2. Machine Learning

2.1 Supervised Learning Algorithms

2.1.2 Naive Bayes

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

נניח שיש מודל המקבל וקטור פיצ'רים בינאריים {כאב ראש, משתעל, חום גבוה}, ומסווג האם אדם בעל תכונות אלה חולה בשפעת או לא. באופן כללי ניתן לומר שיש תלות בין שיעול לבין חום גבוה, כלומר העובדה שיש לאדם חום מעלה את ההסתברות שהוא גם משתעל. למרות זאת, ניתן להניח באופן "נאיבי" שאם כבר יודעים שאדם חולה בשפעת, אז כבר אין יותר תלות בין היותו משתעל להיותו בעל חום. באופן פורמלי, אמנם סביר להניח שמתקיים $p(\text{חום}|\text{משתעל}) > p(\text{משתעל})$, 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, הפיצ'רים שלו בלתי תלויים. הנחה זו כמובן לא תמיד מדויקת, וממילא גם ערכי ההסתברויות הנובעים ממנה ומשמשים לסיווג אינם מדויקים, אך ההנחה מקלה מאוד על חישוב ההסתברויות של הסיווג הבייסיאני, ובמקרים רבים תחת ההנחה זו התקבלו תוצאות סיווג. הסיבה להצלחת המודל נעוצה בכך שבבעיית סיווג העיקר הוא למצוא את הסיווג הסביר ביותר לאובייקט (שפעת או לא-שפעת 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 פיצ'רים $x \in \mathbb{R}^n = (x_1, \dots x_n)$, היכול להיות שייך לאחת מ-K קטגוריות $y = (y_1, \dots y_k)$. ההתפלגות הפריורית $p(y_k)$ ידועה, ובנוסף ידועות ההתפלגויות המותנות של הפיצ'רים בהנתן הסיווג – $p(x_i|y_k)$. בעזרת הנתונים האלה רוצים לסווג את x לאחת מהקטגוריות, כלומר למצוא את y_k שעבורו הביטוי $p(y_k|x)$ הוא מקסימלי. באופן פורמלי ניתן לנסח זאת כך:

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

בשביל למצוא את y_k האופטימלי ניתן להיעזר בחוק בייס:

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

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

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

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

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

כעת נשתמש בהנחת הנאיביות, לפי בהינתן הסיווג y_k, אין תלות בין הפיצ'רים. 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, הפיצ'רים קיבלו ערכים בדידים, ולכן היה ניתן לחשב את ההסתברות המותנית של כל פיצ'ר $p(x_i|y_k)$ על ידי ספירת כמות הפעמים שמופיע כל פיצ'ר באוכלוסייה הנדגמת ולחלק בגודל המדגם. עבור ערכים רציפים (כמו למשל מחיר מניה, גובה של אדם וכדו'), אין אפשרות לחשב כך את ההסתברות המותנית. במקרים

כאלה יש להניח התפלגות מסוימת עבור המדגם, ולחשב את הפרמטרים של ההתפלגות שיטות שונות (למשל בעזרת נראות מרבית – MLE). עבור מדגם המתפלג נורמלית, ההסתברות המותנית היא גאוסיאן:

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

כאשר μ_y, σ_y^2 הם הפרמטרים של ההתפלגות, וכאמור הם 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 פיצ'רים היכולים לקבל ערכים רציפים היא לבצע דיסקרטיזציה לערכים אותם הפיצ'רים יכולים לקבל.

במקרה המולטינומי, בו ההתפלגות היא רב מימדית 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 פיצ'רים $x \in \mathbb{R}^n = (x_1, \dots x_n)$, הנראות מחושבת באופן הבא:

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

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

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

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

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

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

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

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:

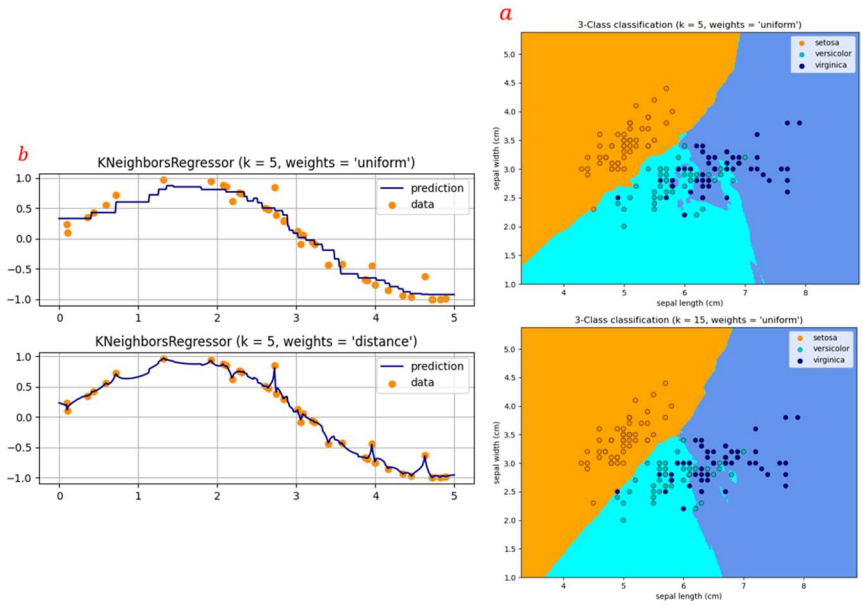
$$\arg\min_k p(y_k)\prod_{i=1}^{n}\frac{1}{p(x_i|y_k)}$$

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 2.1 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 המימד (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:

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

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:

$$y=1 \Leftrightarrow x^Tax + b^Tx + c > 0$$

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:

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

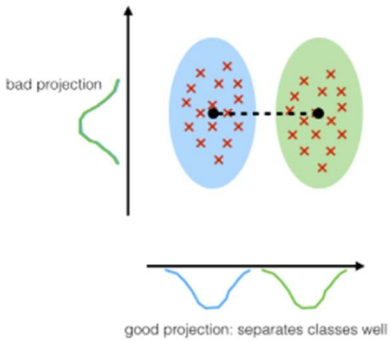
$$b = \frac{1}{2}(\mu_0^T\Sigma^{-1}\mu_0 - \mu_1^T\Sigma^{-1}\mu_1) + \log\frac{p(y=1)}{p(y=0)}$$

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

2.2 Unsupervised Learning Algorithms

2.2.1 K-means

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

סנטרואיד הוא מונח מתחום הגיאומטריה, והוא מתאר את הממוצע האריתמטי של כל הנקודות שמתפרסות על פני צורה כלשהי. באופן אינטואיטיבי ניתן לחשוב על סנטרואיד כנקודת איזון של צורה גיאומטרית 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:

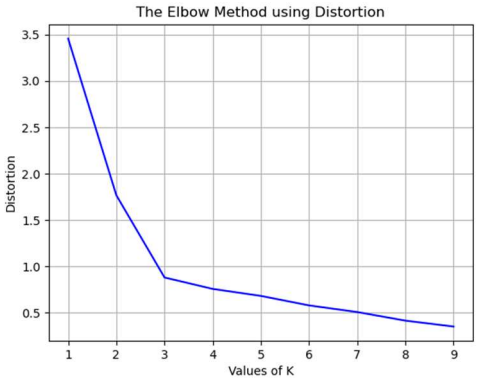
$$WCSS = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

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

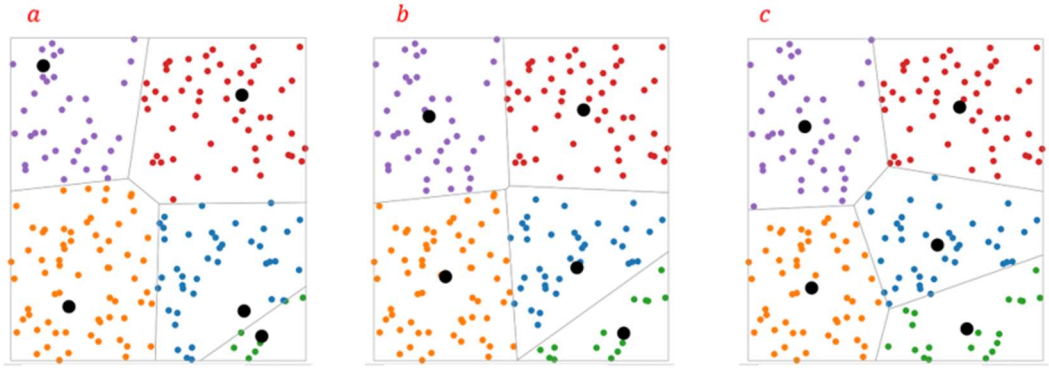
במדד ה- WCSS הוא מתון במידה סבירה. שיטה זו היא היוריסטית ואין דרך חד משמעית לקבוע שה-K הנבחר הוא האופטימלי. בדרך זו ניתן לשכנע מדוע ה-K שנבחר הוא הנכון, אך ההחלטה הסופית נתונה לשיקול דעתו של המשתמש.



איור 2.3 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 2.4 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:

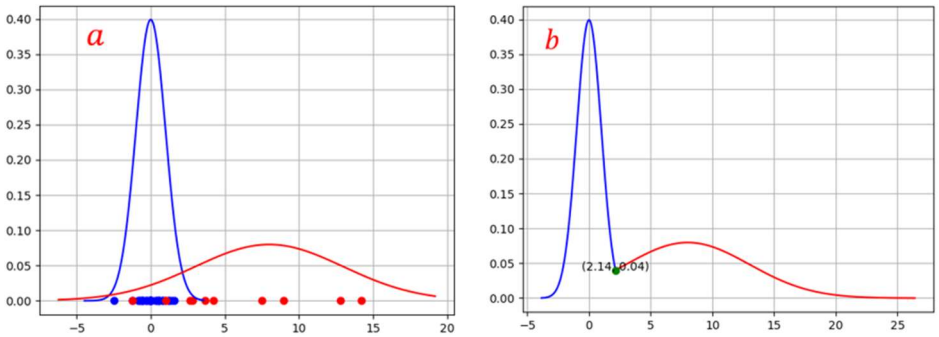
$$p(y=i|x)=\frac{p(y=i)\cdot p(x|y=i)}{p(x)}$$

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:

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

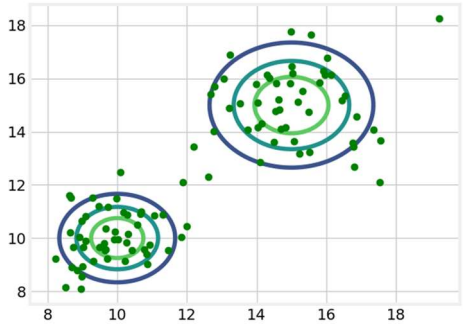


איור 2.5 a) תערובת של שני גאוסיאנים במימד אחד: בשלב ראשון מחלקים את הנקודות לשני אשכולות ומתאימים לכל אשכול התפלגות מסוימת. במקרה זה אשכול אחד (מסומן בכחול) הותאם להתפלגות $\mathcal{N}(0,1)$, ואשכול אחד (מסומן באדום) הותאם להתפלגות $\mathcal{N}(8,5)$. b) נקודה חדשה x תסווג לאשכול הכחול אם $x<2.14$, כיוון 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 מימד, אך הן יכולות להיות גם במימד יותר גבוה. למשל אם מסתכלים על מישור, ניתן להתאים לכל אשכול התפלגות נורמלית דו-ממדית. במקרה ה-n מימדי, 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 2.6 תערובת של שני גאוסיאנים בדו-מימד: 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 סט 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:

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

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

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

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

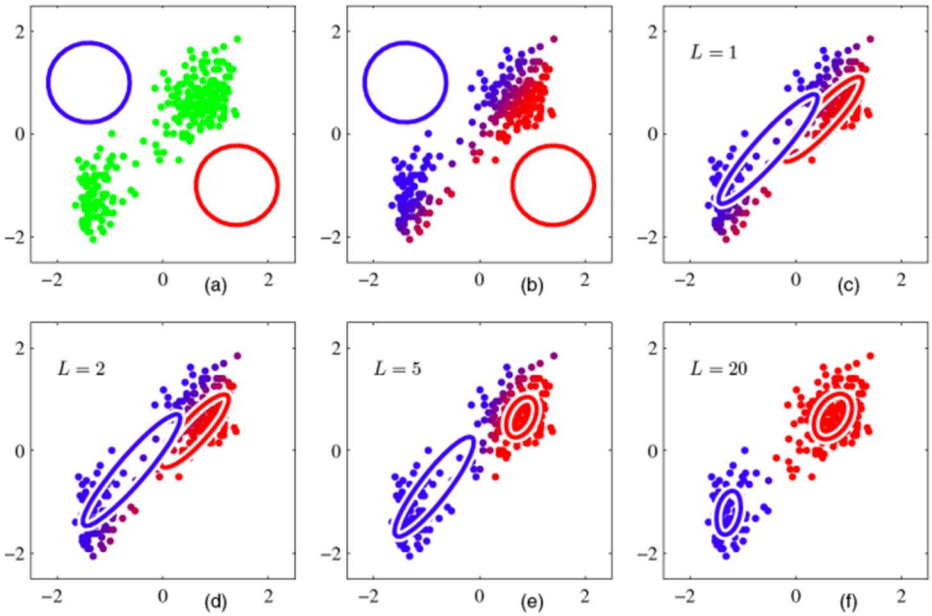
וכעת ניתן לגזור ולמצוא אופטימום:

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

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

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

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



איור 2.7 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: $\min\{d(a,b): a \in A, b \in B\}$.

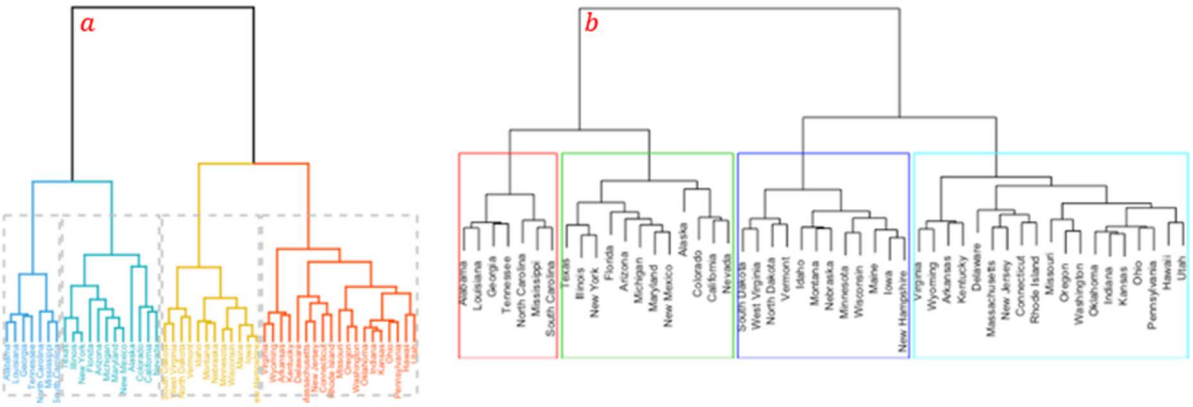
Unweighted average linkage clustering (UPGMA): $\frac{1}{|A|\cdot|B|}\sum_{a\in A}\sum_{b\in B} d(a,b)$.

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 2.8 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.

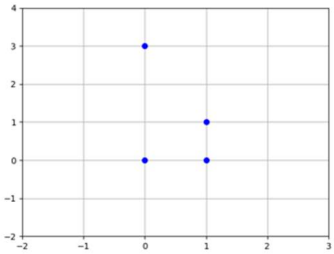
כדי להמחיש את התהליך נסתכל על הסט הבא: $\{A = (0,0), B = (1,0), C = (1,1),\ D = (0,3)\}$, ונקבע $k = 2$.
נחשב את ה-k-distance של כל נקודה במונחים של מרחק מנהטן:

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

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

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

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



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:

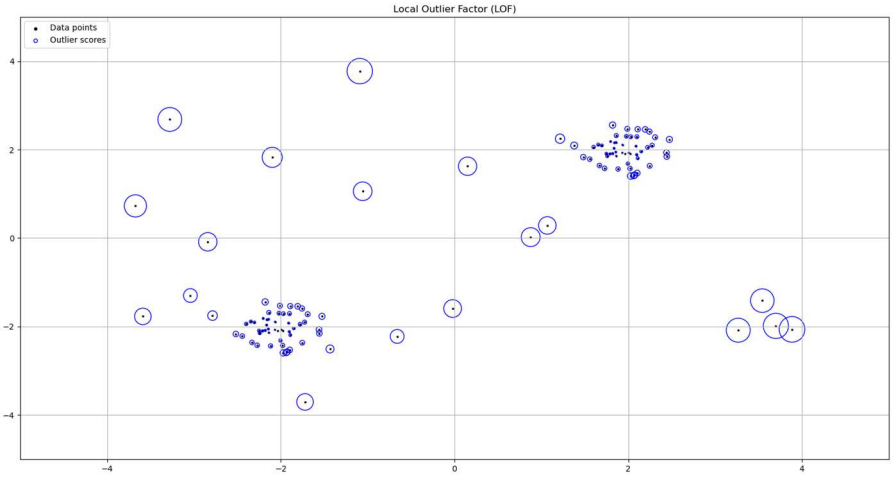
$$LOF_2(A) = \frac{LRD_2(B) + LRD_2(C)}{k} \times \frac{1}{LRD_2(A)} = 0.87$$

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

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

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

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



איור 2.9 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).

References

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