העץ בוחר אוטומטית במשתנים הטובים ביותר על מנת לבצע את החיזוי.
- עץ החלטה רגיש פחות לתצפיות חריגות מאשר רגרסיה.

<u>חסרונות:</u>

- שינוי קטן בנתונים יכול לגרום לשינוי גדול במבנה עץ ההחלטה ולגרום לחוסר יציבות.
- לפעמים החישוב בעצי החלטה יכול להיות מורכב מאוד ביחס לאלגוריתמים אחרים.
- עצי החלטה לעיתים תכופות דורשים יותר זמן הרצה לאימון המודל, ומשכך מדובר באלגוריתם "יקר" במשאבים.
- האלגוריתם של עץ ההחלטה אינו מספיק ליישום רגרסיה ולניבוי ערכים רציפים.
- עץ החלטה נוטה לעיתים תכופות לנטות ל-Overfitting.

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 כלשהיא, כך שאם ננסה להניח צורה, משולש לדוגמא, 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.

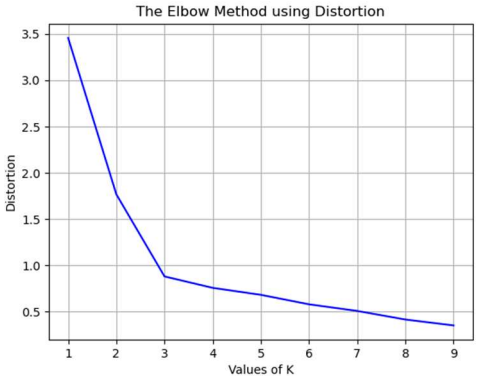
מדד WCSS הוא אחד הדרכים המקובלות ביותר להעריך את תוצאות החלוקה לאשכולות ב-K-means. היתרון של מדד זה הוא האפשרות לראות באופן כמותי את מידת ההצלחה של המודל, כלומר לקבל מספר ממשי שמכמת את הצלחת המודל. מנגד, WCSS הוא מספר ללא תחום מסוים והוא דורש פרשנות, כיוון שהערך והמשמעות שלו משתנים ממודל למודל. ערך מסוים 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:

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

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.

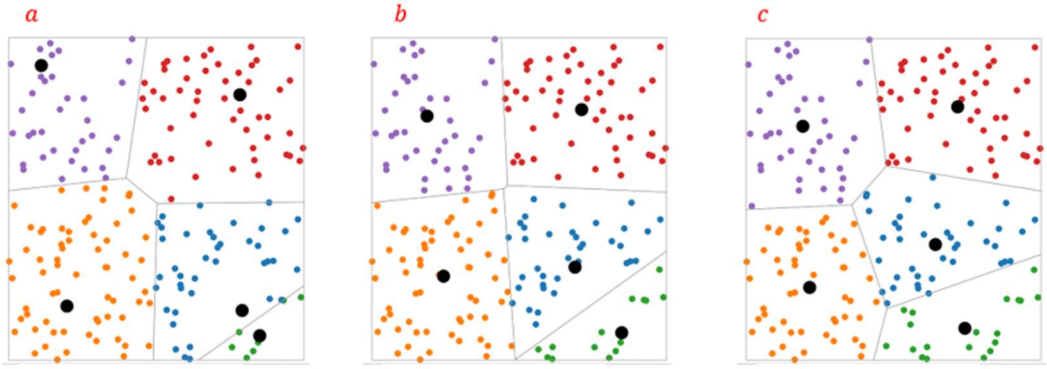
לכן מצד אחד, נרצה לבחור K גדול שימזער את ה-WCSS; מצד שני, הסיבה שהשתמשנו ב-K-means מלכתחילה היא בכדי לפשט את הנתונים למספר סביר של אשכולות, כזה שיאפשר לנו לערוך אנליזה נוחה. שיטת המרפק (Elbow method) היא טכניקה שמשמשת לפתרון סוגייה זו. הרעיון הוא לבחור את ה-K הקטן ביותר שממנו השיפור במדד ה- WCSS הוא 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.



איור 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. **שיוך:** 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). **שיוך** 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 זה 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 שככל 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:

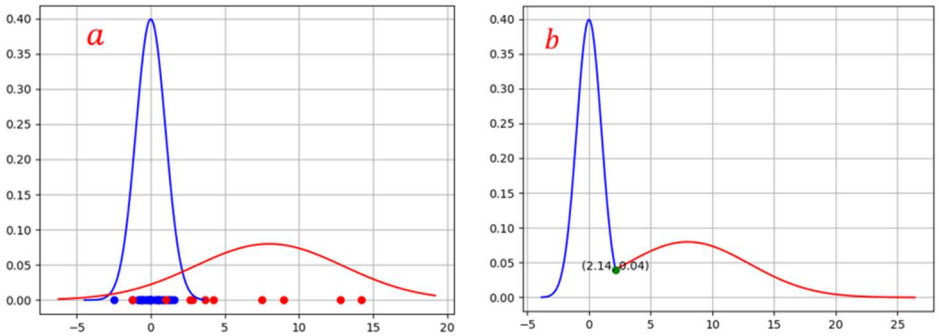
$$p(y=i|x)=\frac{p(y=i)\cdot p(x|y=i)}{p(x)}$$

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:

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

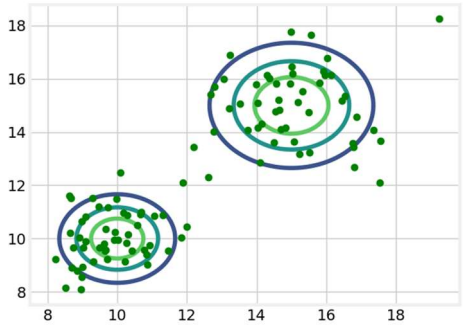


איור 2.8 a) תערובת של שני גאוסיאנים בממד אחד: בשלב ראשון מחלקים את הנקודות לשני אשכולות ומתאימים לכל אשכול התפלגות מסוימת. במקרה זה אשכול אחד (מסומן בכחול) הותאם להתפלגות $\mathcal{N}(0,1)$, ואשכול אחד (מסומן באדום) הותאם להתפלגות $\mathcal{N}(8,5)$. b) נקודה חדשה x תסווג לאשכול הכחול אם $x<2.14$, כיוון 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 שעבורה 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 בדו-ממד: אשכול אחד מתאים לגאוסיאן עם וקטור תוחלות $\mu_1 = [10, 10]$ ומטריצת covariance: $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, והאשכול השני מתאים לגאוסיאן עם וקטור תוחלות $\mu_1 = [15, 15]$ ומטריצת covariance: $\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$.

כיוון שהאלגוריתם mixture model מספק התפלגויות, ניתן להשתמש בו כמודל גנרטיבי, כלומר מודל שיודע 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:

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

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

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:

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

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 הפוסטריור 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:

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

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

$$=\sum_{i=1}^{n}\sum_{j=1}^{k}w_{ij}\log\alpha_j+\sum_{i=1}^{n}\sum_{j=1}^{k}w_{ij}\log\mathcal{N}\left(\mu_j,\sigma_j^2\right)$$

$$=\sum_{i=1}^{n}\sum_{j=1}^{k}w_{ij}\log\alpha_j-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{k}w_{ij}\left(\log\sigma_j^2+\frac{\left(x_i-\mu_j\right)^2}{\sigma_j^2}\right)$$

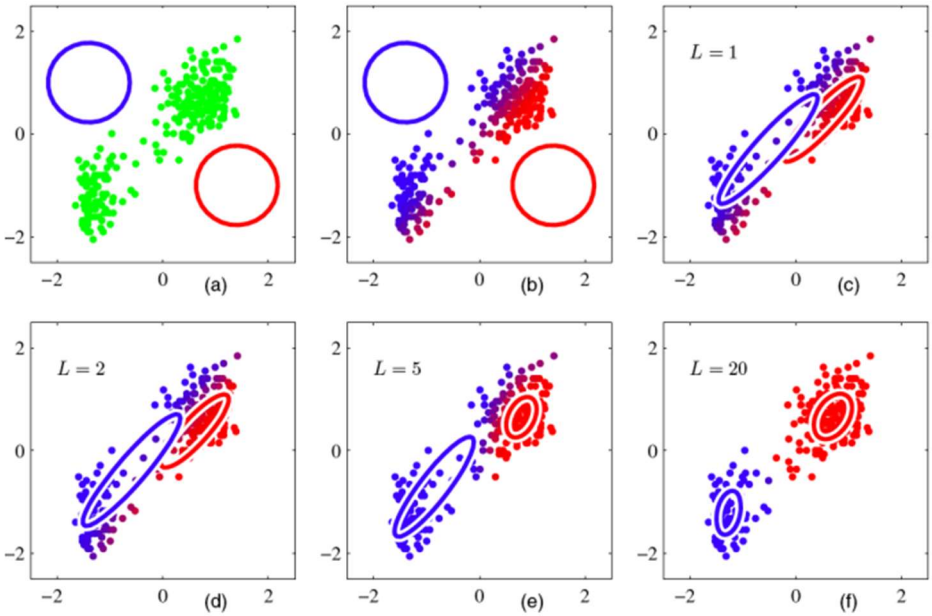
וכעת ניתן לגזור ולמצוא אופטימום:

$$\hat{\alpha}_j=\frac{1}{n}\sum_{i=1}^{n}w_{ij}$$

$$\hat{\mu}_j=\frac{\sum_{i=1}^{n}w_{ij}x_i}{\sum_{i=1}^{n}w_{ij}}$$

$$\hat{\sigma}_j^2=\frac{\sum_{i=1}^{n}w_{ij}\left(x_i-\mu_j\right)^2}{\sum_{i=1}^{n}w_{ij}}$$

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: $\log p(x;\theta) \geq \log p(x;\theta_0)$:

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

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

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

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:

$$\log p(x;\theta) - \log p(x;\theta_0) = Q(\theta,\theta_0) + H(\theta,\theta_0) - Q(\theta_0,\theta_0) - H(\theta_0,\theta_0)$$

$$= Q(\theta,\theta_0) - Q(\theta_0,\theta_0) + H(\theta,\theta_0) - H(\theta_0,\theta_0)$$

לפי אי-שיוויון 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: $\min\{d(a,b): a \in A, b \in B\}$.

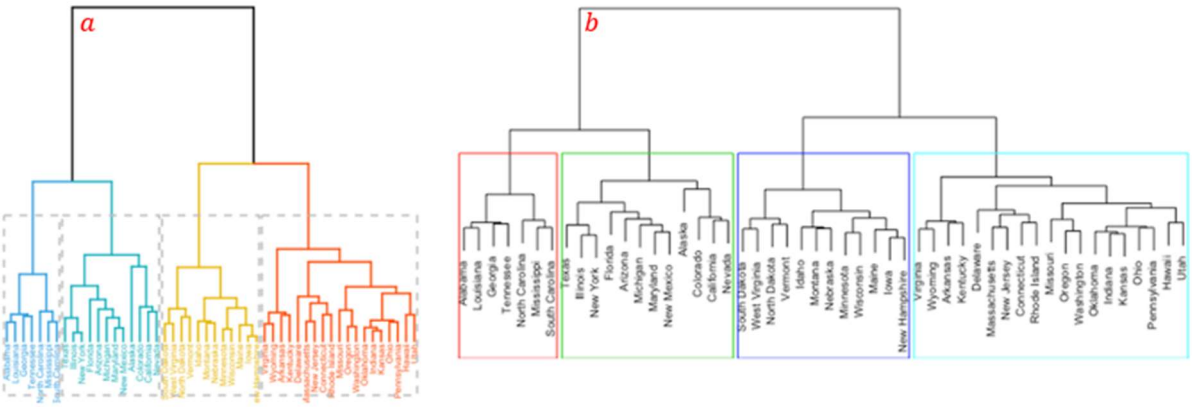
Unweighted average linkage clustering (UPGMA): $\frac{1}{|A|\cdot|B|}\sum_{a\in A}\sum_{b\in B} d(a,b)$.

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 ההולכים 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 החלקה 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 $RD(x_i, x_j) = \max\left(\text{k} - \text{distance of } x_i, distance(x_i, x_j)\right)$. 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 בוחנים 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.

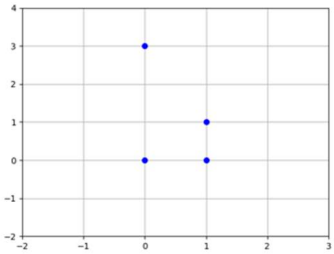
כדי להמחיש את התהליך נסתכל על האוסף הבא: $\{A = (0,0), B = (1,0), C = (1,1),\ D = (0,3)\}$, ונקבע $k = 2$. נחשב את ה-k-distance של כל נקודה במונחים של מרחק מנהטן:

$$k(A) = distance(A, C) = 2$$

$$k(B) = distance(B, A) = 1$$

$$k(C) = distance(C, A) = 2$$

$$k(D) = distance(D, C) = 3$$



נחשב את ה-LRD:

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

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

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

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

ולבסוף נחשב את ה-LOF:

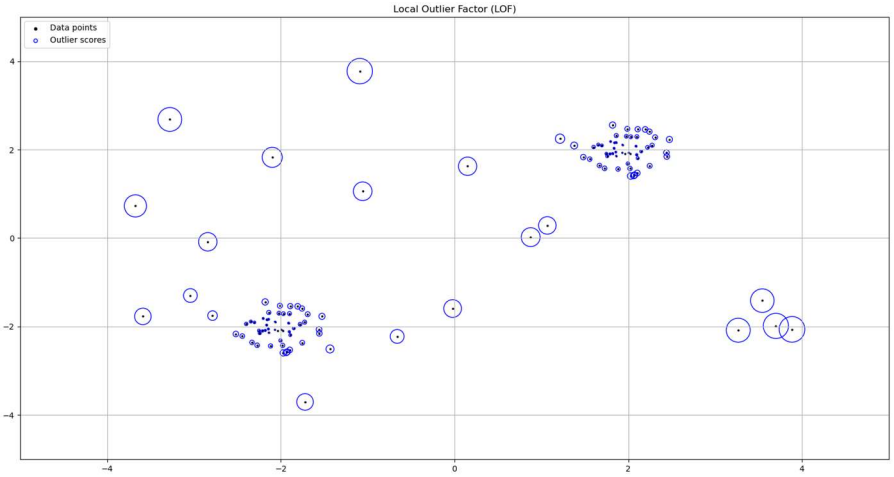
$$LOF_2(A) = \frac{LRD_2(B) + LRD_2(C)}{k} \times \frac{1}{LRD_2(A)} = 0.87$$

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

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

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

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 תזוהה 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 outliner אך עבור ערכי LOF גדולים מ-1 אין כלל חד משמעי עבור איזה ערך הנקודה היא outliner 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 בוויזואליזציה 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 לסווגן 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 מכלילים, 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 (principle component analysis) ובשתי טרנספורמציות לא-לינאריות (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 ליניארי 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-PCA2 יהיה אורתוגונלי ל-PCA1: $\mathrm{PCA}_1 \perp \mathrm{PCA}_2$. הגורם השלישי, הוא צירוף לינארי בעל השונות השלישית הגדולה ביותר, ומאונך לשני הגורמים הראשונים – $\mathrm{PCA}_2 \perp \mathrm{PCA}_3$ וגם $\mathrm{PCA}_1 \perp \mathrm{PCA}_3$, וכן הלאה כך שהגורם הראשי מסדר i, הוא בעל השונות ה-i-ית הגדולה ביותר תחת אילוץ של גורמים מאונכים – $\mathrm{PCA}_i \perp \mathrm{PCA}_j, \forall i < j$. 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 לינארית 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 השד, 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 $\hat{X} = \begin{bmatrix} \vec{X}_1 \\ \vdots \\ \vec{X}_M \end{bmatrix} = \left[\vec{X}^1, \dots, \vec{X}^N\right] \in \mathbb{R}^{M\times N}$, 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:

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

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 $\widehat{W}$:

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

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

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

כאשר המטריצה $\widehat{W} \in \mathbb{R}^{N\times K}$, 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: $\left\|\widehat{W}^k\right\|^2 = \sum_{m=1}^{M}\left(W_{m,k}\right)^2 = 1$.
2) השונות עבור המאפיין הk-י , המוגדרת על ידי $s_k^2 = \left(\vec{T}^k\right)^T \vec{T}^k = \sum_{m=1}^{M}\left(T_{mk}\right)^2$, מקיימת: $s_k^2 > s_{k+1}^2$.
3) העמודות של $\widehat{W}$ אורתוגונליות זו לזו, זאת אומרת $\widehat{W}^k \perp \widehat{W}^{k\prime}$ לכל שתי עמודות k, k'.

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

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

זאת אומרת:

$$\widehat{W}_1 = \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}(s_1^2) = \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}\left(\left(\vec{T}^1\right)^T \cdot \vec{T}^1\right) = \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}\left(\left(\hat{X}\widehat{W}^1\right)^T \cdot \hat{X}\widehat{W}^1\right)$$
$$= \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}\left(\left(\widehat{W}^1\right)^T\left(\hat{X}\right)^T \cdot \hat{X}\widehat{W}^1\right)$$

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

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

כאשר מטריצה $\hat{S} \in \mathbb{R}^{(N\times N)}$ הינה מטריצת השונות המשותפת (covariance), המוגדרת על ידי $\hat{S} = \left(\hat{X}\right)^T \cdot \hat{X}$. מטריצה זו, מסדר $N \times N$, מגדירה את השונות המשותפת בין שני מאפיינים, כאשר $S_{\nu_1,\nu_2} = \sum_{m=1}^{M} X_{\nu_1,m}X_{m,\nu_2}$. ניתן לשים לב כי מטריצה זו סימטרית וממשית (ולכן הרמיטית).

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

$$\lambda_1 = \underset{\|\widehat{W}\|=1}{max}\left((\widehat{W}^1)^T \cdot \hat{\mathrm{S}} \cdot \widehat{W}^1\right)$$

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

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

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

$$\widehat{W}^2 = \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}(s_2^2) = \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}\left(\left(\vec{T}^2\right)^T \cdot \vec{T}^2\right)$$
$$= \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}\left(\left(\widehat{W}^2\right)^T\left(\tilde{X} + \hat{X}\widehat{W}^1\left(\widehat{W}^1\right)^T\right)^T \cdot \left(\tilde{X} + \hat{X}\widehat{W}^1\left(\widehat{W}^1\right)^T\right)\ \widehat{W}^2\right)$$
$$= \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}\left(\left(\widehat{W}^2\right)^T\left(\tilde{X}\right)^T \cdot (\tilde{X})\ \widehat{W}^2\right)$$

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

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

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

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

כך שλ_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:

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

כאשר $\widehat{\Lambda}$ הינה מטריצה אלכסונית, ו-$\Lambda_{ii} = \lambda_i$ הינם הערכים העצמיים של $\hat{S}$ המסודרים לפי גודלם מהגדול לקטן - $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M$, ומטריצה $\widehat{W}$ מורכבת מווקטורי עמודה שהינם הווקטורים העצמיים המתאימים לערכים

העצמיים. הווקטורים העצמיים בהגדרתם הינם אורתוגונליים זה לזה, וכיוון ש-$\|\widehat{W}^k\| = 1$ לכל k, הם בעצם אורתונורמליים.

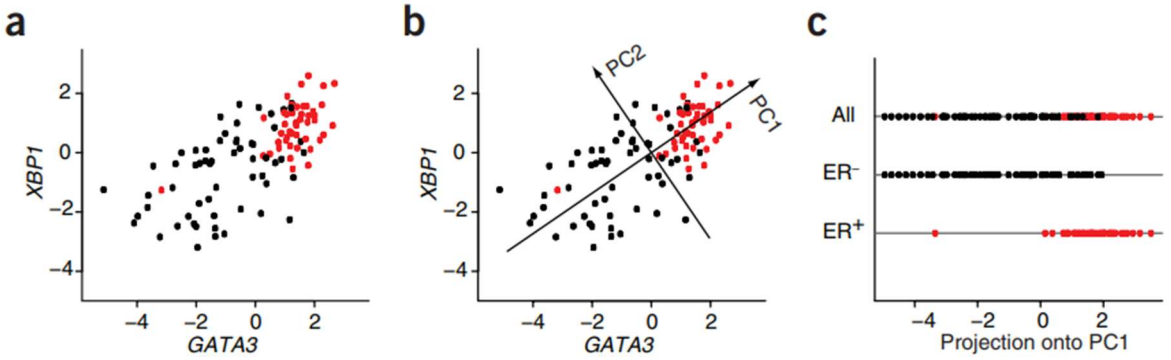
לסיכום, על מנת למצוא את הגורמים הראשיים עבור המידע הנתון $\hat{X}$:

א. "מרכז" את הנתונים כך שהממוצע עבור כל מאפיין הוא אפס: $\hat{X}^m = \hat{X}^m - mean_n(\hat{X}^m)$.
ב. מצא את מטריצת השונות המשותפת $\hat{S} = (\hat{X})^T \hat{X}$.
ג. מצא את $\hat{S} = \widehat{W} \cdot \widehat{\Lambda} \cdot \widehat{W}^T$.
ד. חשב $\hat{T} = \hat{X} \cdot \widehat{W}$.
ה. הגורמים הראשיים נתונים על ידי וקטורי העמודה $PCA_k \equiv \vec{W}^k$, וההטלה של מדידה m למערכת המאפיינים החדשה נתונה על ידי $\vec{T}^m = \vec{X}^m \widehat{W}$.

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 לדוגמא 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-ליניארית היא החשובה יותר. כמו כן, לעיתים "מרכוז" 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-ליניאריות ,לדוגמא שיטת kernel PCA, או שיטות להתמודדות עם בעיית המרכוז 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 האסטרוגן 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 האסטרוגן בתאי סרטן השד על ידי הגורם הראשי הראשון PCA_1, כפי שניתן לראות באיור. יש לשים לב שהגורם הראשי הראשון, PCA_1, מכיל מידע משני החלבונים.



איור 2.13 a) רמות ביטוי של שתי חלבונים GATA3 (ציר ה-X) ו-XBP1 (ציר ה-Y). קולטני אסטרוגן 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 PCA_1 הינה מקסימלית. c) הצגת תוצאות המדידה כפונקציה של PCA_1 בלבד. בגרף זה ניתן לראות בבירור כיצד הורדת הממד מסייעת למצוא הבחנה פשוטה (בממד אחד) בין קולטני האסטרוגן.

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:

$$mean_1(X^m) = \sum_{m=1}^{5} X_{m1} = -0.5 - 0.4 + 0.1 + 0.3 + 0.5 = 0$$

ועבור המאפיין השני:

$$mean_2(X^m) = \sum_{m=1}^{5} X_{m2} = -0.4 - 0.1 + 0 + 0.3 + 0.2 = 0$$

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:

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

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 $\widehat{W}^1$, מקיים:

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

כך ש:

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

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

כך שמטריצת המשקלים נתונה על ידי:

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

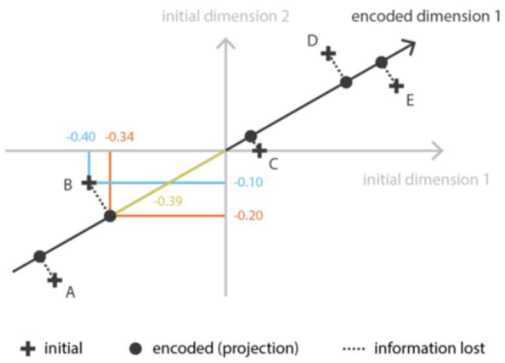
הטלת המדידות למערכת המאפיינים החדשה נתונה על ידי:

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

לכן, המדידות של הגורם הראשי הראשון, נתונות על ידי

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

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.

<u>נספח: משפט המינימום- מקסימום (קורנט-פישר-ויל):</u>

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

$$\lambda_m = \min_U \left\{ \max_{\substack{\vec{x} \in U, \\ \|x\|=1}} \left\{ x^\dagger S^\dagger x \middle| x \in U, x \neq 0 \right\} \middle| \dim(U) = M - m + 1 \right\}$$
$$= \min_U \left\{ \max_{\vec{x} \in U,} \left\{ \frac{x^\dagger S^\dagger x}{x^\dagger x} \middle| x \in U, x \neq 0 \right\} \middle| \dim(U) = M - m + 1 \right\}$$

הערך העצמי המקסימלי מקיים:

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

כאשר $\widehat{W}^1$, 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-$\mathcal{X}$ 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 $\mathcal{X}$ הינה הסתברותית. עבור שתי מדידות m_1, m_2 במרחב המקורי ה-N-ממדי, ההתפלגות הנורמלית המשותפת P_{m_1,m_2} הינה:

$$P_{m_1,m_2} = \frac{\mathcal{Z}_1^{-1}}{2N} \exp\left(-\frac{\left\|\vec{X}_{m_1} - \vec{X}_{m_2}\right\|^2}{2\sigma_1^2} \right) + \frac{\mathcal{Z}_2^{-1}}{2N} \exp\left(-\frac{\left\|\vec{X}_{m_1} - \vec{X}_{m_2}\right\|^2}{2\sigma_2^2} \right)$$

כאשר σ_i נקרא פרפלקסיות (perplexity) והוא פרמטר שנקבע מראש, ו-$\mathcal{Z}$ הינו קבוע הנורמליזציה, המוגדר על ידי $\mathcal{Z}_i = \sum_{k \neq i} \exp\left(-\frac{\|\vec{X}_i - \vec{X}_k\|^2}{2\sigma_i^2} \right)$. עבור נקודות קרובות יותר, עבורן הביטוי $\left\|\vec{X}_{m_1} - \vec{X}_{m_2}\right\|$ קטן, ההסתברות

שהנקודה $\vec{X}_{m_1}$ שכנה של $\vec{X}_{m_2}$ גדולה. לעומת זאת כאשר הנקודות רחוקות זו מזו, כלומר $\left\|\vec{X}_{m_1}-\vec{X}_{m_2}\right\|$ 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: $\begin{bmatrix}\vec{X}_1\\ \vdots\\ \vec{X}_M\end{bmatrix} \to \begin{bmatrix}\vec{Y}_1\\ \vdots\\ \vec{Y}_M\end{bmatrix}$, כך שהממד של $\vec{Y}_m$ הינו נמוך (2 או 3 ממדים). בנוסף, נדרוש שנקודות דומות ("שכנות") במרחב $\mathcal{X}$, 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 בפונקציות הסתברות אלו בהמשך). כך, נכמת את הדמיון בין m_1 לבין m_2, על ידי ההסתברות המשותפת Q_{m_1,m_2} המוגדרת באופן הבא:

$$Q_{m_1,m_2} = \mathfrak{Z}^{-1}\frac{1}{1+\left\|\vec{Y}_{m_1}-\vec{Y}_{m_2}\right\|^2}$$

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

המיפוי בין מרחב המקורי $\mathcal{X}$ 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 $\vec{Y}_{m_i}$:

$$\begin{aligned}\frac{\delta C}{\delta \vec{Y}_{m_i}} &= \frac{\delta}{\delta \vec{Y}_{m_i}}\left[\sum_{m_1} P_{m_1,m_i}\log\left(\frac{P_{m_1,m_i}}{Q_{m_1,m_i}}\right)+\sum_{m_2} P_{m_1,m_2}\log\left(\frac{P_{m_i,m_2}}{Q_{m_i,m_2}}\right)\right]\\ &= 4\sum_{m_1}\left(P_{m_1,m_i}-Q_{m_1,m_i}\right)\left(1+\left\|\vec{Y}_{m_1}-\vec{Y}_{m_i}\right\|^2\right)^{-1}\left(\vec{Y}_{m_1}-\vec{Y}_{m_i}\right)\end{aligned}$$

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 $\alpha(t)$.

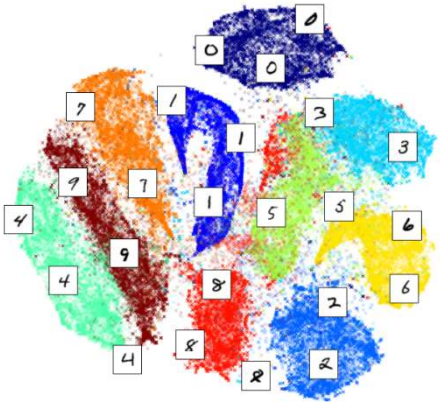
ב. חשב את P_{m_1,m_2}.

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 ההתחלתים 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 $\mathbb{R}^{28\times28}$, 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 (ויזואליזציה) 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 בגווני 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 הגאוסיאנית 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:

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

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 גאוסיאנית עבור שתי מדידות במרחב $\mathcal{Y}$. שיטה זו נקראת *SNE*, והגרדיאנט של פונקציית המחיר במקרה זה נתונה על ידי:

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

ג. אולם, פונקציית דמיון גאוסיאנית במרחב $\mathcal{Y}$ יכולה לגרום לכך שנקודות לא מאוד קרובות במרחב $\mathcal{X}$, ימופו לנקודות קרובות במרחב $\mathcal{Y}$, כיוון שהגאוסיאן 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" או מתקרבות 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.

<u>לשיטת t-SNE יש שלוש מגבלות עיקריות</u>

א. 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.

ב. קללת הממדיות: t-SNE מבוססת על מאפיינים מקומיים בין נקודות. השיטה, המבוססת על מטריקת מרחק אוקלידית, וכך מניחה ליניאריות מקומית על גבי היריעה המתמטית בה מתקיימות הנקודות. אולם, במערך נתונים בו הממד הפנימי גבוה, שיטת t-SNE עלולה להיכשל כיוון שהנחת הליניאריות 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.

ג. 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.

2.3.3 Uniform Manifold Approximation (UMAP)

המודל

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. אלגוריתם bagging הינו שיטה נפוצה ופשוטה יחסית לשיפור ביצועים, אם כי הוא עשוי 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 ליניארית וכו'. כפי שהוסבר לעיל, 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 כקלט ל"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.

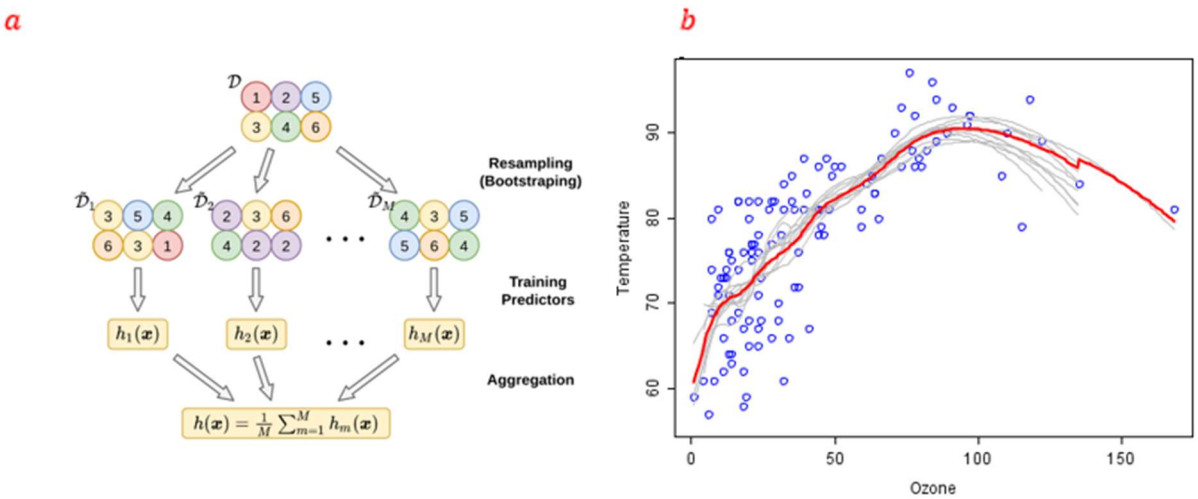
אופן חיבור המודלים המתבצע ב-bagging מבוסס על אותו רעיון של K-NN, כאשר מודלים אלו יכולים לשמש הן למטרות סיווג והן למטרות רגרסיה. כאמור לעיל (בפרק 2.1.3), באלגוריתם השכן הקרוב כל "שכן" העיד על התווית שלו, ולאחר הכרעת הרוב נקבעה התווית של התצפית החדשה. במקרה שבו נספור את תדירות כל התוויות השכנות, התווית 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. כאשר bagging יעבוד כמסווג, כל weak learner יבצע תחזית, והתווית השכיחה ביותר תהיה התוצאה של האנסמבל (-הכרעת הרוב). 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 מיצוע כל המודלים:

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



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) דוגמא 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:

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

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

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

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 אחוד "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 קומבינציה 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:

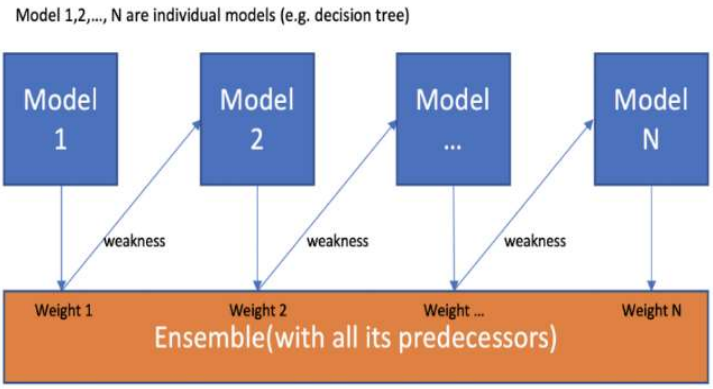
1. אתחול הנתונים: $\mathcal{D}_1=\mathcal{D}$.
2. עבור $t=1,\dots,T$:
   אימון מודל חלש על $\mathcal{D}_t$: $c_t(x)=L(\mathcal{D}_t)$.

חישוב שגיאת המסווג: $\epsilon_t = P_{x\sim D_t}\big(c_t(x) \neq f(x)\big)$.

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 המרכזיות שבהן.



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)

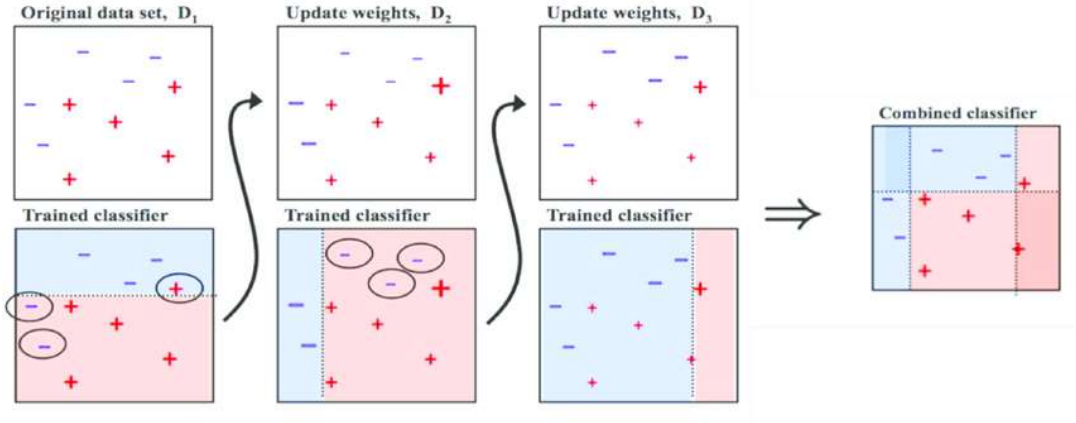
Adaboost היא אחת הטכניקות הראשונות של boosting, ועל אף שקיימות טכניקות boosting נוספות, Adaboost היא בין הפופולריות ביותר בתחום (אם כי יש לה מספר לא מבוטל של וריאנטים). העוצמה הגלומה בטכניקה זו נובעת מכך שגם בהינתן מספר מאפיינים רב, האלגוריתם מצליח להיפגע פחות מ"קללת הממדיות" ולשמור על יכולות ניבוי טובות, 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 לחפיפיות 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 עבוד 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: $w_i^{t+1} = w_i^t \exp\big(-\alpha_t y_i c_t(x_i)\big)$.
   נרמול המשקלים בהתאם לסכומם הכולל: $N_{t+1} = \sum_i w_i^t \to w_i^{t+1} = \frac{w_i^t}{N_{t+1}}$.
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.

להוסיף הוכחה על חסם שגיאת האימון (כתוב כבר, אברהם צריך לערוך).

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.