למידת מכונה

מחברת מתוך ההרצאות של פרופ' ליעד גוטליב
אוניברסיטת אריאל, 2022

למידת מכונה - מחברת הרצאות

נערך ונכתב על-ידי דור עזריה

2022

ספר זה לא נבדק על ידי מרצה, יתכן שימצאו טעויות.

לסיכומים נוספים שלי במדעי המחשב ומתמטיקה:
https://dorazaria.github.io/

מוזמנים לעקוב אחרי 😊:
LinkedIn: https://www.linkedin.com/in/dor-azaria/
GitHub: https://github.com/DorAzaria

תוכן עניינים

מבוא ללמידת מכונה

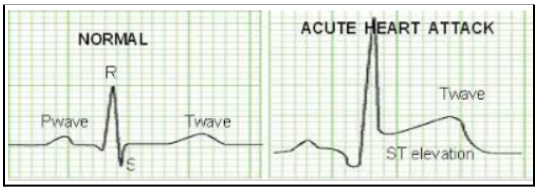
<u>מה זה למידת מכונה?</u>

אנו מבקשים לתכנת מחשבים כך שהם יוכלו "**ללמוד**" מתוך קלט העומד לרשותם.
בגסות, למידה היא **תהליך** של המרת ניסיון למומחיות או ידע.
הקלט של אלגוריתם הלמידה יהיה נתוני אימון (Training Data), ייצוג ניסיון והפלט היא ידיעה כלשהי.
בחיפוש אחר הבנה מתמטית פורמלית של מושג זה, נצטרך להיות מפורשים יותר לגבי מה שאנו מתכוונים לכל אחד מהמונחים המעורבים: מהם נתוני ההכשרה שאליהם יגיעו התוכניות שלנו? כיצד ניתן להפוך תהליך הלמידה לאוטומטי? כיצד נוכל להעריך את הצלחתו של תהליך כזה (כלומר, איכות הפלט של תוכנית למידה)?
בקצרה, למידת מכונה יש מאגר מידע ועל סמך המאגר הזה אפשר יהיה להסיק מסקנות על מידע אחר.

- לדוגמה
  - בהתחשב בגובה של מישהו, אתה יכול לנחש את המשקל?
  - בהתחשב בטמפרטורה ובדופק הלב של מישהו אתה יכול לנחש את המין?
  - זיהוי הודעות ספאם:
    - לשרת המיילים יש מערכי נתונים של מיליוני פריטים שמשתמשים סימנו כספאם

<u>דוגמה נוספת</u>

איך בונים מערכת שמזהה האם בן אדם קיבל התקף לב ב-EKG?



- בעיות בבניית מערכת שכזאת:
  - האם קיימים מומחים?
  - אם קיימים, האם הם מכירים את החוקים?
  - האם הנבדק הוא בן אדם בריא?
- המטרה: מחשב שילמד באופן **אוטומטי** את החוקים ללא שום עזרה אנושית.

<u>דוגמה נוספת</u> - **Netflix Prize**

בן אדם עושה מנוי לנטפליקס, רואה כמות סרטים מסוימת עד שנגמר המנוי שלו.
לנטפליקס היה מערכת שהייתה מציעה לך סרטים על פי הסרטים שראית בעבר.
כלומר, על כל סרט שראית תגיד האם אהבת אותו או לא ולפי זה נמליץ לך על סרטים נוספים.
מערכת מסוג זה נקראת **Recommender System**.
בעצם היה להם מערכת שמטרתה לחזות המלצות סרטים לכל משתמש.
הטכניקה שנטפליקס השתמשה נקראת **PCA - Principal Component Analysis**. (נלמד עליה בהמשך).

סטיית מטבע

בהינתן מטבע, נרצה לדעת מה הסטייה שלו p?
אם המטבע הוגן אז הסטייה שלו היא ½ .
לכל הטלת מטבע .... $x_1, x_2, x_3,$ הוא מתפלג ברנולי $\{0, 1\}$ עם משתנה מקרי $Pr[1] = p,\ Pr[0] = 1 - p$.
אם יש **n** הטלות, אז ערך הממוצע (התוחלת) יהיה:

$$X = \frac{\sum_{i=1}^{n} x_i}{n}$$

נרצה למדוד את הסטייה של המטבע, לכן נטיל מטבע המון פעמים, אבל זה לא אומר לנו ב-100% שנקבל את התוצאה אלא נגיד "**בהסתברות גבוהה, נקבל את התוצאה X**".

חסם הופדינג (Hoeffding's Inequality)

בתורת ההסתברות, חסם הופדינג (Hoeffding), הוא חסם עליון על ההסתברות שסכום של משתנים מקריים יהיה שונה מהתוחלת שלו.

<u>בתכל'ס</u>
יש משתנה ברנולי שההסתברות שלו זה האמת, נרצה לדעת מה זה p, ניקח את n דגימות אקראיות לפי הפונקציה Ber , ה-X הוא הממוצע (התוחלת) ונרצה לדעת מה ההסתברות שמה שדגמנו שונה מהאמת ב-ε מסוים.

<u>הגדרה</u>
יהא Ber ברנולי $\{0, 1\}$ התפלגות עם פרמטר p.
נדגום n פעמים באופן מקרי ואחיד, לפי ההתפלגות Ber, ו-X הממוצע (התוחלת).
אז לכל $0 < \varepsilon < 1$ מתקיים:

$$Pr[\ |X - p| > \varepsilon\ ] < 2 \cdot e^{-2 \cdot n \cdot \varepsilon^2}$$

ברגע שיש לנו אפסילון מספיק גדול אז באופן אקספוננציאלי, ההסתברות יורדת.
בהסתברות גבוהה - קרוב לאמת. בהסתברות יורדת - אנחנו רחוקים מהאמת.

דוגמה

הטלנו מטבע n פעמים, ראינו שהממוצע הוא $X = \frac{\sum_{i=1}^{n} x_i}{n}$

בהינתן לכל $0 < \varepsilon < 1, \quad 0 < \delta < 1$ נרצה לקיים את הטענות הבאות:[1]

- **ההסתברות** שאני קרוב לאמת $1 - \delta$.
- כמה אני רחוק מהאמת (**הסטייה** של המטבע) $X \pm \varepsilon$.

מה גודל המדגם שאנו צריכים?

לפי חסם הופדינג, נציב את ההסתברות שאנחנו טועים להיות קטן או שווה ל-δ כי אחנו רוצים שההסתברות שאנחנו צודקים יהיה לפחות $(1 - \delta)$ כלומר:

$$2 \cdot e^{-2 \cdot n \cdot \epsilon^2} \leq \delta$$

או:

$$n \geq \frac{\ln(\frac{2}{\delta})}{2 \cdot \epsilon^2}$$

הגענו לחסם הזה לפי השלבים הבאים:

$$2 \cdot e^{-2 \cdot n \cdot \epsilon^2} \leq \delta \quad /: 2$$

$$e^{-2 \cdot n \cdot \epsilon^2} \leq \frac{\delta}{2} \quad / \cdot ln$$

$$-2 \cdot n \cdot \epsilon^2 \leq \ln(\frac{\delta}{2})$$

$$n \geq -\frac{\ln(\frac{\delta}{2})}{2 \cdot \epsilon^2}$$

$$n \geq \frac{\ln(\frac{2}{\delta})}{2 \cdot \epsilon^2}$$

חשוב: התוצאה היא **א-סימטרית**.

הרבה יותר קל לקבל הסתברות כאשר (δ נמוך) מאשר לקבל היטל הדוק יותר (ϵ נמוך).

[1] שימוש בדלתא הוא עבור ההסתברות, שימוש באפסילון הוא עבור המרחק.

Memorizer

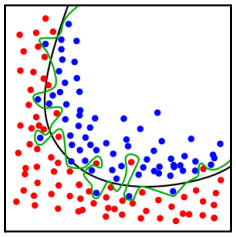
בהינתן מדגם אקראי ו**חוק פשוט**, אז החוק הזה בהסתברות גבוהה תצליח גם על כל העולם.
המשפט הזה אומר שלמידת מכונה הוא דבר אפשרי.

כללים חשובים

1. המדגם חייב להגיע מאותה התפלגות.
2. חוק פשוט (אם יש חוק פשוט שהוא טוב על המדגם שבהסתברות גבוהה הוא יעבוד גם על כל העולם).
3. מצב - שונות הסטייה.

Overfitting

התאמת יתר (overfitting) היא בעיה יסודית בסטטיסטיקה ובלמידת מכונה שבה המודל מותאם יתר על המידה לאוסף מסוים של נתונים (למשל האוסף שהיה זמין לשם אימונו) ועל כן *מצליח פחות בבצוע תחזיות*.
כלומר, הוא מתאים טוב מדי לקבוצת האימון ולכן הביצועים שלו לא טובים על קבוצת המבחן.
התאמת יתר מתרחשת כאשר המודל נקבע על ידי יותר פרמטרים מאשר הנתונים מצדיקים.



הקו הירוק מייצג מודל עם **התאמת יתר**, הקו השחור מייצג מודל **מוסדר** (regularized - מודל שפשטותו מאולצת במפורש). הקו הירוק מתאים יותר לנתוני האימון, אך הוא תלוי בהם יותר מדי ולכן הוא צפוי להיות בעל שגיאה גדולה יותר בסווג נתונים חדשים מאשר המודל השחור.

מטרות הלמידה

- קלסיפיקציה Classification - ערכים דיסקרטיים - הקלטים מחולקים לשתי קבוצות או יותר, ועל הלומד לייצר מודל המקצה קלטים בלתי נראים לאחת או יותר מהקבוצות האלה.
- מולטיקלאס Multiclass - למשל נושאים של אתרים.
- רגרסיה Regression - למשל חיזוי שוק - אתה מאמן את המחשב עם נתוני שוק היסטוריים ומבקש מהמחשב לחזות את המחיר החדש בעתיד.
- דירוג Ranking - (כמו דירוג של מנוע החיפוש של Google).
- מקבוץ Clustering - (בהינתן קבוצה של נתונים עם מאפיינים שונים, אין כאן תיוגים כמו "כן או לא").
- רדוקציה מימדית - PCA.

חזרה על הסתברות

הסתברות מותנית

$$\Pr(A \mid B) = \frac{\Pr(A \cap B)}{\Pr(B)}$$

נוסחת בייס

$$\Pr(A \mid B) = \frac{\Pr(A) \cdot \Pr(B \mid A)}{\Pr(B)}$$

Union Bound

$$\Pr(A \cup B) \leq \Pr(A) + \Pr(B)$$
$$\Pr(A \cap B) \leq \Pr(A) + \Pr(B)$$

אי-תלות

נגיד ש-2 משתנים בלתי-תלויים אם:

$$\Pr(A \cap B) = \Pr(A) \cdot \Pr(B)$$

או:

$$\Pr(A \mid B) = \Pr(A)$$

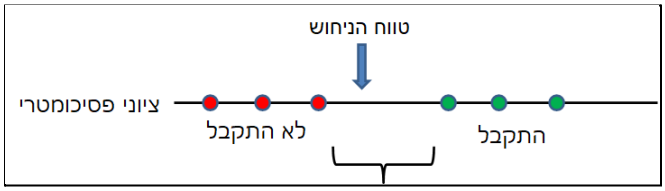
למידת PAC

למידת PAC או **P**robably **A**pproximately **C**orrect זה משפט שאומר כמה אנחנו בערך קרובים לאמת.
למשל בדוגמת המטבעות ראינו כי $1-\delta$ זה ה-"כנראה" והסטייה של המטבע הוא "בערך": $X \pm \varepsilon$.
כלומר נרצה בהסתברות גבוהה להגיד שהאמת הוא בערך "ערך כלשהו". קישור סרטון הסבר.

דוגמה שתלווה את הנושא

נדגום n סטודנטים בהתפלגות אחידה (כלומר, כולם שווי הסתברות $\frac{1}{n}$), כאלה שהתקבלו או לא התקבלו לתוכנית לימודים מסוימת. אנו בנוסף יודעים את ציון הפסיכומטרי של כל אחד מה-n סטודנטים שנדגמו, זה הקריטריה היחידה שידועה לנו על כל סטודנט.
אז בהינתן סטודנט חדש וציון הפסיכומטרי שלו, האם אנו יכולים לנחש שהוא התקבל לתוכנית הלימודים?

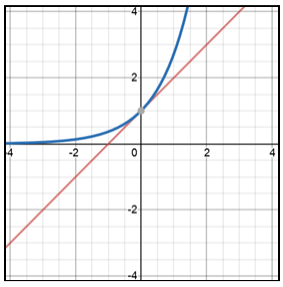
המידע של המדגם נראה כך:



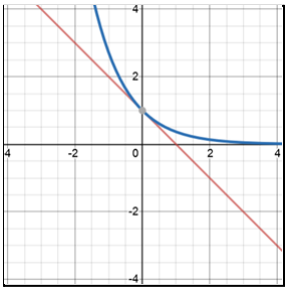
במבט ראשון, יהיה אפשר לדעת שהסטודנט החדש התקבל אם הציון שלו נמצא במינימום של סף המתקבלים.
או שהוא לא התקבל אם הוא נמצא במקסימום הסף של הלא מתקבלים או פחות מזה.
אבל מה קורה אם הציון שלו נמצא בטווח הניחוש? - נציג כמה כלים תיאורטיות כהקדמה לשאלה זו:

לכל x מספר ממשי נרצה להוכיח:

$1 + x \leq e^{x}$

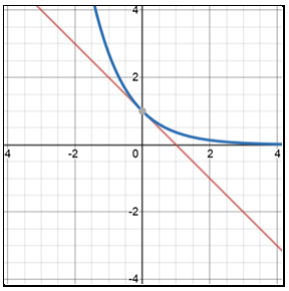


$1 - x \leq e^{-x}$



נשים לב תחילה כי אם $x = 0$ נקבל שוויון - נקרא לזה "חסם הדוק", כלומר קיים מפגש ממש קרוב בין הפונקציות בנקודה 1 על הגרף. - זה בעצם היתרון בשימוש חסמים שכאלה והם מאוד שימושיים.
נוכיח את $1 - x \leq e^{-x}$ ובאופן שקול זה נובע גם עבור $1 + x \leq e^{x}$.

<u>הוכחה</u>



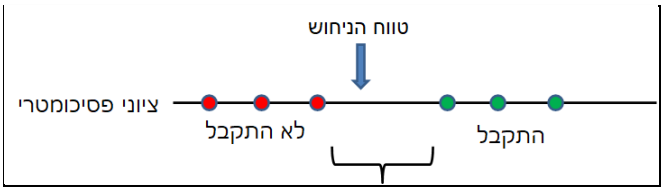
נרצה להוכיח $1 - x \leq e^{-x}$ לכל x ממשי.

יהיו: $f(x) = 1 - x, \quad g(x) = e^{-x}$

והנגזרות שלהם: $f'(x) = -1, \quad g'(x) = -e^{-x}$.

- כאשר $x = 0$ אז נקבל $f(x) = g(x) = 1$.
- כאשר $x < 0$ אז נקבל $f'(x) = -1 > g'(x)$ (כלומר בערכים שליליים, הפונקציה g יותר מאוד מהר. ומכאן - אם הפונקציה g יורדת יותר מהר מ-f והם נפגשים באותו המקום (1) אז זה אומר ש-g הוא גדול מ-f ויורד יותר מהר עד למקום שהם נפגשים.)
- כאשר $x > 0$ אז נקבל $f'(x) = -1 < g'(x)$ (כלומר בערכים חיוביים ה-f קטן הרבה יותר מהר מ-g ולכן גם בחלק הזה g גדול יותר מ-f.)

נחזור לדוגמה שלנו, נתעסק בשאלה - בכמה אחוז מהסטודנטים אנחנו טועים בניחוש שלנו?
כלומר, צריך להוכיח כמה סטודנטים טעינו בניחוש שלנו?



משפט PAC

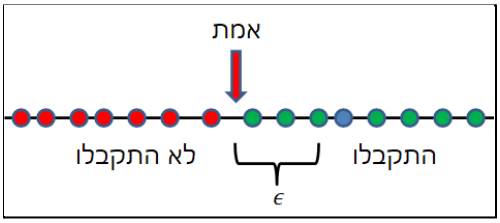
נקבע ε - הקרבה שלנו לאמת וגם δ - כמה אנחנו טועים בהסתברות מסוימת.

בהסתברות גבוהה $1 - \delta$ (כאשר ε, δ הם מספרים קרובים לאפס) נרצה לטעון שאנחנו רק טועים ב-ε מהסטודנטים.

(למשל אם $\varepsilon = 0.01$ אז אנחנו רק טועים ב-1% מהתלמידים).

אנחנו לא נסתכל על המדגם הפעם אלא נעבור אל <u>העולם</u> הלא ידוע.

כל התלמידים שנמצאים **משמאל** לאמת הם בוודאות לא התקבלו.



אך נניח והסף קבלה שלנו הוא העיגול **הכחול**, אז כל הסטודנטים שנמצאים בינו לבין האמת כלומר ε אנחנו נגיד שהם לא התקבלו למרות שהם כן התקבלו, כלומר אנחנו נטעה ב-ε.

אז ההסתברות שדגימה אחת תפספס את אזור הטעות יהיה $1 - \varepsilon$.

ההסתברות ש-n הדגימות יפספסו את אזור הטעות יהיה מאוד קטן, כמה קטן, כזה: $(1 - \varepsilon)^n \leq e^{-\varepsilon \cdot n}$.

- האי שוויון הזה בעצם מתאר שכמה שנקח יותר דגימות ככה יהיה לנו טווח קטן יותר לטעות.

נשאל מה ההסתברות שהמדגם מפספס אחוז מסוים ε מתוך העולם (האמיתי)?

אנחנו נרצה שהטעות תהיה יותר קטנה מ-δ (יותר קטנה מההסתברות לטעות) לכן נרצה:

$$e^{-\varepsilon \cdot n} \leq \delta$$

נציב ln ונקבל:

$$-\varepsilon \cdot n \leq ln(\delta)$$

נחלק ב- $(-\varepsilon)$ ונקבל:

$$n \geq \frac{-ln(\delta)}{\varepsilon} = \frac{\ln(1) - \ln(\delta)}{\varepsilon}$$

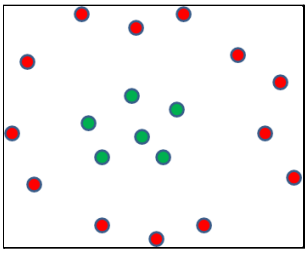
ניתן להציג גם כך:

$$n \geq \frac{ln(\frac{1}{\delta})}{\varepsilon}$$

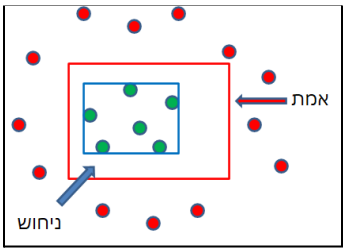
כלומר אנחנו צריכים לפחות n דגימות כדי להבטיח שהטעות תהיה קטנה יותר מ-δ (קטן מההסתברות לטעות).

נקודה חשובה - אנחנו לא יכולים על סמך המדגם לקבוע רטרואקטיבי שאלה ואז להגיד שהמדגם עונה על השאלה הזאת.

דוגמה נוספת



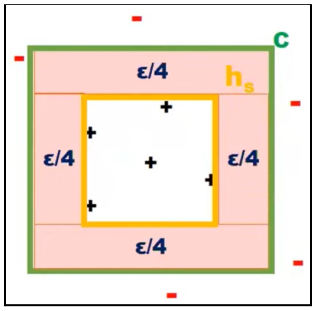
השאלה הקודמת שלנו הייתה בהסתכלות על 1D, נרצה לקחת דוגמה שבנויה ב-2D. לבית החולים הצטרפה אחות חדשה, נולד תינוק והאחות לא יודעת מה החוק (הסף) לגבי טיב הבריאות של תינוקות שנולדו. האחות רוצה לדעת האם צריך לשלוח את התינוק לבדיקות או האם התינוק בריא ושלם. נתונים לנו את הגובה והמשקל של התינוק. כל מה שהאחות יודעת מניסיון של שבוע נמצא בגרף משמאל הראשון:



הניחוש של האחות מוצג בגרף משמאל באופן הבא:

המסגרת האדומה זה האמת (שהאחות לא יודעת אותו) - (העולם האמיתי).

המסגרת הכחולה זה הניחוש של האחות, כלומר היא ראתה כמות תינוקות מסוימת באזור הכחול ולכן אם התינוק נופל באזור הזה הוא לא צריך בדיקה.



כעת מאחר ומדובר ב-2D אנחנו לא יכולים לשאול כמו ב-1D, לכן נחלק את זה ל-4 שאלות, כל שאלה במשקל של $\frac{\varepsilon}{4}$ והמשקל הכללי קטן או שווה ל-ε.

למשל, ההסתברות שדגימה אחת מפספסת את אזור A יהיה:

$$\Pr(A) = (1 - \frac{\varepsilon}{4})^n \leq e^{-\frac{\varepsilon \cdot n}{4}}$$

ולכן ההסתברות שפספסנו בכל האזורים יהיה (לפי Union Bound):

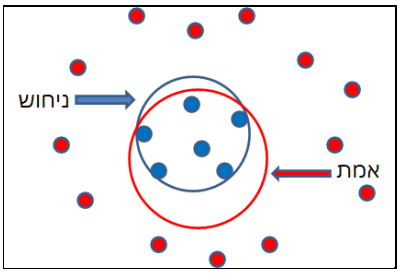
$$\Pr(A, B, C, D) = \Pr(A) + \Pr(B) + \Pr(C) + \Pr(D) \leq 4 \cdot e^{-\frac{\varepsilon \cdot n}{4}}$$

לכן נקבל:

$$4 \cdot e^{-\frac{\varepsilon \cdot n}{4}} \leq \delta \quad \Rightarrow \quad n \geq \frac{4}{\varepsilon} \cdot \ln(\frac{4}{\delta})$$

כלומר, אם אנחנו רוצים שיערוך של ε וודאות של לפחות $1 - \delta$, אנחנו צריכים לבחור n דגימות כך ש:

$$n \geq \frac{4}{\varepsilon} \cdot \ln(\frac{4}{\delta})$$



נקודה חשובה - מה אם נרצה ללמוד בטווח המדגם בצורה של עיגול או מלבן? למשל עבור עיגול, נקח חוק אפשרי כלשהו למשל שימוש ברדיוס על עיגול. אבל איך נדע איך לקבוע את העיגול? - האינטואיציה אומרת לנו ששימוש בעיגול הוא חוק פשוט וטוב לנו. אך האנליזה שעשינו עד כה לא עובדת במקרה כזה, אנחנו צריכים אנליזה יותר מתוחכמת משל זאת.

חסם כללי

אם יש לי מספר סופי של אובייקטים (צורות), העולם של העיגולים הוא אינסופי, לכן אנחנו יכולים לקבוע אינסוף מרכזים ואינסוף רדיוסים.

אם יש לנו מספר **סופי** של צורות (חוקים), אז נוכיח את המשפט הבא:

יהא X אוסף הנקודות, D ההתפלגות של הנקודות X, ו-H אוסף סופי של חוקים[2].

נסמן ב-S דגימה מקרית בה דגמנו n נקודות לפי ההתפלגות D כלומר $n \sim D$.

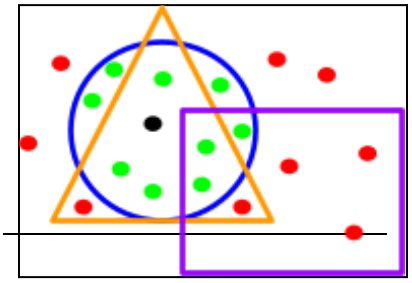
נסמן ב-$e(h)$ את הטעות האמיתית על כל העולם.

ננית וקיים חוק $h \in H$ שהוא לגמרי **עקבי** על המדגם (ללא שגיאות), אז בהסתברות גבוהה הוא דיי טוב על כל העולם. כאשר ההסתברות היא לפחות $1 - \delta$, אז הטעות האמיתי של h על כל העולם[3] יהיה חסום על ידי:

$$e(h) \leq \frac{1}{n} \cdot \left(\ln|H| - \ln \frac{1}{\delta}\right)$$

הסבר פשוט (תכל'ס)

אם יש לנו מספר קבוע של חוקים אפשריים (צורות), אז אם אנחנו דוגמים n מספיק גדול, ויצא לנו שאחד מהחוקים (צורות) האלה הוא עקבי לגמרי על המדגם (לא טעה בכלל) אז בהסתברות $\frac{1}{n} \cdot \left(\ln|H| - \ln \frac{1}{\delta}\right)$ זה חוסם את הטעות שלנו לכל העולם - כלומר הטעות שלנו על כל העולם היא דיי נמוכה. (זה טוב).



למשל בציור משמאל, אפשר לראות כי החוק (הצורה) עיגול עקבי לגמרי על המדגם שלנו (לא טעה בכלל).

לכן, עבור הנקודה השחורה החדשה שנכנסה אפשר יהיה לדעת כי בהסתברות גבוהה אנחנו לא נטעה לגביה.

[2] נניח שב-H לדוגמה יש לנו 3 סוגים יש בו משולש, עיגול ומלבן - הם נקראים **חוקים**.
[3] כל הנקודות ב-X.

הוכחת החסם הכללי

לכל חוק $h_i \in H$, אם הטעות האמיתית שלו על העולם גדול מ-ε כלומר אם $e(h_i) > \varepsilon$,

אז:

$$\Pr\left(h_i \text{ is consistent on all } S\right) \leq (1-\varepsilon)^n$$

(כלומר, אז ההסתברות ש-h_i יצא עקבי על המדגם S יהיה קטן או שווה ל-$(1-\varepsilon)^n$, אז ההסתברות שלא נראה טעויות בכלל על המדגם הוא חסום על ידי $(1-\varepsilon)^n$ שהוא מאוד קטן). אם יש לנו מספר קבוע של חוקים ואז דגמנו אז בהסתברות מאוד גבוהה, הטעות האמיתית של החוק חסום על ידי $\frac{1}{n} \cdot \left(\ln|H| + \ln\frac{1}{\delta}\right)$.

$$\begin{aligned}
&\Pr\left(\exists h \in H:\ e(h) > \varepsilon \cap h \text{ is consistent with } S\right) \overset{4}{=} \\
&= \Pr\Big(\left[h_1 \text{ is consistent with } S \mid e(h_1) > \varepsilon\right] \cup \\
&\cup \left[h_2 \text{ is consistent with } S \mid e(h_2) > \varepsilon\right] \cup \ \ldots\Big) \overset{5}{\leq} \\
&\leq \Sigma_i \Pr\left(e(h_i) > \varepsilon \cap h_i \text{ is consistent with } S\right) \overset{6}{=} \\
&= \Sigma_i \Pr\left(e(h_i) > \varepsilon \mid h_i \text{ is consistent with } S\right) = \\
&= |H| \cdot (1-\varepsilon)^n \leq |H| \cdot e^{-\varepsilon \cdot n}
\end{aligned}$$

ומכאן נקבל:

$$|H| \cdot e^{-\varepsilon \cdot n} \leq \delta \quad \Rightarrow \quad \varepsilon \geq \frac{1}{n} \cdot \left(\ln|H| + \ln\frac{1}{\delta}\right)$$

∎

הוכחנו משפט אם החוק הוא עקבי לגמרי על המדגם, אך מה אם החוק שמצאנו הוא **לא לגמרי עקבי** על המדגם. כלומר יש לו טעות אמפירית $\bar{e}(h)$ על המדגם, אך היינו מצפים שהוא עדיין יהיה טוב.

עבור חוק שלא לגמרי עקבי

יהא X אוסף הנקודות, D ההתפלגות של הנקודות X, ו-H אוסף סופי של חוקים.

נסמן ב-S דגימה מקרית בה דגמנו n נקודות לפי ההתפלגות D כלומר $n \sim D$.

ומצאנו חוק $h \in H$ עם טעות אמפירית (טעות על המדגם) דיי נמוכה $\bar{e}(h)$.

אז בהסתברות גבוהה של לפחות $1-\delta$, הטעות של h על כל העולם $(e(h))$ יהיה חסום ע"י:

$$e(h) \leq \bar{e}(h) + \sqrt{\frac{\ln 2|H| + \ln\frac{1}{\delta}}{2 \cdot n}}$$

- במה החסם הזה שונה ממקודם? - אפילו שאם החוק שלנו לא לגמרי עקבי, הוא עדיין הולך להיות טוב על כל העולם (בהנחה שיש לנו רק מספר קבוע של חוקים).

[4] אנחנו שואלים מה ההסתברות שיש איזשהו חוק בעולם שלנו שהטעות שלו גדולה אבל בכל זאת יצא שהוא טוב על המדגם.

[5] אז או שהחוק הראשון מקיים את זה, או החוק השני או .... כלומר אחד מהחוקים מקיים את ההסתברות הזאת.

[6] נשתמש ב-Union Bound ונחבר את כל החוקים האלה ביחד.

<u>הוכחה</u>

לפי אי שוויון הופדינג, אם יש לנו התפלגות $Ber\ \{0,1\}$ עם פרמטר p.

נדגום n דגימות מתוך המדגם בהתפלגות $n\ \sim\ Ber$, יהא X הממוצע של הדגימות שדגמנו.

אז:

$$\Pr(|X\ -\ p|>\ \varepsilon)\ <\ 2\ \cdot\ e^{-2\cdot n\cdot\varepsilon^2}$$

ולכן נובע ישירות המשפט שהצגנו ולפי הודפינג:

- עבור חוק (צורה) יחידי h נקבל:

$$\Pr(\left|\bar{e}(h)\ -\ e(h)\right|\geq\ \varepsilon)\ \leq\ 2\ \cdot\ e^{-2\cdot n\cdot\varepsilon^2}$$

- עבור אוסף של חוקים H נקבל:

$$\Pr(\exists h\ \in\ H:\ \left|\bar{e}(h)\ -\ e(h)\right|\geq\ \varepsilon)\ =$$

$$=\ \Pr\Big(\ [\ \left|\bar{e}(h_1)\ -\ e(h_1)\right|\geq\ \varepsilon\ ]\ \cup\ [\ \left|\bar{e}(h_2)\ -\ e(h_2)\right|\geq\ \varepsilon\ ]\ \cup\ ....\ \Big)\leq$$

$$\leq\ |H|\ \cdot\ 2\ \cdot\ e^{-2\cdot n\cdot\varepsilon^2}$$

כלומר, ההסתברות שקיים חוק כך שהטעות האמפרית סוטה בהרבה מהטעות האמיתית הוא טעות קטנה.

ולכן נקבל:

$$|H|\ \cdot\ 2\ \cdot\ e^{-2\cdot n\cdot\varepsilon^2}\ \leq\ \delta\quad\Rightarrow\quad\varepsilon\ \geq\sqrt{\frac{\ln 2|H|+\ln\frac{1}{\delta}}{2\cdot n}}$$

חסם צ'רנוף

יהיו $X_1,....,X_n$ משתנים מקריים בלתי תלויים המקבלים ערכים בקטע $[0,\ 1]$ ויהי $X\ =\ \sum_{i=1}^{n}X_i$.

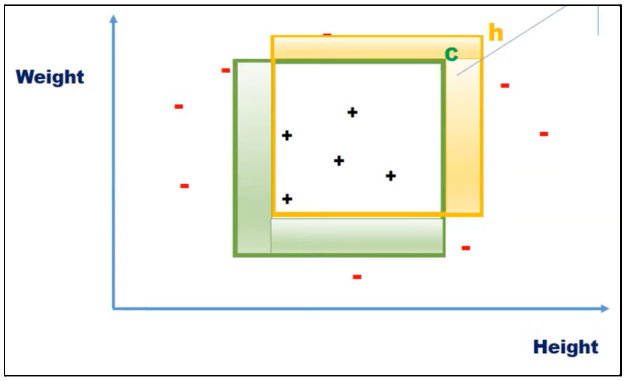
יהא $\delta\ <\ 1$ וגם $\mu\ =\ n\ \cdot\ p\ =\ E(X)$ אז:

$$\Pr(X\ >\ (1\ +\ \delta)\ \cdot\ \mu)\ <\ e^{\frac{-\delta^2\cdot\mu}{3}}$$

- כאשר נרצה להציג <u>חסם עליון</u>, נרצה "לעקוף" את התוחלת נגיד פי 2 או פי מספר מסוים כלשהו.

סיכום כללי על PAC

ה-ε נותן את החסם העליון של השגיאה (בערך) ש-h משוערך בו. (בערך $1 - \varepsilon$)
ה-δ נותן את ההסתברות לשגיאה בהינתן השערוך הזה (נכונות של $1 - \delta$).



ה-**c** הוא טווח הניחוש שלנו (המדגם)
ה-**h** זה טווח האמת (העולם שלא ידוע לנו).
השטח הירוק זה שגיאה שניחשנו לא נכון.
השטח הצהוב זה שגיאה שניחשנו נכון.

ההשערה אומרת שנקבל משהו יחסית נכון
אם $error \leq \varepsilon$.
לכן ההסתברות לשטחים האלה הם
$\Pr(c \oplus h) \leq \varepsilon$.
כלומר $\Pr(Error(h) \leq \varepsilon) > 1 - \delta$.

דוגמה

נניח ונתון לנו $\varepsilon = 0.05,\ \delta = 0.20$.

- נניח שההיפותזה (שיעור) H_1 יוצרת 10 דגימות של שגיאה שונות:

0.04, 0.03, **0.09**, **0.06**, 0.025, 0.049, 0.04, 0.03, 0.05, 0.043

אז אפשר לראות ש-2 מתוך ה-10 דגימות האלה נפסלו (מסומן באדום) זאת כיוון שטווח הטעות שלנו הוא לכל היותר 0.05 כלומר $\varepsilon > 0.09 \ \wedge \ \varepsilon > 0.06$.
לכן, אפשר להגיד כי ההיפותזה צודקת בערך $\Pr(H_1) = \frac{8}{10} = 0.8$.
ומכאן,

$$\Pr(H_1) \geq 1 - \delta$$

כלומר במקרה שלנו,

$$\Pr(H_1) = 0.8 \geq 1 - 0.20$$

ולכן אפשר להגיד כי H_1 היא ככל הנראה **נכונה**.

- נניח שההיפותזה (שיעור) H_2 יוצרת 10 דגימות של שגיאה שונות:

0.04, 0.035, 0.039, **0.06**, 0.025, **0.059**, 0.04, 0.03, **0.055**, 0.043

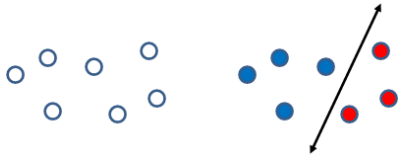
אז אפשר לראות ש-3 מתוך ה-10 דגימות האלה נפסלו (מסומן באדום) זאת כיוון שטווח הטעות שלנו הוא לכל היותר 0.05. כלומר במקרה שלנו,

$$\Pr(H_1) = 0.7 < 1 - 0.20 = 0.8$$

ולכן אפשר להגיד כי H_2 היא ככל הנראה **לא** נכונה.

התנפצות ומימד-VC

נרצה לדעת כמה קווים יש לנו שיפיע כל המדגם.
אנחנו לא נרצה לדעת כמה קווים שונים יש לנו, נרצה לדעת כמה התנהגויות (תיוגים) שונים הקווים יכולים לתת לנקודות.



ייתכן ואם נרצה לתייג את 7 הנקודות האלו אז יש לנו 2^7 אפשרויות תיוג.
אך נשים לב שבדוגמה הבאה לא ניתן לתייג בקו ישר:

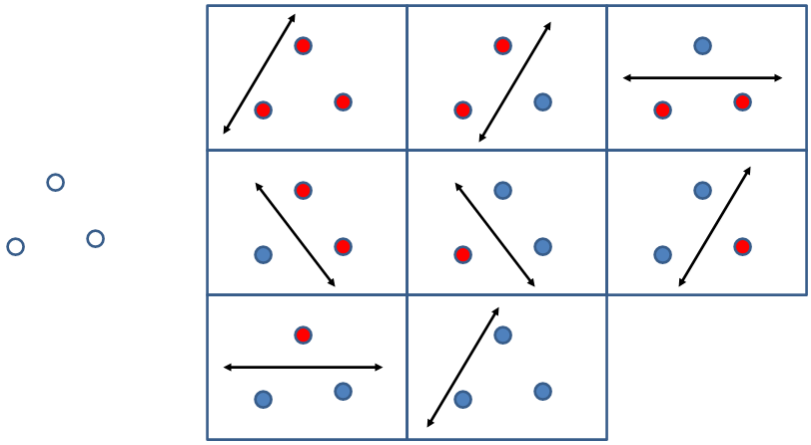
נרצה למצוא את הקווים (התיוגים) שכן יהיה אפשר להציג בקו ישר.

תכונת ה-Shattering

בהינתן קבוצת נקודות P כאשר $|P| = k$ ואוסף של חוקים H נאמר שהקבוצה H מנפצת את P עם 2^k אפשרויות תיוג.

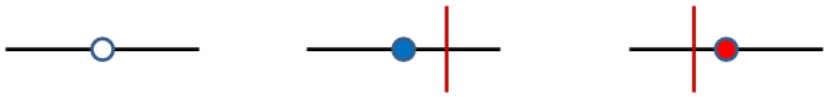
דוגמה:

עבור הנקודות הבאות יש אוסף של חוקים **המנפצים (Shattering)** כל תיוג אפשרי ($2^3 = 8$ תיוגים).



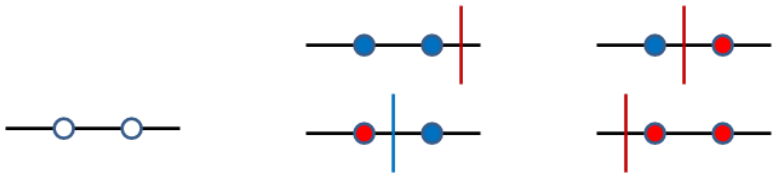
דוגמה:

אוסף **אינסופי** של קווים ניתן לנפץ עם **נקודה אחת**. ($2^1 = 2$ תיוגים).



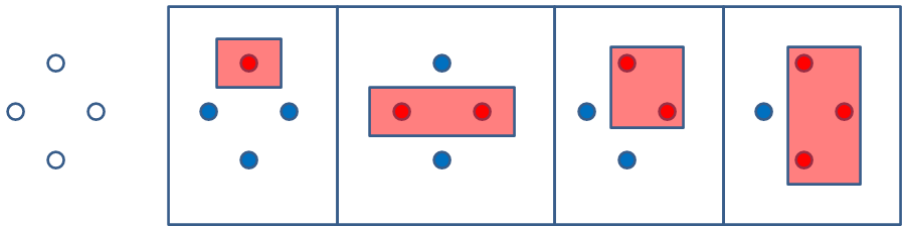
דוגמה:

אוסף **אינסופי** של קווים ניתן לנפץ עם **2 נקודות** ($2^2 = 4$ תיוגים).



דוגמה:

אוסף אינסופי של מלבנים יכול לנפץ אוסף של 4 נקודות ($2^4 = 16$ תיוגים).

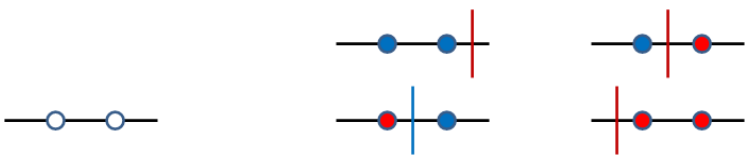


מימד VC

יש אוסף אינסופי של קווים חד-צדדיים שיכול לנפץ נקודה 1 אבל לא קיים אוסף של 2 נקודות שהוא יכול לנפץ.
זה נקרא VC ממימד 1.

כלומר, לא קיים קו חד-צדדי שיכול לנפץ 2 נקודות:

יש אוסף אינסופי של קווים דו-כיווניים שיכול לנפץ 2 נקודות אבל לא קיים אוסף של 3 נקודות שהוא יכול לנפץ.
זה נקרא VC ממימד 2.



טענה

אוסף אינסופי של קווים לא יכול לנפץ 4 נקודות.

כלומר, לא קיים אף אוסף של קווים בכלל שיכול לנפץ 4 נקודות.

- **קיים** אוסף של 3 נקודות שהם יכולים לנפץ.

אך זה לא אומר שהוא יכול לנפץ כל אוסף של 3 נקודות למשל:

מלבנים חד כיוונים

נרצה להוכיח שהמימד-VC שלהם הוא 4.

המלבנים האלה יכולים לנפץ 4 נקודות (כמו בדוגמה בעמוד הקודם).

טענה

מלבנים חד כיוונים לא יכולים לנפץ **אף** אוסף של 5 נקודות.

למשל במבט כללי, אם ניקח את 4 הנקודות המקסימליות והמינימליות לציר x ו-y ונתחום אותם למלבן עדיין נקבל מצב לא אפשרי לתיוג (למשל עם העיגול הכחול):

שאלה - ביחס לאוסף של נקודות כלשהו, כמה סוגים של קווים שונים יש לנו, כלומר כמה תיוגים שונים ניתן לייצג לאוסף.

- ראינו שיש אינסוף קווים, אך אם יש המון קווים שמתנהגים אותו הדבר, אז אין לנו צורך להשתמש בכל אחד מהם.

המשפט של סאואר (Sauer)

בהינתן אוסף S של נקודות בגודל n.

אוסף של חוקים H עם מימד VC של d.

יהי $\Pi(H,S)$ כל התיוגים האפשריים ש-H (החוקים) מתאימים עבור S (הנקודות).

אז:

$$\Pi(H,S) \leq \Sigma_{i=0}^{d} \binom{n}{i} = O(n^d)$$

- בפועל, מספר התיוגים שיש ב-n נקודות- 2^n. אך מספר הקווים האפשריים עבור התיוגים הוא n^d שזה בהרבה יותר קטן מ-2^n.
- המשפט אומר שאם לחוקים יש VC ויש n נקודות אז מספר החוקים שיש לנו "תכלס" הוא n^d אפילו שיש לנו אינסוף חוקים כאלה.

הוכחת המשפט

לפי קומבינטוריקה:

$$\binom{n}{i} = \binom{n-1}{i} + \binom{n-1}{i-1}$$

"אם לא בחרתי את הכדור הראשון אז עדיין נותרו i כדורים ואם כן בחרתי אותו אז נותרו i-1 כדורים"

	x_1	x_2	x_3	x_4	x_5
h_1	0	1	1	0	0
h_2	0	1	1	0	1
h_3	0	1	1	1	0
h_4	1	0	0	1	0
h_5	1	0	0	1	1
h_6	1	1	0	0	1

ניקח את כל החוקים ב-H.

אם שני חוקים שונים רק בנקודה האחרונה, נשבץ אותם ב-2 קבוצות שונות.

מספר התיוגים האפשריים עבור H_1, H_2 על S לא משתנה אם נמחק x_5.

H

	x_1	x_2	x_3	x_4	x_5
h_1	0	1	1	0	0
h_2	0	1	1	0	1
h_3	0	1	1	1	0
h_4	1	0	0	1	0
h_5	1	0	0	1	1
h_6	1	1	0	0	1

H_1

	x_1	x_2	x_3	x_4	x_5
h_1	0	1	1	0	0
h_3	0	1	1	1	0
h_4	1	0	0	1	0
h_6	1	1	0	0	1

H_2

	x_1	x_2	x_3	x_4	x_5
h_2	0	1	1	0	1
h_5	1	0	0	1	1

הגדרות:

$$VC - Dim(H) = d$$
$$VC - Dim(H_1) \leq VC - Dim(H) = d$$
$$VC - Dim(H_2) \leq d - 1$$

" למשל אם אני מחבר את h1 ואת h2 אז אני לא רק מנפץ את x4 אלא אני מנפץ גם את x5 באותו הזמן. "

נוכיח באינדוקציה:

- מקרה בסיס:

עבור $n = d$ נקודות מתקיים $\sum_{i=0}^{d} \binom{d}{i} = 2^d$ הוא פשוט כל הדרכים לתיוג d נקודות שזה 2^d.

- הנחת האינדוקציה:

נניח עבור $n - 1$ נקודות או פחות, וגם מימד ה-VC שזה d או פחות

- צעד האינדוקציה:

נוכיח נכונות עבור n נקודות במימד VC של d.

$$\Pi(H,S) \le \Pi(H_1,S) + \Pi(H_2,S) \le \sum_{i=0}^{d}\binom{n-1}{i} + \sum_{i=0}^{d-1}\binom{n-1}{i} =$$
$$= \sum_{i=0}^{d}\binom{n-1}{i} + \sum_{i=0}^{d}\binom{n-1}{i-1} = \sum_{i=0}^{d}\binom{n}{i}$$

"

1. *ההתנהגויות של H על כל האוסף **חסום** על ההתנהגויות של H1 על כל האוסף וגם של H2 על כל האוסף.*
2. *מספר החוקים השונים ב-H1 לא משתנה אם אנחנו מוחקים את הנקודה האחרונה ולכן (n-1) ואם אנחנו מוחקים אותו לא הקטנו את מספר התיוגים.*
3. *ה-d זה כמה נקודות אנחנו יכולים לנפץ.*
4. *המעבר ל-(i-1) הוא אפשרי כי השלמנו מ-(d-1) ל-(d).*

החסם הכללי בעזרת סאואר

אם ראינו ש:

$$e(h) \le \bar{e}(h) + \sqrt{\frac{\ln 2|H| + \ln\frac{1}{\delta}}{2\cdot n}}$$

לכן עבור אוסף אינסופי של חוקים שונים נשנה את $|H|$ ב- $\left(\frac{e\cdot n}{d}\right)^d$ ונקבל:

$$e(h) \le \bar{e}(h) + \sqrt{\frac{8\cdot d\cdot \ln\frac{2\cdot e\cdot n}{d} + 8\cdot\ln\frac{4}{\delta}}{n}}$$

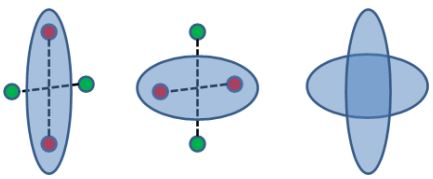
(כאשר ה-8 ו-4 בנוסחה הם פשוט שיקולים מתמטיים שנועדו לדיוק ואין צורך להוכיח אותם).

טענה

מימד ה-VC של אוסף אינסופי של כדורים חד-כיווניים במימד 2D הוא 3.

הוכחה

1. כדורים יכולים לנפץ אוסף של 3 נקודות.
2. כדורים לא יכולים לנפץ אוסף של 4 נקודות.
   a. אם צומת אחת נמצאת בתוך שאר הנקודות (Convex) הוא חייב להיות באותו התיוג.
   b. אם הנקודות נמצאות במיקום כללי, מצאו זוגות מנוגדים. חייב להיות עיגול המכיל כל זוג.



הוכחנו שאוסף אינסופי של קווים/עיגולים לא יכולים לנפץ הרבה נקודות.
נרצה לחפש אוסף של חוקים שכן יהיה אפשר לנפץ הרבה נקודות.
לאורך הקורס אנחנו נלמד על אלגוריתמים שמנסים למצוא חוקים שיהיה אפשר איתם לנפץ את רוב הנקודות.

חוק פשוט

אוסף חוקים ש:

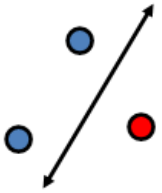
1. ה- $VC - Dimension$ שלהם נמוך
2. הם לא יכולים לנפץ הרבה נקודות (למשל קווים, עיגולים...)

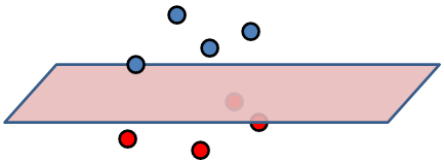
מישורים

- **על-מישור או Hyperplane** הוא מרחב ממימד $n - 1$ בתוך מרחב ממימד n.
- קו נקרא אובייקט ממימד 1, מישור הוא פשוט קו ממימד גדול יותר.
- אם המימוש הוא של d מימדים, אז הוא יכול לנפץ $d + 1$ קודקודים.
- למשל אם הנקודות ב-$d = 2$ מימדים אז המישור שחותך ביניהם יכול לנפץ $d + 1 = 3$ נקודות.
- ואם הנקודות ב-$d = 3$ מימדים אז המישור שחותך ביניהם יכול לנפץ $d + 1 = 4$ נקודות.

טענה

באוסף של היפר-מישורים ממימד $d - 1$ יש $VC - Dimension$ של $d + 1$ על R^d.





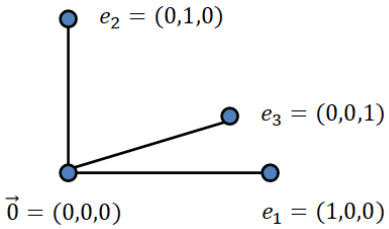
נרצה להוכיח שמישור ממימד d יכול לנפץ $d + 1$.
נרצה להציג חסם עליון שלא אפשרי שמישור ממימד d יכול לנפץ $d + 2$ ויותר.

חסם תחתון

אוסף של מישורים של $d - 1$ מימדים יש $VC - Dimension$ של $d + 1$ על R^d מימדים.

באופן טריוויאלי שמישור יכול לנפץ את הנקודות $\vec{0},\ e_1,\ e_2,\ldots,e_d$

באיור משמאל למשל יש מימד 3 וניפצנו 4 נקודות.



חסם עליון

נוכיח שאוסף של היפר-מישורים של $d - 1$ מימדים לא יכולים לנפץ $d + 2$ נקודות ב- R^d.
נוכיח את זה בעזרת משפט Radon :

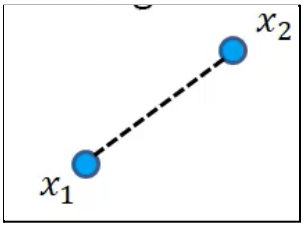
משפט ראדון (Radon)

כל קבוצה של $d + 2$ נקודות ב- R^d (כלומר ב-d מימדים) יכולה להיות מחולקת לשתי קבוצות זרות והקמורות (המעטפת) שלהן מתנגשות (כלומר יש להן נקודת חיתוך).

(כלומר יש לפחות נקודה אחת שנמצאת בתוך הקמור של קבוצה A ושל קבוצה B יחד ולכן יש חיתוך).

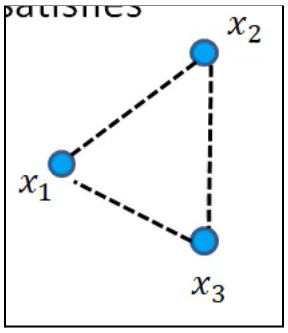
גוף קמור

- בהינתן 2 נקודות x_1, x_2 כל נקודה הנמצאת על הצלע המחברת שלהם היא על: $\alpha_1 x_1 + \alpha_2 x_2$ כאשר $\alpha_1, \alpha_2 \geq 0, \quad \alpha_1 + \alpha_2 = 1$.



- בהינתן 3 נקודות x_1, x_2, x_3 כל נקודה שנמצאת על הצלע המחברת שלהם או בתוך המעטפת עצמה היא על: $\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3$
כאשר $\alpha_1, \alpha_2, \alpha_3 \geq 0, \quad \alpha_1 + \alpha_2 + \alpha_3 = 1$.

" למשל אם $\alpha_1 = 0$ אז הנקודה נמצאת בדיוק על הצלע המחברת בין קודקוד 2 ל-3 ואם כל ה-"אלפות" לא אפס אז הנקודה בתוך המעטפת שלהם. "



- לכן באופן כללי, בהינתן n נקודות $x_1, \ldots, x_n$ כל נקודה שנמצאת על הצלע המחברת שלהם או בתוך המעטפת עצמה היא על: $\Sigma_i \alpha_i x_i$.
כאשר $\Sigma_i \alpha_i = 1, \quad \alpha_i \geq 0$.

הוכחת משפט Radon

יהיו $x_1,....,x_{d+2}$ נקודות.

עבור $\alpha_1,....,\alpha_{d+2}$ (לא אפסים) מתקיים בהתאם ש:

- $\sum_{i=1}^{d+2} \alpha_i x_i = \vec{0}$
- $\sum_{i=1}^{d+2} \alpha_i = 0$

נשבץ ל-2 קבוצות זרות:

אם $\alpha_i > 0$ אז נשבץ את x_i בקבוצה I, אחרת נשבץ אותו בקבוצה J.

הגוף הקמור של קבוצות I, J חייבות "להחתך / להצתלב " ביחד:

$$\Sigma_{x_i \in I} \frac{\alpha_i x_i}{A} = \Sigma_{x_j \in J} \frac{-\alpha_j x_j}{A}$$

כאשר:

$$A = \Sigma_{x_i \in I} \alpha_i = \Sigma_{x_j \in J} -\alpha_j$$

כלומר שקיימת נקודה שנמצאת בגוף הקמור של קבוצה I וגם של קבוצה J יחדיו.

ולכן החסם העליון על ה- $VC - Dimension$ של מישורים הוא $d + 1$.

מבוא לאלגוריתמי למידה

- מישור זה אובייקט שמאוד קל לעבוד איתו לכן הרבה אלגוריתמי למידה מנסים למצוא מישור שהוא טוב על כל המדגם.
- אנחנו רוצים ללמוד עם מישורים, הוא אובייקט פשוט וקל לכתוב לו אלגוריתם.
- מישור הוא פשוט קו במימד יותר גבוהה.
- אם יש לנו אינסוף מישורים, ומצאנו מישור עם טעות אימפרית קטנה $\bar{e}(h)$ אז הטעות האמיתית שלו $e(h)$ חסומה בטעות האימפירית פלוס משהו קטן (נזכר):

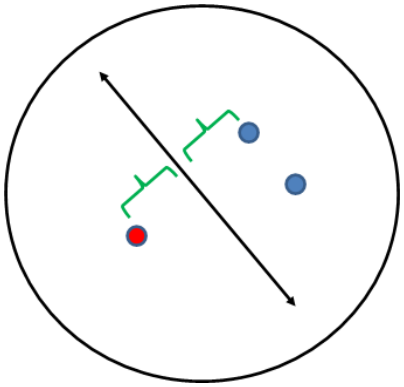
$$e(h) \leq \bar{e}(h) + \sqrt{\frac{8 \cdot d \cdot \ln\frac{2 \cdot e \cdot n}{d} + 8 \cdot \ln\frac{4}{\delta}}{n}}$$

בעיה

אם המדגם ענק אנחנו רוצים חוקים פשוט ולא מסובכים כדי ש-VC לא יהיה גדול מדי.

אם המימד הוא גדול נצטרך מדגם ענק, נציג כעת פתרון לבעיה זו:

מישורים עם מרווחים



אם יש לנו מישור רב מימדי אבל יש לו **מרווח** γ אז הוא יכול לנפץ רק כמה נקודות שהוא $O(R/\gamma)^2$. כאשר R הוא רדיוס.

אם המישור מפריד את הנקודות אז מימד ה-VC תלוי במרווח **ולא** במימד של המישור.
מישור רב מימדי יש לו VC גבוהה אבל מישור שיש לו מרווח יש לו VC שתלוי במרווח לא משנה מה המימד וזה יתרון.

האינטואיציה

ככל שיש יותר נקודות במדגם אז המרווח ביניהם קטן וקטן...
כלומר הם מצטופפים יותר ויותר...
עבור k נקודות המרווח הוא $\frac{1}{\sqrt{k/2}}$.

חיבור של חוקים פשוטים

מה היינו אומרים על חוקים שהם בעצמם חיבור של חוקים פשוטים?
אם יש לנו חוק שהוא חיבור או חיתוך של חוקים פשוטים אז החוק הוא גם פשוט.

משפט

יהיו $H \subseteq h_1, h_2, \ldots$ קבוצה של חוקים פשוטים עם $VC - Dimension$ של d.
יהי $H' = \left\{ \bigcup_{i=1}^{S} h_i \right\}$ איחוד של החוקים הפשוטים.
אז המימד VC של H' חסום בצורה הבאה:

$$VC_{dim}(H') \leq 2 \cdot d \cdot S \cdot \log_2(3 \cdot S)$$

הוכחה

(אפשר להוכיח את זה גם על חיתוך).
לכל קבוצת נקודות P בגודל n מתקיים:

$$\Pi(H', P) \leq [\Pi(H, P)]^S \leq \left(\frac{e \cdot n}{d}\right)^{d \cdot s}$$

ניקח את P להיות האוסף הכי גדול ש-H' יכול לנפץ, אז:

$$2^n < \left(\frac{e \cdot n}{d}\right)^{d \cdot s}$$

ניקח:

$$n \geq 2 \cdot d \cdot s \cdot \log_2(3 \cdot s)$$

ואז:

$$2^{2 \cdot d \cdot s \cdot \log_2(3 \cdot s)} < \left(\frac{e \cdot 2 \cdot d \cdot s \cdot \log_2(3 \cdot s)}{d}\right)^{ds}$$
$$\frac{9 \cdot s}{2 \cdot e} < \log_2(3 \cdot s)$$

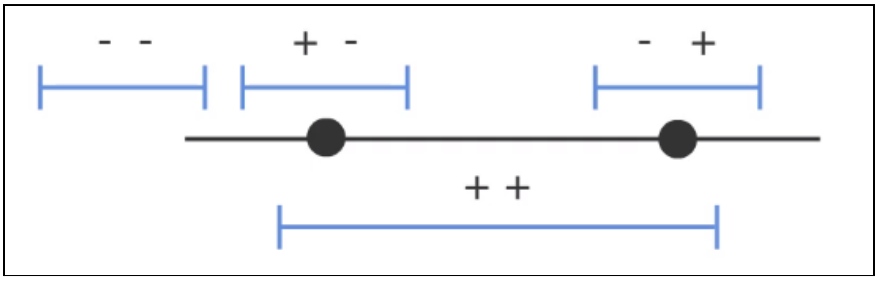
לא מתקיים עבור $s \geq 2$.

∎

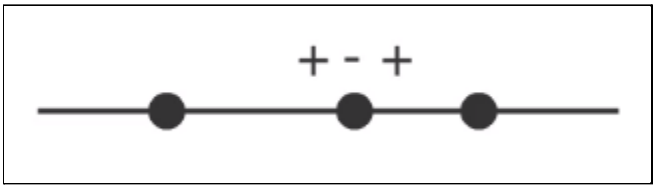
העניין הקריטי פה לגבי איחוד הוא לפי $\Pi(H', P) \leq [\Pi(H, P)]^s$.

הסבר בתכל'ס על VC

אנחנו יכולים לקשר את ממד ה-VC של מחלקה לכמות הנתונים שצריך כדי להיות מסוגל ללמוד ביעילות באותה מחלקה. אז כל עוד הממדיות הזו היא סופית, גם אם מחלקת ההשערה היא אינסופית, נוכל לומר דברים על כמה נתונים אנחנו צריכים ללמוד.
לדוגמה, בהינתן קבוצה בעלת שתי הנקודות על הקו, נקודות במרווח יקבלו את התווית "+".



במקרה זה של 2 נקודות, הקבוצה C בעלת 4 קווים $\{+-,\ --,\ ++,\ -+\}$ והקבוצה של 2 הנקודות היא הקבוצה A בעלת $\{+-,\ -+\}$.
כעת נרחיב את הקבוצה ל-3 נקודות, אז בקבוצה של 3 נקודות קיים "+-+".



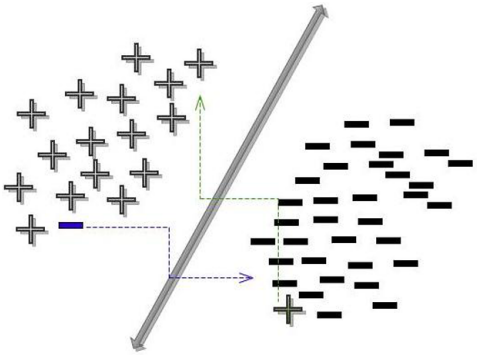
במקרה כזה, אנחנו לא יכולים להפריד את הקבוצה עם קווים.
VC-dimension הוא הביטוי של מרווחים מקסימליים להפרדת קבוצות. והיא הגודל הגדול ביותר של קבוצה שיכולה להתנפץ על ידי H. כאן מוזכר VC-dimension, ויש לו בעיה במרחב ממדי גבוה. כדי להתמודד עם זה, משתמשים מרווחים כדי להגביל את הגבול העליון. באמצעות זה, אנו יכולים להסביר בבירור את המודל הליניארי עם החסם הכללי.

אלגוריתמי למידה של מסווג לינארי

זה פרק שלא הוצג בהרצאות אבל יכול לחזק את ההבנה של האלגוריתמים שנלמד בקורס.

האלגוריתמים הבאים שנלמד מתעסקים בלמידה מסווגת לינארית:

1. אלגוריתם Winnow
2. אלגוריתם Perceptron



מבוא

בלמידת מכונה, מודל Online הוא מודל הפועל כפי שנציג בדוגמה הבאה:

- רובוט מסווג תפוזים לאיכות טובה או לא טובה.
- לאחר כל סיווג הוא מקבל משוב ממומחה.
- הרובוט יכול לעדכן את הפונקציה שלו, ואז לעבור לתפוז הבא.

תיאור המודל הכללי שלנו לפי שלבים:

1. האלגוריתם מקבל דגימה x.
2. האלגוריתם נותן תחזית לסיווג של x - זה נקרא השערה נוכחית או $current\ hypothesis$.
3. האלגוריתם מקבל את הסיווג הנכון - זוהי פונקציית המטרה.
4. האלגוריתם ממשיך לדגימה הבאה.

מה זה למידה של מסווג לינארי?

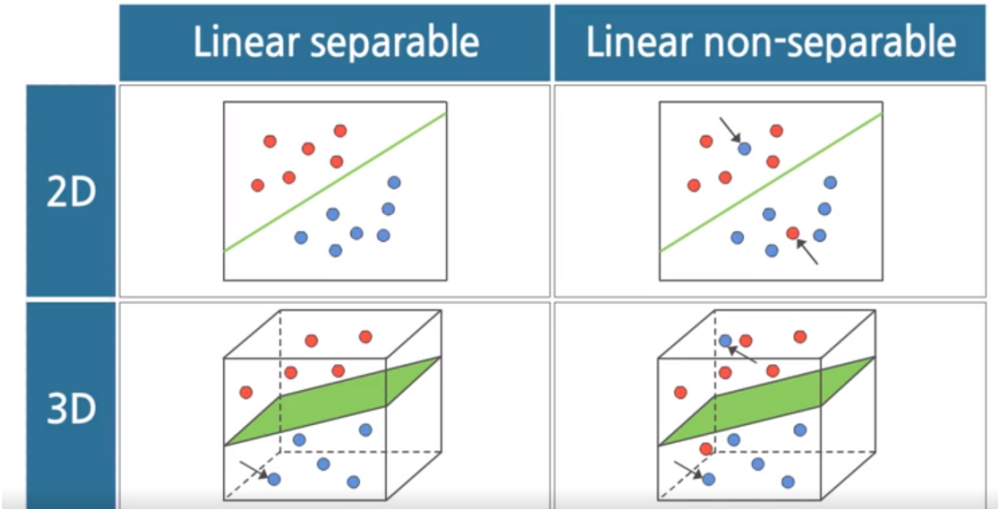
יש אוסף של נקודות חיוביות ואוסף של נקודות שליליות ב-R^n (או $\{0,1\}^n$).

אנחנו רוצים למצוא וקטור w וחסם w_0 כך ש- $wx = w_0$.

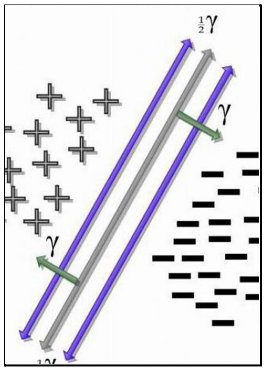
האלגוריתם שלנו יהיה $wx > 0$ גורר סיווג חיובי או $wx < 0$ גורר סיווג שלילי.

בעיית ההפרדה הלינארית

בהתאם להתפלגות בנתונים, צריך לבחור בשיטת הלמידה השונה כדי למצוא את ההשערה הליניארית הטובה ביותר. וגם לקבוע אם הבעיה ניתנת להפרדה ליניארית או לא.

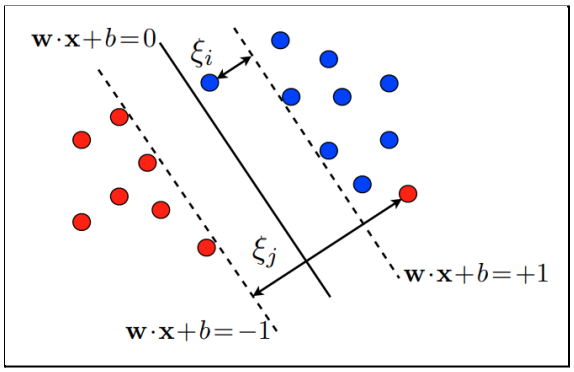


אם מערך הנתונים ניתן להפרדה ליניארית, מישור חוצה אחד ($Hyperplane$) יכול לסווג בצורה מושלמת את התווית ($Labels$) בעצמו.
אבל אם לא ניתן להפרדה ליניארית, שום מישור חוצה ליניארי לא יכול לסווג לחלוטין שני מערכי נתונים.



במקרה של מערך נתונים הניתן להפרדה ליניארית, ישנם מועמדים רבים לבחור ב-Hyperplane. - אבל איזה מהם הוא הטוב ביותר?
אחד הקריטריונים הנפוצים לבחירת הטוב ביותר הוא מרווח שנסמן כ-γ.
המרווח הוא קריטריון חשוב לבחירת היפר-מישור מתאים.
המרווח מוגדר כמרחק הגיאומטרי מהמישור המפריד לנקודות הנתונים הקרובות ביותר.

יש מקרים שאי אפשר לבצע הפרדה לינארית גם עם מרווח טוב למשל כמו באיור מלמטה:



אלגוריתם Winnow

אלגוריתם Winnow הוא אלגוריתם מסווג לינארי.
מטרת האלגוריתם היא להוריד את גודל המדגם מ-$O(n)$ ל-$O(r\log(n))$ ופותר זאת על ידי מפריד לינארי במרחב $\{0,1\}^n$. יש לנו וקטורים עם המון מימדים אבל רוב המימדים לא משמעותיים, אנו לא יודעים מה חשוב ומה לא אך אנו יודעים שמתוך כל ה-n האלה רק r מהם חשובים.
אנו יודעים שבתוך הפיצ'רים החשובים האלה אנחנו רק צריכים '1' באחד מהם כך שהוא יהיה '+' אחרת '-'.

נתבונן בטבלה הבאה כדוגמה:
יש לנו אנשים X_1, X_2, X_3, X_4.
יש לנו פיצ'רים F_1, F_2, F_3, F_4.
למשל הפיצ'ר F_1 הוא פיצ'ר שמגדיר האם האיש ה-X_i הוא זקן או לא.
אם לאדם ה-X יש '1' באחד מהפיצ'רים החשובים F אז התיוג Coffee מקבל עבורו '+' ואחרת הוא מקבל '-'.

	F_1	F_2	F_3	F_4	label
Person	old	male	tall	resident	Coffee
X_1	1	0	1	0	+
X_2	0	1	0	1	+
X_3	1	1	0	0	-
X_4	0	1	0	0	-

אם נתבונן בטבלה אפשר להגיד שהפיצ'רים החשובים שלנו הם F_3, F_4 כי אפשר לראות את ההשפעה של ה-'1' על כל אחד מהפיצ'רים עבור התיוג Coffee.

נסמן **n** מספר כל הפיצ'רים F ו-**r** מספר הפיצ'רים החשובים וגם $F_j(X_i)$ הוא הפיצ'ר F_j עבור הנקודה X_i.

האלגוריתם

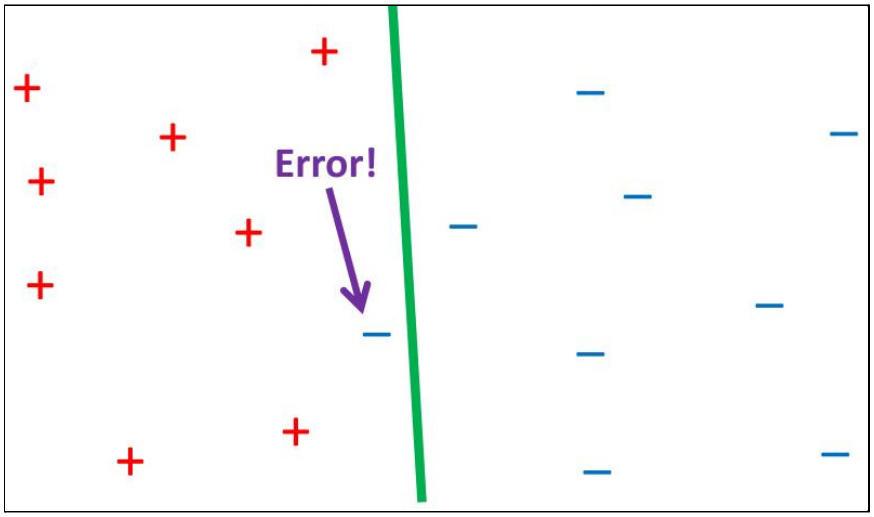
1. נתונה קבוצה של משקלים $w_1,\ldots,w_n = 1$ (כל אחד מהם הוא שווה ל-1).
2. לכל וקטור X_i נקיים:
   - אם $\sum_{j=1}^{n} w_j \cdot F_j(X_i) \geq n$ אז ננחש '+' (עבור קפה למשל מהדוגמה למעלה).
   - אחרת, ננחש '-' (עבור קפה למשל מהדוגמה למעלה).
   - אם עשינו טעות עבור X_i אז נתקן:
     A. אם ננחש '+' אבל '-' הוא הנכון אז לכל j כך ש: $F_j(X_i) = 1$ אז נתקן $w_j = 0$.
     B. אם ננחש '-' אבל '+' הוא הנכון אז לכל j כך ש: $F_j(X_i) = 1$ אז נתקן $w_j = 2w_j$.
3. נטייל על כל נקודה עד שלא נמצא עוד טעויות.

טענה

מספר השגיאות שאלגוריתם Winnow עושה הוא לכל היותר $O(2r\log_2(n))$.

נעזר בסעיפים A,B מהעמוד הקודם של האלגוריתם:

- טעות (B) יכול לקרות לכל היותר $O(r\log(n))$ פעמים.
  - אחרי $r\log(n)$ פעמים, כל משקל חשוב w_j יהיה בעל המשקל n.
- טעות (A) יכול לקרות לכל היותר $O(r\log(n))$ פעמים.
  - אחרי טעות מסוג A, אז $\sum_{j=1}^{n} w_j$ גדל לפחות ל-n.
  - טעות (B) מוסיפה ל-$\sum_{j=1}^{n} w_j$ פחות מ-n, המשקל הכולל יהיה $nr\log(n)$.
  - מספר הטעויות מסוג A הוא לכל היותר $O(r\log(n))$.



אלגוריתם Perceptron

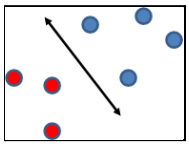
הרעיון

אלגוריתם Perceptron הוא אלגוריתם מסווג לינארי.
תפקידו לאמן את המישור המסווג/המפריד (Hyperplane) וגם לעדכן אותו לפי החוקיות של האלגוריתם.
הרעיון המרכזי העומד בבסיס האלגוריתם הוא שאם אין שגיאה, אין צורך לעדכן את ההשערה הנוכחית שלנו, ואם יש שגיאה, אז נעדכן את ההשערה הנוכחית בכיוון השגיאה.
האלגוריתם יודע למצוא מישור טוב עם מרווח עבור אוסף של נקודות.

מבוא

למדנו שמישורים שיש להם מרווח γ אז יש להם $VC\ Dimension$ שתלוי במרווח ולא במימד כלומר: $\frac{O(R)}{\gamma^2}$.

איך אפשר למצוא מישור טוב בשביל אוסף של נקודות?



1. אפשר לעשות Brute Force (כלומר כל האפשרויות).
   כלומר ננסה את כל האפשרויות של 3 נקודות מתוך ה-n נקודות של S.
   נבדוק שאין שום נקודה שנופלת בתוך המרווח.
   לכן $O(n^4) \rightarrow O(n^{d+2})$.
2. יש אלגוריתם יותר יעיל למצוא מישור טוב עם מרווח - אלגוריתם Perceptron.

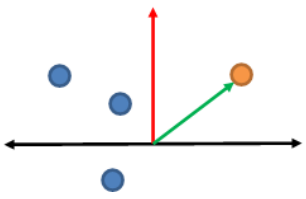
מכפלה פנימית

- מכפלה פנימית של וקטורים שקול ל-Cos.
- וקטורים אורתוגונליים בעלי מכפלה פנימית של 0.
- מכפלה פנימית של וקטור עם עצמו הוא 1.
- למשל הוקטורים הבאים מאונכים לגמרי אחד מהשני והמכפלה הפנימית ביניהם הוא 0:

$(1/\sqrt{2}, 1/\sqrt{2})$

$(1/\sqrt{2}, -1/\sqrt{2})$

- נתבונן בגרף הבא:



הוקטור האדום יכול לתאר לנו במידה מסוימת את המישור השחור כי אם ניקח את הנקודה הכתומה של הוקטור הירוק אז ההטלה של הוקטור הירוק על הוקטור האדום הוא בדיוק המרחק של הנקודה הכתומה מהמישור השחור.

- כלומר, בהינתן כל מישור אנחנו יכולים לקחת את הוקטור שמונח עליו עם הנורמה 1 וזה יכול לייצג לי את המישור המקורי (השחור), כלומר ניקח כל נקודה (שהיא בעצם וקטור) ונעשה מכפלה פנימית עם הוקטור (האדום) ויצא לנו בדיוק את המרחק מהמישור המקורי (השחור).
- היחס בין מכפלה פנימית למישור המפריד הוא יחס ישיר וגם בין הגודל של המכפלה הפנימית למרחק של הנקודה מהמישור המפריד.
- אלגוריתם Winnow למשל מוצא וקטור שמייצג מישור שמפריד בין כל הנקודות, הוא לא מתייחס למרווח אבל הוא כן מצא וקטור מפריד כלשהו.

האלגוריתם

1. נגדיר וקטור $w_1 = \vec{0}$ אורתוגונלי לוקטור $w^{\#}$.
2. עבור $t = 1, 2, \ldots$ סיבובים:
3. נטייל על כל הנקודות x_i:
4. אם $w_t \cdot x_i > 0$ אז ננחש '+'
5. אחרת, ננחש '-'
6. אם טעינו:
7. אם x הוא באמת '+' וניחשתי '-' אז $w_{t+1} \leftarrow w_t + x_i$.[7]
8. אם x הוא באמת '-' וניחשתי '+' אז $w_{t+1} \leftarrow w_t - x_i$.[8]
9. נסיים את הסיבוב ה-t.
10. אם לא קרו טעויות בסיבוב הזה, נסיים את האלגוריתם.

טענה

אלגוריתם $Perceptron$ ימצא מישור מפריד $(Hyperplane)$ עם S ומספר הסבבים שלו חסום על ידי $\frac{1}{\gamma^2}$ כאשר γ הוא המרווח.

- עד עכשיו ראינו ש γ^2 חוסם לנו את ה- $VC\ Dimension$ אפשר להוכיח ש-$\frac{1}{\gamma^2}$ חוסם את מספר הסבבים ולכן בכל סיבוב אנחנו רואים לכל היותר n נקודות ולכן מספר הנקודות שנראה עד שנסיים את האלגוריתם זה $\frac{n}{\gamma^2}$, כלומר הוא מסיים דיי מהר.

[7] כלומר הוקטור היה יותר מדי קטן לכן נוסיף למשקל את הנקודה שטעינו וזה כי הגדרנו שהוא יהיה '+' אם $wx>0$.

[8] כלומר הוקטור היה יותר מדי גדול לכן נוריד מהמשקל את הנקודה שטעינו וזה כי הגדרנו שהוא יהיה '-' אם $wx<0$.

הוכחה

יהי $w^{\#}$ להיות הפתרון האמיתי[9], הוא הוקטור שמייצג את המישור שמפריד את הנקודות S עם מרווח הכי גדול γ.
נציין כי w_t לא מנורמל.

1. אנחנו טוענים ש:[10]

$$w_{t+1} \cdot w^{\#} \geq w_t \cdot w^{\#} + \gamma$$

זה אומר לנו בגדול שהוקטור של המשקל מתקרב לכיוון של $w^{\#}$.

אם הטעות של x היא '+' אז:[11]

$$w_{t+1} \cdot w^{\#} = (w_t + x) \cdot w^{\#} = w_t \cdot w^{\#} + x \cdot w^{\#} \geq w_t \cdot w^{\#} + \gamma$$

אם הטעות של x היא '-' אז:

$$w_{t+1} \cdot w^{\#} = (w_t - x) \cdot w^{\#} = w_t \cdot w^{\#} - x \cdot w^{\#} \geq w_t \cdot w^{\#} + \gamma$$

- כלומר בין אם הנקודה היא שלילית או חיובית, המכפלה הפנימית גדלה בלפחות γ ולכן המשפט הזה נכון כי איכשהו הוקטור שלנו מתקרב יותר ויותר לכיוון האופטימלי למרות שהוא לא מנורמל.

2. אנו טוענים ש: $\left||w_{t+1}|\right|^2 \leq \left||w_t|\right|^2 + 1$.

זה אומר שהנורמה בריבוע לא גדל יותר מדי מהר, אנחנו נרצה לטעון שהוא גדל בכל סיבוב לכל היותר ב-1.

אם הטעות של x היא '+' אז:

$$\left||w_{t+1}|\right|^2 = \left||w_t + x|\right|^2 = \left||w_t|\right|^2 + 2w_t \cdot x + ||x||^2 =$$
$$= \left||w_t|\right|^2 + 2w_t \cdot x + 1 < \left||w_t|\right|^2 + 1$$

אם הטעות של x היא '-' אז:

$$\left||w_{t+1}|\right|^2 = \left||w_t - x|\right|^2 = \left||w_t|\right|^2 - 2w_t \cdot x + ||x||^2 =$$
$$= \left||w_t|\right|^2 - 2w_t \cdot x + 1 < \left||w_t|\right|^2 + 1$$

[9] זה הפתרון שאנחנו מחפשים, לא נגיע ממש אליו אבל נתקרב אליו, הוא הוקטור המנורמל שמייצג את המרווח הכי גדול.
[10] בכל סיבוב, המכפלה הפנימית של הוקטור של המשקל עם הוקטור שמייצג את המישור האופטימלי גדל בלפחות גמא כל פעם. נגיד שהיו לנו 2 וקטורים מנורמלים, אנו יודעים שאם הם מאונכים לגמרי אז המכפלה הפנימית היא 0 ואם זה מכפלה פנימית של אותו וקטור אז המכפלה הפנימית הוא 1, אז אם יש לנו 2 וקטורים שהמכפלה הפנימית שלהם גדולה אז הם דיי באותו הכיוון וזה למה שהמשפט הזה חשוב - איכשהו המכפלה הפנימית גדלה וגדלה בכל פעם.
[11] למרווח של הוקטור האופטימלי יש מרווח גמא וזה אומר שכל הוקטורים שהם '+' אין להם מרחק לפחות גמא מהוקטור האופטימלי כי המכפלה הפנימית שלהם עם הוקטור האופטימלי הוא לפחות גמא כי יש למישור האופטימלי מרווח גמא.

אז לפי טענה (1) הוכחנו שאחרי M סיבובים, בכל סיבוב המכפלה הפנימית גדלה בלפחות γ ולכן:

$$w_{M+1} \cdot w^{\#} \geq M \cdot \gamma$$

ולפי טענה (2) הוכחנו שאחרי M סיבובים, בכל סיבוב הנורמה בריבוע גדל לכל היותר ב-1 ולכן:

$$\left\|w_{M+1}\right\|^2 \leq M$$

$$\left\|w_{M+1}\right\| \leq \sqrt{M}$$

יש לנו:

$$\left(\frac{w_{M+1}}{\left\|w_{M+1}\right\|}\right) \cdot w^{\#} \leq 1$$

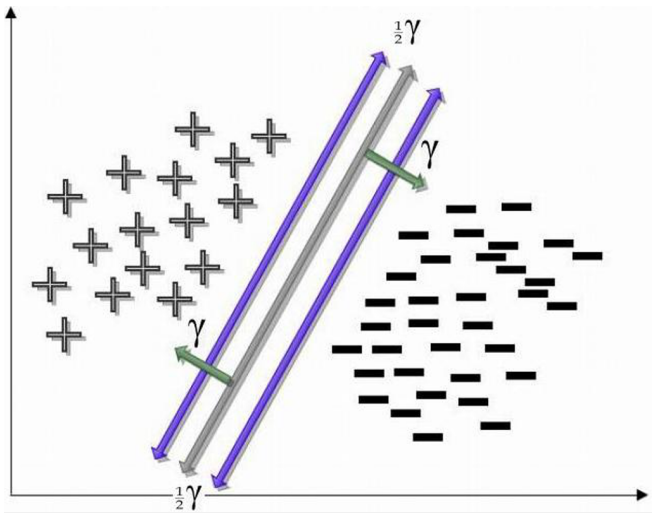
$$w_{M+1} \cdot w^{\#} \leq \left\|w_{M+1}\right\|$$

$$M\gamma \leq \sqrt{M}$$

$$M \leq \frac{1}{\gamma^2}$$

כלומר, מספר הסבבים שיכולים לי להיות כאשר בכל סיבוב אני עושה טעות יכול להיות לכל היותר $\frac{1}{\gamma^2}$.

∎



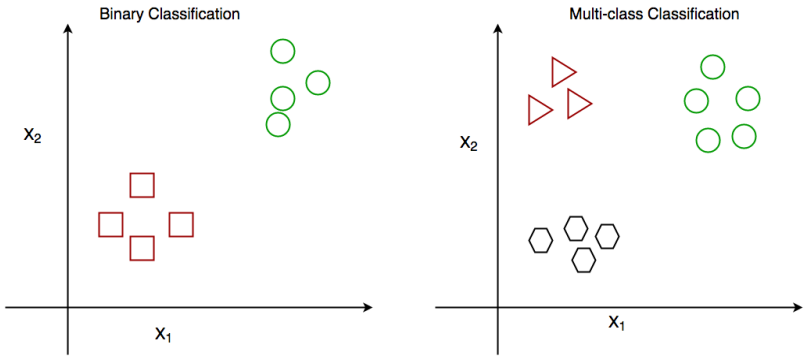
הסבר בתכל'ס על האלגוריתם

כמה מילים על למידה מפוקחת / $Supervised\ Learning$-

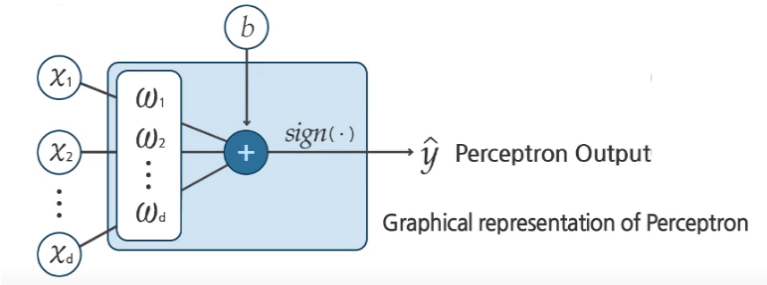
- מתחילים עם אוסף של פיצ'רים (תכונות) מצורה של זוגות וקטור/ערכים.
- המטרה : למצוא מודל שמנבא ערך עבור פיצ'ר חדש שלא נראה בוקטור בעבר.

כמה מילים על סיווג בינארי / $Binary\ Classification$-

- תפקיד הסיווג הוא לסווג את הנתונים עם תוויות ($Label$) מסוימות.
- אם יש לנו שני סוגים של תוויות, המשימה שלה נקראת סיווג בינארי, ואם יש יותר מ-2, אז המשימה היא סיווג מרובה. בסיווג בינארי, משתנה (או תווית) הוא 0 או 1, או נכון או לא נכון או '-' ו- '+'.



אז גילינו שכדי לטפל בסיווג בינארי, אנחנו צריכים למצוא היפר-מישור מה- $data$ שקיבלנו.
איך אנחנו יכולים לעשות את זה? - תשובה אחת היא $Perceptron$.
ה-$Perceptron$ הוא אלגוריתם ללמידה מפוקחת של בעיית סיווג בינארי.
הוא דורש את ה-$dataset$ הכולל את קלט x ואת התיוג שלו y.



מהאיור, אנו יכולים לנחש בקצרה את תהליך הניבוי של ה$Perceptron$.
סימן הסכום של כל הערך הופך לתווית החזויה.
בקיצור, זה סכום הערך מפורק לפי מכפלה פנימית בין המשקל לווקטור הקלט, עם הוספת הטיה.
אפשר לראות כי תהליך זה דומה להגדרה של היפר-מישור.

אז אנחנו יכולים להשתמש באלגוריתם הזה כדי למצוא את המישור וזה מה שעשינו בפרק הזה.

בהינתן $dataset$ שצריך לאמן: $(X_1, y_1),\ (X_2, y_2),...,(X_{10}, y_{10})$ כאשר $y_i \in \{+, -\}$.

בקבוצה 1 יש את התיוג '+' ולקבוצה 2 יש את התיוג '-'.

אנחנו יכולים להגדיר את המישור המסווג (Hyperplane) בצורה הבאה:

$$f(X) = W^T X = 0$$

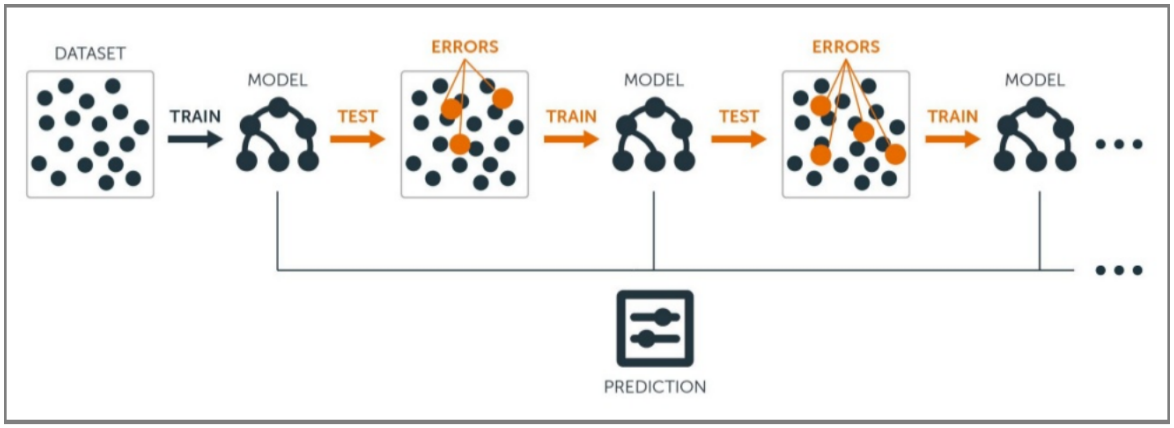
ואם נשתמש באלגוריתם לפי השלבים, אנחנו נמצא באופן מהיר יחסית מישור מסווג שהוא גם אופטימלי.

אלגוריתמי חיזוק - Boosting

- 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 אינטואטיבית, מודל חלש הוא כזה שתוצאותיו מעט טובות יותר מניחוש אקראי בעוד שאחד חזק מתקרב ממש לביצועים אופטימליים.
- העיקרון המנחה כאן הוא לשרשר את המודלים באופן כזה שכל מודל שמתווסף יטפל בשגיאות שקודמיו פספסו.
- היופי הנעוץ בכך ש-Boosting מוכיח כי למידה חלשה בהכרח מצביעה על קיום של שיטת למידה חזקה.
- לרוב, מודלים מבוססי Boosting מתמקדים בבעיות סיווג בינארי.

אנו נלמד את אלגוריתמי החיזוק הבאים:

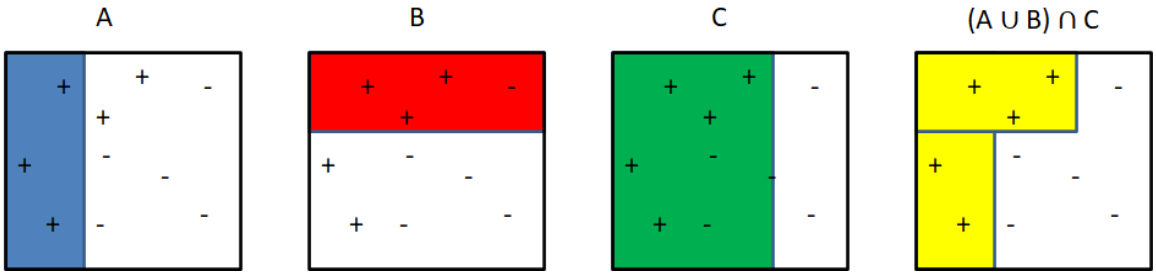
1. Adaboost
2. Clarkson 1
3. Clarkson 2



אלגוריתם AdaBoost

מבוא

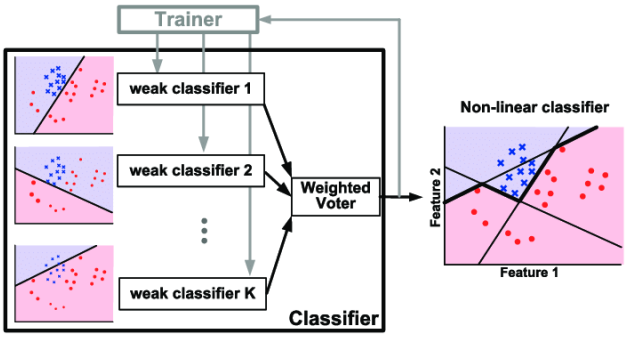
- אלגוריתם AdaBoost הוא אלגוריתם האצה או Boosting Algorithm כלומר שהוא בונה "לומד חזק".
- אלגוריתם AdaBoost = Adaptive Boost זהו אלגוריתם למידה המחפש מספר קטן של מסווגים/חוקים "חזקים" מתוך קבוצה של מסווגים/חוקים "חלשים".
- אלגוריתם AdaBoost מאפשר לחבר מספר קטן של חוקים לא כל כך טובים ולהשיג חוק מאוד טוב שהטעות האמפירית שלו קטנה.
- האלגוריתם מעניק משקל גדול לשגיאות בזיהוי ובכך מגדיל את סיכוייהן לסיווג מתאים בהמשך.



שאלה

האם נוכל לחבר חוקים חלשים כדי להפיק חוק חזק יותר?

- ניתן לכל חוק משקל בעל חשיבות מסוימת (k ,....,1 חוקים/מסווגים חלשים כמו בציור למטה).
- ואז ניקח הצבעה משוקללת (Weighted Voter).



מה שהוא הולך לעשות זה לתת לכל חוק מדד חשיבות (משקל).

נגיד שיש מיליון חוקים ו-10 מהם ממש חשובים אז ניתן למשל לראשון את המשקל '10' והשני את המשקל '7' וכו...

ואז בהינתן נקודה חדשה נשאל את החוק הראשון איפה הוא ביחס לנקודה החדשה ונעדכן את המשקל שלו בהתאם. נשאל כך גם את החוק השני, השלישי וכך הלאה...

שאלה

כמה חוקים חלשים אנחנו צריכים לבחור?

- הפשרה בין ההטייה לבין השונות
  - הטייה/bias - טעות אמפירית במדגם
  - שונות/variance - מורכבות הפתרון (מימד VC).
- מושג ה-Overfitting
  - שגיאה אמפירית נמוכה אך מימד VC גבוה
  - לא טוב בעולם (שגיאה אמיתית)

האלגוריתם

- קלט:
  - קבוצה S של נקודות, $x_i \in S$, עם התיוגים y_i
  - k מספר האיטרציות
  - קבוצה H של T חוקים חלשים כאשר $h_j: S \to \{-1, 1\}^{|S|}$
- פלט: משקל α_j לכל חוק h_j.
- פונקציות ההחלטה:
  - $F(x) = \Sigma_{t=1}^{T} \alpha_t h_t(x)$
  - $H(x) = sign[F(x)]$

צעדי האלגוריתם

1. נאתחל את משקלי הנקודות $D_1(x_i) = \frac{1}{n}$.
2. לכל איטרציה $t = 1 \ldots k$ נבצע:
3. ניתן מדד איכות על כל חוק - כלומר לכל $h \in H$ ניתן את משקל השגיאה שלו:

$$\varepsilon_t(h) = \Sigma_{i=1}^{n} \left(D_t(x_i) \cdot \left[h(x_i) \neq y_i \right] \right)$$

4. ניקח את המסווג/החוק עם הטעות הממושקלת הכי נמוכה h_t:

$$h_t = argmin_h \left(\varepsilon_t(h) \right)$$

5. ניתן לחוק h_t את המשקל שלו α_t בהתבסס על הטעות שלו:

$$\alpha_t = \frac{1}{2} \cdot \ln\left(\frac{1-\varepsilon_t(h_t)}{\varepsilon_t(h_t)} \right)$$

- אם הטעות הממושקלת היה ממש קטן (קרוב ל-0) אז היינו רוצים שהמשקל של החוק יהיה גבוהה.
- אם הטעות הממושקלת היה קרוב ל-½ (הכי גרוע), אז המשקל של החוק יהיה ממש נמוך.

6. נעדכן את המשקלים על הנקודות:

אם טעינו על נקודה אז המשקל שלו יגדל בסיבוב הבא.

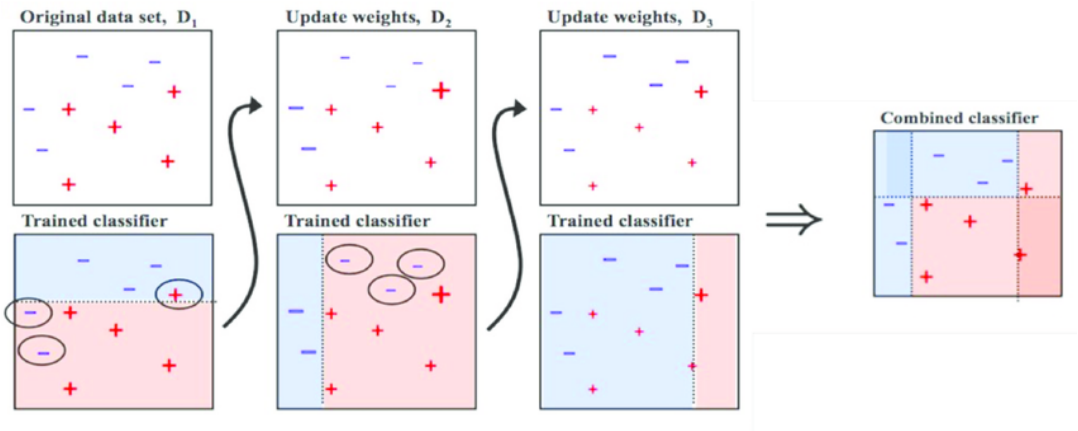
אחרת, אם צדקת על הנקודה אז המשקל שלו יקטן.
כשנבחר את החוק הבא, נרצה לבחור חוק שיתפוס נקודות שהחוק הנוכחי לא הצליח לתפוס.

$$D_{t+1}(x_i) = \frac{1}{Z_t} \cdot D_t(x_i) \cdot e^{-\alpha_t h_t(x_i) y_i}$$

כאשר Z_t הוא מספר קבוע שמנרמל את הביטוי ומקיים $\Sigma_i D_{t+1}(x_i) = 1$.

- לגבי הביטוי $e^{-\alpha_t h_t(x_i) y_i}$ שהוספנו שם:
  - אם $h_t(x_i) = y_i$ אז $h_t(x_i) \cdot y_i = 1$ ומשקל הנקודה תקטן.
  - אם $h_t(x_i) \neq y_i$ אז $h_t(x_i) \cdot y_i = -1$ ומשקל הנקודה יגדל.

כלומר, אם הנקודה לא שייכת לסיווג הנוכחי, אז נוסיף לה חשיבות גבוהה יותר כדי שבאיטרציה הבאה יהיה לה משקל גבוהה יותר וכך נמצא חוק אחר שיטפל בה.
זה בעצם המשמעות של ה-"Boosting", ככה בונים מודל למידה חזק ע"י שיפורים בכל שלב.



הוכחה

$$D_{t+1}(x_i) = \frac{1}{Z_t} \cdot D_t(x_i) \cdot e^{-\alpha_t h_t(x_i) y_i}$$

$$= \frac{1}{Z_t Z_{t-1}} \cdot D_{t-1}(x_i) \cdot e^{-y_i \cdot (\alpha_t \cdot h_t(x_i) + \alpha_{t-1} \cdot h_{t-1}(x_i))}$$

$$= \frac{1}{Z_t Z_{t-1} \cdot \ldots \cdot Z_1} \cdot D_1(x_i) \cdot e^{-y_i \cdot \Sigma_{j=1}^{t} \alpha_j h_j(x_i)}$$

$$= \frac{1}{Z} \cdot \frac{1}{n} \cdot e^{-y_i \cdot F(x_i)}$$

כאן Z קבוע - המכפלה של כל הנירמולים:

$$Z = Z_t Z_{t-1} \cdot \ldots \cdot Z_1 = \frac{1}{n} \cdot \Sigma_{i=1}^{n} e^{-y_i F(x_i)}$$

טענה

$$EmpiricalError(H) \leq Z$$

$$\frac{1}{n} \cdot \Sigma_{i=1}^{n} \left[H(x_i) \neq y_i \right] \leq \frac{1}{n} \cdot \Sigma_{i=1}^{n} e^{-y_i F(x_i)}$$

- כלומר, ש-Z הוא החסם העליון של הטעות האמפירית של החוק H.

הוכחה

$$F(x_i) = sign\left(F(x_i)\right) \cdot \left|F(x_i)\right| = H\left(x_i\right) \cdot \left|F(x_i)\right|$$

- אם $H(x_i) \neq y_i$ אז:
  - מצד שמאל של אי השוויון:

$$Error(H(x_i)) = 1$$

  - מצד ימין של אי השוויון:

$$e^{-y_i F(x_i)} = e^{-y_i H(x_i)\left|F(x_i)\right|} = e^{\left|F(x_i)\right|} \geq 1$$

- אם $H(x_i) = y_i$ אז:
  - מצד שמאל של אי השוויון:

$$Error(H(x_i)) = 0$$

  - מצד ימין של אי השוויון:

$$e^{-y_i F(x_i)} = e^{-y_i H(x_i)\left|F(x_i)\right|} = e^{-\left|F(x_i)\right|} \geq 0$$

נרצה למזער את Z.

נזכיר ש-Z הוא מכפלת כל הנירמולים $Z = Z_t Z_{t-1} \cdot \ldots \cdot Z_1$ ונשים לב:

$$Z_t = \Sigma_{i=1}^{n} D_t(x_i) \cdot e^{-y_i \alpha_i h(x_i)} =$$

$$= \Sigma_{x_i \in A} D_t(x_i) \cdot e^{-\alpha_t} + \Sigma_{x_i \in \bar{A}} D_t(x_i) \cdot e^{\alpha_t}$$

כלומר - עבור הנקודות שצדקנו המשקל שלהם ירד + הנקודות שטעינו אז המשקל שלהם יעלה

נגזור את Z_t:

$$0 = Z_t' = -\Sigma_{x_i \in A} D_t(x_i) \cdot e^{-\alpha_t} + \Sigma_{x_i \in \bar{A}} D_t(x_i) \cdot e^{\alpha_t}$$

נרצה לבחור α_t שיקיים לנו:

$$\Sigma_{x_i \in A} D_t(x_i) = \Sigma_{x_i \in \bar{A}} D_t(x_i) \cdot e^{2\alpha_t}$$

לכן נבחר:

$$\alpha_t = \frac{1}{2} \cdot \ln\left(\frac{1-\varepsilon_t(h_t)}{\varepsilon_t(h_t)}\right)$$

שזה בדיוק הערך שהאלגוריתם בוחר עבור α_t.

- האלגוריתם בחר את הנקודה שתאפשר את הנירמול הכי קטן.

אם ניקח את α_t שימזער את Z_t נקבל:

$$Z_t = \Sigma_{x_i \in A} D_t(x_i) \cdot e^{-\alpha_t} + \Sigma_{x_i \in \bar{A}} D_t(x_i) \cdot e^{\alpha_t} =$$
$$= 2\sqrt{\varepsilon_t(h_t) \cdot (1 - \varepsilon_t(h_t))}$$

באלגוריתם הזה, הטעות האמפירית דועכת באופן אקספוננציאלי.

נחליף $\gamma_t = \frac{1}{2} - \varepsilon_t(h_t) \quad , \quad \gamma_t \in [0, \frac{1}{2}]$ ונקבל:

$$Z_t = \sqrt{1 - 4(\gamma_t)^2} \leq e^{-2(\gamma_t)^2}$$

ולכן נקבל ש:

$$EmpiricalError(H) \leq Z \leq e^{-2\Sigma_{t=1}^{|T|}(\gamma_t)^2}$$

וקטן מהר יותר ככל שהטעויות קטנות יותר.

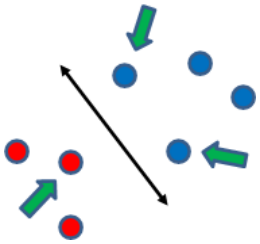
∎

אלגוריתם Clarkson הראשון

מבוא

- ה-Support Vector Machine או SVM הוא המפריד הליניארי אשר יוצר מרווח גדול ככל האפשר בינו לבין הדוגמאות הקרובות לו ביותר בשתי הקטגוריות.
- כאשר מוצגת נקודה חדשה, האלגוריתם יזהה האם היא ממוקמת בתוך הקו המגדיר את הקבוצה, או מחוצה לו.

שאלה



ראינו שזמן הריצה של אלגוריתם Perceptron הוא $O(\frac{1}{\gamma^2})$.

כמה יעלה לנו למצוא את המישור עם המרווח הגדול ביותר?

הגדרה

ה-**Support Vector** של S הם נקודות שמגדירים את המרווח הגדול ביותר של S.

- אם אחד מהנקודות נמחק, ייתכן שנמצא מרווח גדול יותר.
- כל תת קבוצה של S שמכילה את ה-Support Vectors בעלי אותה מרווח מקסימלי.
- במרחב d מימדים יש לכל היותר $d + 1$ וקטורים תומכים (Support Vectors).

אלגוריתמי קלרקסון

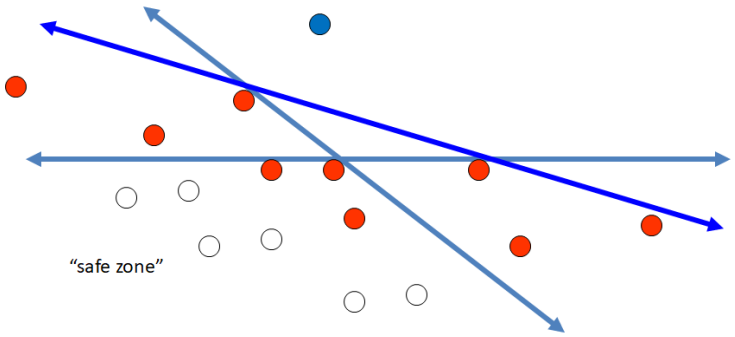
קלרקסון נתן 2 אלגוריתמים כדי למצוא את המישור עם המרווח הכי גדול.

הוא בעצם שבר את הבעיה של מציאת המישור לתתי בעיות וכך הוא ייעל את השיטה ופתרון הבעיה.

יהי P אלגוריתם שמוצא את המרווח הגדול ביותר.

יהי $P(n, d)$ פונקציה על קבוצה בגודל n ממימד d.

במקום להריץ את P על כל הקבוצה, אנחנו נרוץ בתתי-בעיות שקטנות מ-d.



אלגוריתם קלרקסון 1

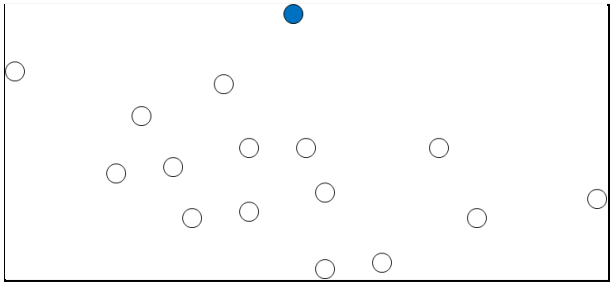
קלט: קבוצת הנקודות S בגודל n ממימד d.

1. נגדיר קבוצה $S' = \emptyset$
2. נדגום מתוך S תת קבוצה B מגודל m.
3. $S' = S' \cup B$
4. נריץ P על S' שימצא את המרווח המקסימלי h על הקבוצה S' בזמן $P(|S'|, d)$.
5. אם h הוא המרווח המקסימלי שמסווג את הקבוצה S, נסיים את האלגוריתם.
6. נוסיף ל-S' נקודות של S שלא סווגו על ידי h.
7. נחזור לשלב 4.

כלומר במקום להריץ את כל הנקודות של S כדי למצוא את המרווח המקסימלי, הרצנו רק על $O(d\sqrt{n})$ נקודות.

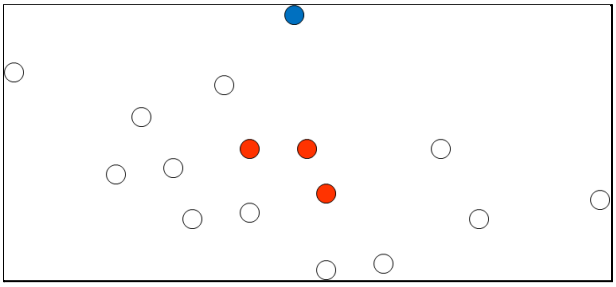
דוגמה ויזואלית לקלרקסון 1

נניח שיש קבוצת קודקודים ואחד מהם בצבע כחול.

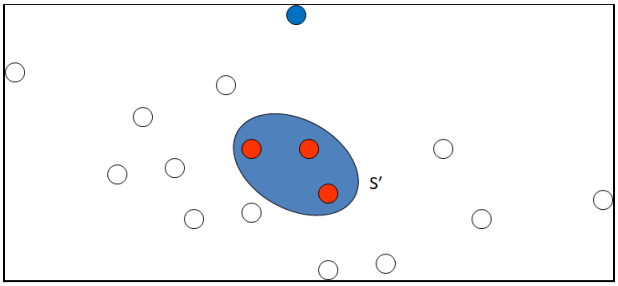


שלב 1: נגדיר קבוצה חדשה S'.

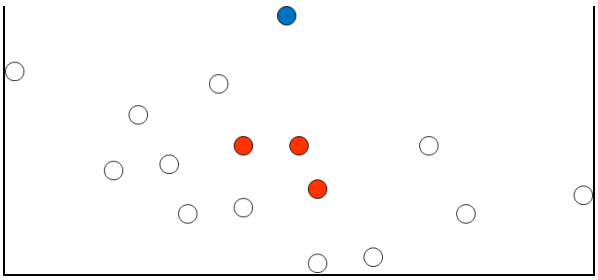
שלב 2: נדגום מתוך S תת קבוצה B מגודל 3.



שלב 3: $S' = S' \cup B$

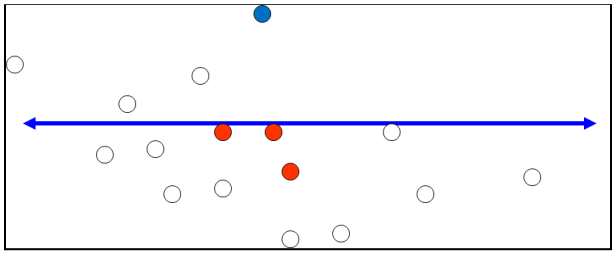


שלב 4: נריץ P על S' בזמן $P(|S'|,\ d)$.



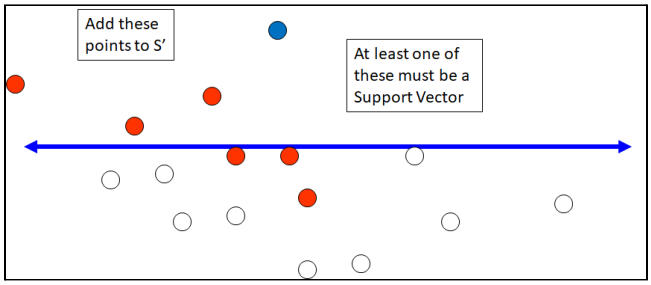
שלב 5: נמצא את המרווח המקסימלי ביותר h על S'.

נשים לב שכשאנחנו מדברים על המרווח המקסימלי, מדובר על המרווח הגדול ביותר בין הצומת **הכחולה** לבין צומת אחרת **אדומה**.

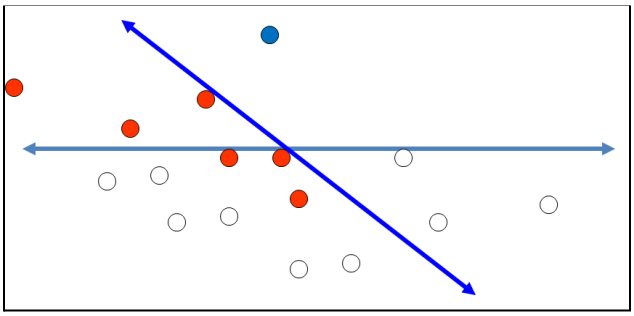


נחזור לשלבים הקודמים:

נוסיף 3 קודקודים חדשים ל-S' ונריץ P על S' בזמן $P(|S'|,\ d)$.

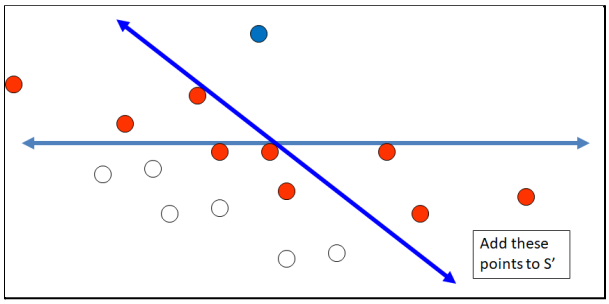


שלב 5: נמצא את המרווח המקסימלי ביותר h על S'.

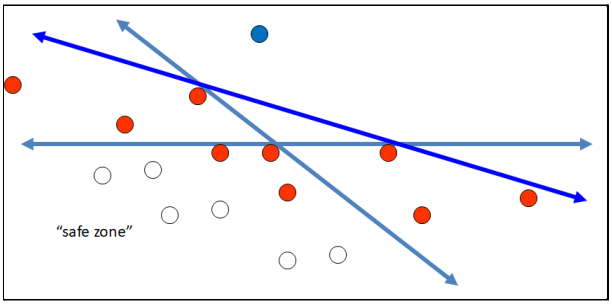


נחזור לשלבים הקודמים:

נוסיף 3 קודקודים חדשים ל-S' ונריץ P על S' בזמן $P(|S'|,\ d)$.



שלב 5: נמצא את המרווח המקסימלי ביותר h על S' ונסיים את האלגוריתם.



טענה

בכל איטרציה, h עקבי בכל ה-S, או מסווג שגוי של לפחות וקטור תמיכה SV אחד.

הוכחה

לפי הגדרה, מישור בעל מרווח מקסימלי שעקבי על כל וקטורי התמיכה גם עקבי על כל הנקודות ב-S.

∎

הערה

קיימים $d\ +\ 1$ וקטורי תמיכה SV, אז יש לכל היותר $d\ +\ 2$ איטרציות.

שאלה

כמה גדול קבוצה המדגם שלנו S צריך להיות?

נזכר כי בהסתברות $(1\ -\ \frac{1}{m})$, כל השערה שמסווגת נכון את כל הנקודות של מדגם S' של m נקודות, מסווגת לא נכון נקודות מ- S בלכל היותר בגודל:

$$O\left(\frac{n}{m}\cdot\left[\log(m)\ +\ d_{VC}\log\left(\frac{m}{d_{VC}}\right)\right]\right)=\ O\left(d_{VC}\frac{n}{m}\right)$$

ניתוח קלרקסון 1

נבחר $m = O\left(d\sqrt{n}\right)$.

בכל איטרציה, h מסווג לא נכון את המספר הבא של נקודות:

$$O\left(d_{VC}\frac{n}{m}\right) = O\left(d\frac{n}{d\sqrt{n}}\right) = O\left(\sqrt{n}\right)$$

ולאחר $d + 2$ איטרציות נקבל ש:

$$|S'| = O\left(d\sqrt{n}\right)$$

אלגוריתם Clarkson השני

האלגוריתם השני של קלרקסון משתמש בשיטה איטרטיבית עבור עדכון משקלי הנקודות.

תכונות

- עובד עם איטרציות מרובות של דגימה עם החלפה.
- כל איטרציה מייצרת היפר-מישור (מישור-על).
- לכל נקודה יש משקל שאפשר לעדכן אותו בהתאם לשיטת האלגוריתם.

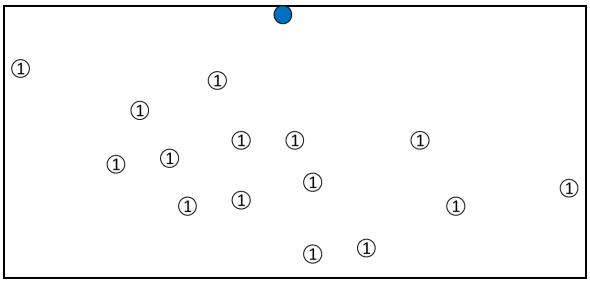
האלגוריתם

קלט: קבוצה **S** של n נקודות ממימד d, וקבוע כלשהו **c**.

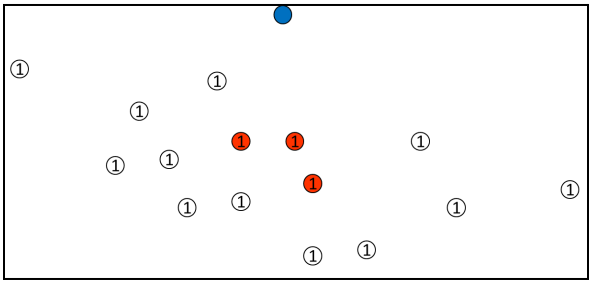
1. לכל נקודה $x \in S$ נציב את המשקל $w(x) \leftarrow 1$.
2. נגדום מתוך S לפי משקל את תת הקבוצה **B** מגודל cd^2.
3. נריץ את P על B כדי למצוא את המרווח h המפריד המקסימלי ביותר של הנקודות מ-B בזמן של $P(cd^2, d)$.
4. אם h עקבי על **S**, נסיים את האלגוריתם.
5. עבור כל נקודה $x \in S$ שסווגה באופן שגוי על ידי h נשנה את המשקל שלה להיות $w(x) \leftarrow 2w(x)$.
6. חזור לשלב (2).

דוגמה ויזואלית

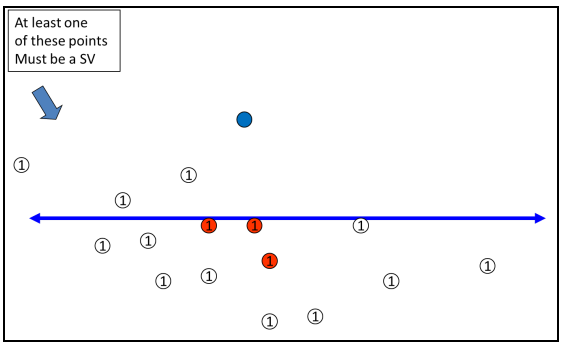
עבור אוסף של נקודות S:



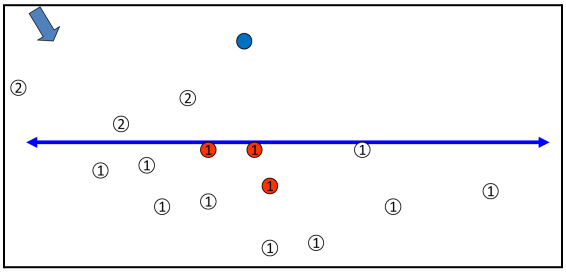
נדגום תת-קבוצה B מגודל m.



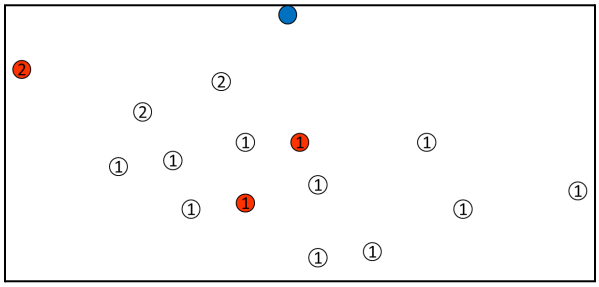
לפחות אחד מהנקודות חייב להיות ה-Support Vector.



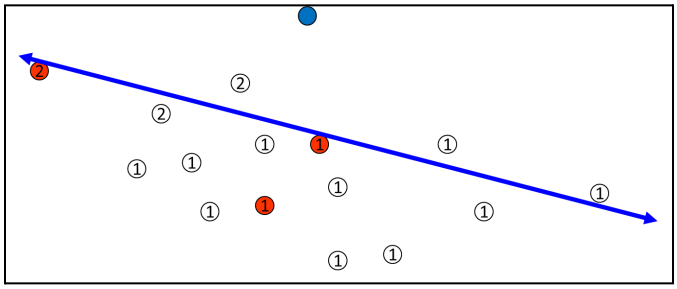
נעדכן את משקלי הנקודות האלה:



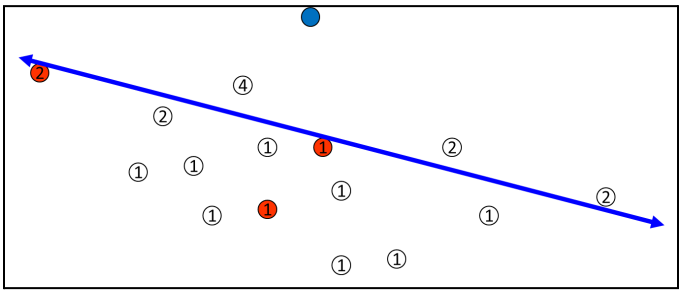
באיטרציה הבאה נדגום שוב תת-קבוצה B מגודל m



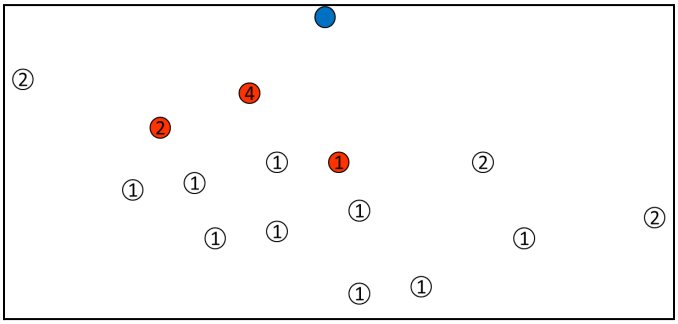
נחשב את המרווח המפריד הגדול ביותר



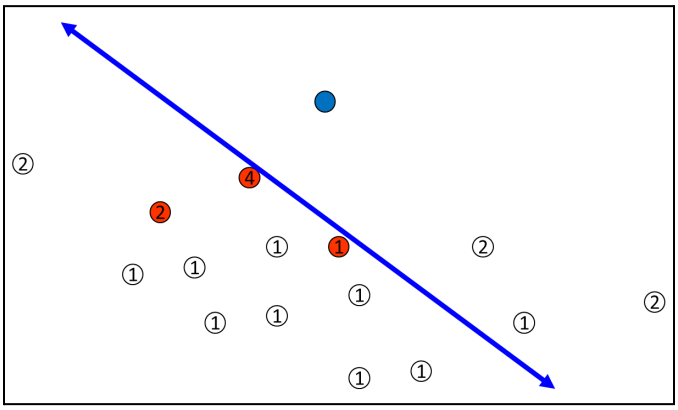
נכפיל את המשקלים של הנקודות שטעו בסיווג



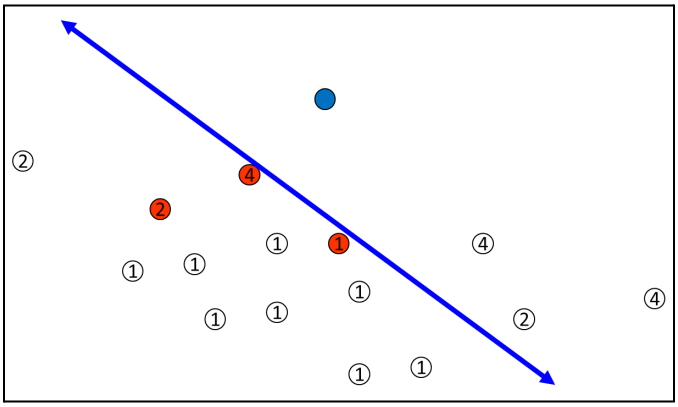
באיטרציה הבאה נדגום שוב תת-קבוצה B מגודל m



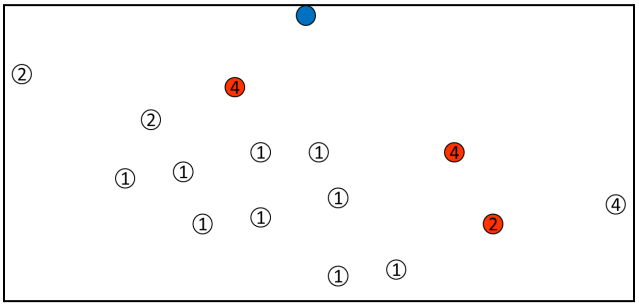
נחשב את המרווח המפריד הגדול ביותר



נכפיל את המשקלים של הנקודות שטעו בסיווג

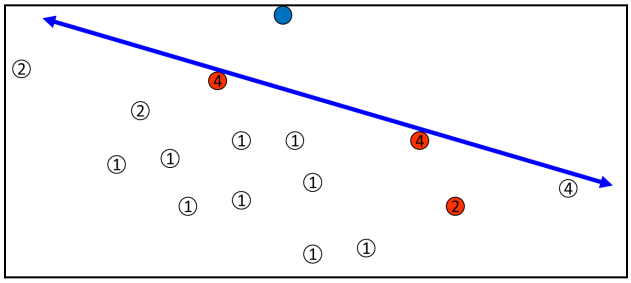


באיטרציה הבאה נדגום שוב תת-קבוצה B מגודל m



נחשב את המרווח המפריד הגדול ביותר.

מאחר והמרווח המפריד (המישור-על) עקבי על כל הנקודות ב-S אז נסיים את האלגוריתם.



אזהרה חשובה

- אם המשקל של נקודות שסווגו בטעות ב-h כבדים מאוד ($2n/cd$ או יותר), אז סיים את האיטרציה ותדגום מחדש.
- הסיכוי שזה יקרה קטן:

$$O(d_{VC}n/m) = O(dn/d^2) = O(n/d)$$

נכונות האלגוריתם

כדי להציג את נכונות האלגוריתם יש להציג 2 עובדות:

1. *<u>עובדה 1</u>*:
   a. בכל איטרציה, משקלו של SV אחד לפחות מוכפל.
   b. לאחר איטרציות d+1, המשקל של קבוצת ה-SV הוא לכל הפחות מוכפל.
   c. לאחר איטרציות $k(d+1)$, המשקל של קבוצת SV הוא לפחות $2^k(d+1)$.
2. *<u>עובדה 2</u>*:
   a. בכל איטרציה, המשקל של כל הקבוצה גדלה לכל היותר ב- $2w(S)/cd$.
   b. לאחר kd איטרציות, המשקל של כל הקבוצה היא לכל היותר:

$$(1+\frac{2}{cd})^{kd}n < e^{2\frac{k}{c}}n$$

לכן, המשקל של קבוצת ה-SV לא יכולה להיות גדולה יותר ממשקל של הקבוצה כולה.

$$2^k d < e^{2\frac{k}{c}} n$$

ומספר האיטרציות / סיבוכיות:

$$kd = O(d \log(n))$$

אלגוריתמי הפרדה SVM

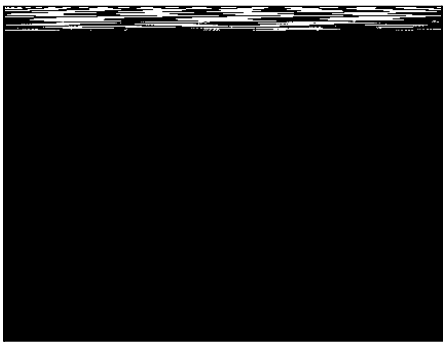
מבוא (לא מההרצאה)

- מודל ה-SVM או Support Vector Machine 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 הוא לינארי, כאשר חלוקת הדוגמאות במרחב הוקטורי נעשית באופן כזה שיווצר מרווח גדול ככל האפשר בין המישור המפריד לבין הנקודות הממוקמות הכי קרוב אליו.
- מרווח זה מכונה שוליים (margin), כאשר בצד האחד של השוליים נמצאות דוגמאות עם label אחד, ובצד השני נמצאות הדוגמאות עם ה-label השני.

אנחנו נלמד בקורס את אלגוריתמי SVM הבאים:

1. Hard Margin
2. Soft Margin
3. RBF Kernel
4. Polynomial Kernel

מה זה Hard Margin או Hard SVM



- במצב הפשוט ביותר, המשוואה עבור כל אחד מצדדיו של המפריד הינה פונקציה לינארית של המאפיינים וכל הדוגמאות אשר סווגו נכון.
- מצב זה מכונה "הפרדה קשיחה" בו האלגוריתם מוצא את המישור עם השול הרחב ביותר האפשרי, ולא מאפשר לדוגמאות להיות בין הוקטורים התומכים.
- זוהי למעשה ההפרדה המושלמת, והוקטורים התומכים הם למעשה הנקודות בקצוות השוליים כמו באיור משמאל:

מה זה Soft Margin או Soft SVM

- נקרא "הפרכה רכה" נרצה למצוא מפריד טוב ככל הניתן למרות שיש לנו טעויות.

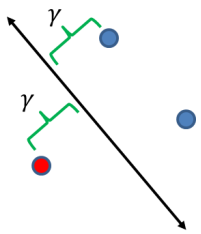
הפרדת היפר-מישורים

ראינו שיש כמה אלגוריתים שמטרתם למצוא את המישור המפריד (Hyperplane):

- Brute Force
- Winnow
- Perdeptron

שאלה

מה ה-VC_{Dim} של מישורים מפרידים (Hyperplanes)?



- ראינו שעבור d מימדים יש $d+1$ (זה יכול להיות גדול).
- ראינו שהמרווח γ מגדיר $\frac{O(1)}{\gamma^2}$.

שאלה

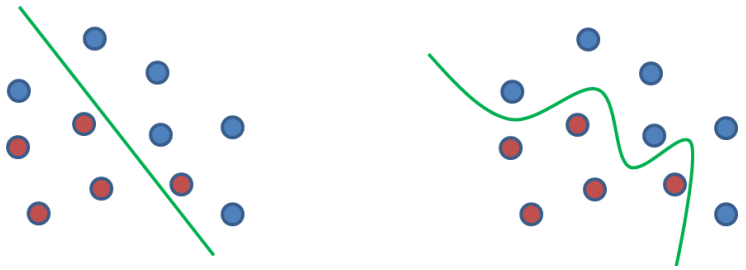
למה VC_{Dim} גבוהה זה רע?

אם יש לנו מסווג עם VC גבוהה שמתאים למדגם אז הוא דיי חסר משמעות.

אם ה-VC גבוהה אז אנחנו בעצם לומדים את המדגם אבל לא את העיקרון מאחורי המדגם.

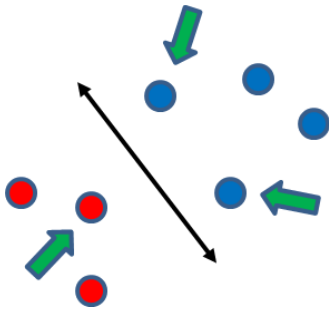
אפשר לראות בציור למטה כי המקרה הימני יותר תואם את המקרה בציור השמאלי.

הבעיה הזאת נקראת *Overfitting* - אנחנו לומדים את המדגם ממש טוב אבל לא את ההתנהגות שלה.



נרצה למצוא מישור המפריד את הנקודות עם המרווח הכי גדול.

תזכורת - Support Vector



ה-**Support Vector** של S (נסמן מעכשיו SV) הם נקודות שמגדירים את המרווח הגדול ביותר של S.

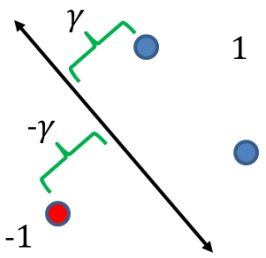
- אם אחד מהנקודות נמחק, ייתכן שנמצא מרווח גדול יותר.
- כל תת קבוצה של S שמכילה את ה-Support Vectors בעלי אותה מרווח מקסימלי.
- במרחב d מימדים יש לכל היותר $d+1$ וקטורים תומכים (SV).

הפרדה קשיחה - Hard Margin SVM

נניח ויש לנו נתונים הניתנים להפרדה.
נרצה למצוא את המישור שמפריד את הנקודות עם *מרווח הכי גדול*.
ה-*הפרדה הקשיחה* או Hard margin עקבי עם קבוצת הנקודות S.

בעיה

נרצה להפריד את מדגם הנקודות S.
בהינתן קבוצה S של נקודות כך ש: $x_i \in S$ עם התיוגים $y_i \in \{-1, 1\}$.



פתרון

נרצה למקסם את המרווח γ ולמצוא w מתאים שיקיים את זה,
כאשר הביטוי הבא חייב להתקיים: $y_i[w \cdot x_i + b] \geq \gamma$ לכל i וגם $||w|| = 1$.

עבור $w \cdot x_i + b$ זה בדיוק המרחק של נקודה ממישור.

אם ה-w מנורמל אז $w \cdot x_i + b$ בדיוק מגדיר את המישור כאשר w זה הכיוון ו-b זה המרחק לכיוון למעלה.

לכן עבור הביטוי שהצגנו $y_i[w \cdot x_i + b] \geq \gamma$ אם הנקודה היא חיובית כלומר $y_i = 1$ אז זה אומר שהנקודה החיובית שאנו מתמקדים בה חייבת להיות רחוקה מהמישור בלפחות γ (כמו בציור מלמעלה - הנקודות הכחולות הם החיוביות).

נוהגים להשמיט את b וזה בעצם אומר שהנקודות יושבות על ראשית הצירים וזה תקין.
אם ניקח את העולם שלנו ונרחיק אותו מאוד מראשית הצירים, אז המישור בגלל שהוא רחוק מראשית הצירים אז אם נעכל אותו קצת יהיה אפשר לגעת איתו בראשית הצירים ולכן מוסיפים עוד מימד כדי להניח שהמישור נוגע בראשית הצירים וכך יהיה יותר קל להתמודד בלי ה-b.

נשים לב ש:

- אם $y_i = 1$ אז $wx_i \geq \gamma$
- אם $y_i = -1$ אז $wx_i \leq -\gamma$

ה-SVM לא מנסה למקסם את γ כאשר $||w|| = 1$ אלא הוא מנסה למזער את w כאשר $\gamma = 1$.

פתרון נוסף

יש דרך אחרת לכתוב את הפתרון שכתוב למעלה:

- אם w מנורמל אז יש ל-γ $y_i[w \cdot x_i] \geq \gamma$ משמעות: ה-γ הוא המרחק מהמישור ונרצה למקסם אותו.
- אם w לא מנורמל אז משהו אחר חייב להחליט את לנו מהם המרחקים, לכן ניקח את המרווח הגדול ביותר $\gamma = 1$.

אנחנו רוצים בעצם לקבוע את γ ולמזער את w.

המטרות שלנו:

- נסמן $\min_{x_i}$ היא הנקודה הכי קרובה אל המישור ונרצה למקסם את $\min_{x_i} \frac{|w \cdot x_i|}{||w||}$.
- נקבע מה המרחק הכי קטן $|w \cdot x_i| \geq 1$ כלומר נקבע ש- $\gamma = 1$.
- אם נרצה למקסם את $\frac{1}{||w||}$ אז נרצה למזער את $||w||$.

המכפלה הסקלרית בין וקטור (x_i) ובין וקטור מנורמל (w) נותנת את ההיטל, לכן אנחנו יכולים לדבר על מרחק של נקודה x_i מהמרווח (כי הנקודה x_i היא כמו וקטור).

בעיה

בהינתן קבוצה S של נקודות כך ש: $x_i \in S$ עם התיוגים $y_i \in \{-1, 1\}$.

פתרון

אם הנקודות x_i לא מנורמלות אז:

נרצה למצוא w שיש לו את הנורמה הכי קטנה $||w||$ שעדיין מקיים את $y_i[w \cdot x_i] \geq 1$ לכל i.

העובדה שהדבר הזה ניתן לפתירה יחסית מהר הוא העבודה למה SVM הוא כל כך פופולארי.

Soft Margin SVM - הפרדה רכה

מישורים עם נקודות שלא ניתנות להפרדה

הצגנו מקרים בהם אנחנו חייבים להיות עקביים על כל הנקודות בעזרת $y_i[w \cdot x_i] \geq 1$.

כעת אנחנו נפגשים במקרה בו אוסף הנקודות לא ניתנות להפרדה, כלומר ישנם נקודות שלא נוכל להפריד אותם בעזרת מישור, לכן יהיה לנו טעות אמפירית $\bar{e}$.

נזכר בחסם ההכללה, אם דגמנו אוסף של n נקודות אז בהסתברות $1 - \delta$ מתקיים:

$$e(h) \leq \bar{e}(h) + \sqrt{\frac{8 \cdot d \cdot \ln\frac{2 \cdot e \cdot n}{d} + 8 \cdot \ln\frac{4}{\delta}}{n}}$$

כלומר, גם אם יש לנו טעויות אנחנו עדיין נרצה להפריד אותם עם איזשהו מרווח.

פשרה

- אם יש לנו מרווח גדול אז ה-d (בחסם ההכללה) יהיה יותר קטן $d = \frac{0(1)}{\gamma^2}$.

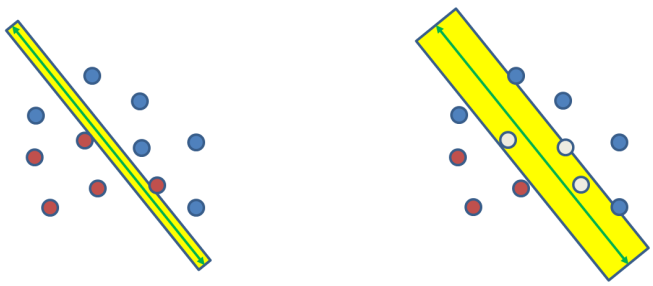
אבל בגלל שהמרווח גדול אולי אנחנו נטעה על יותר נקודות, כלומר יותר נקודות יפלו בתוך המרווח ואותם ניקח בתור הטעויות שלנו.

- אם יש לנו מרווח קטן אז אולי הטעות תהיה קטנה יותר, כלומר ה-$\bar{e}(h)$ תהיה קטנה יותר.

האיזון של הפשרה בעצם מתעסק בחסם ההכללה בין השבר הגדול עם d לבין הטעות האמפירית $\bar{e}(h)$.

דוגמה עבור הפשרה שהצגנו

בשביל אותו מישור אפשר לקחת מרווחים שונים:



- נשים לב כי בציור השמאלי יש מרווח קטן, אומנם הוא טועה על הנקודות אבל הטעות היא קטנה, זה בעצם מייצג את הטעות האמפירית $e(h)$ שהוא קטן יותר.
- נשים לב בציור הימני, יש מרווח יותר גדול אז כל הנקודות שנופלות בפנים הוא טועה עליהם ולכן נתייחס אליהם בתור שגיאה, לכן הטעות האמפירית שלנו יותר גדולה אבל יש לו VC_{Dim} יותר קטן כלומר לשאר הנקודות הוא השיג הפרדה יותר גדולה.

לכן, אנחנו צריכים למצוא איזון שפותר את הבעיה הזאת.

מזעור טעויות

בהינתן אוסף S של נקודות ב-d מימדים.
נרצה למצוא את המישור שעושה את ההכי פחות שגיאות.

הבעיה:
בהינתן S נרצה למצוא היפר-מישור ממימד $d - 1$ עם הטעות הכי קטנה שאפשר לקבל.

רק הבעיה למצוא את המישור הכי טוב אך עדיין עם שגיאות היא עדיין קשה מאוד לפתרון.

טענה

כדי למצוא את המישור הכי טוב למרות שיש שגיאות - הבעיה היא NP-Hard.

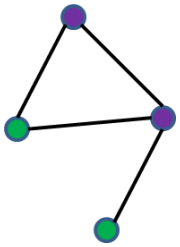
הוכחה

ברדוקציה לפי $Min\ Vertex\ Cover$.

מה זה Vertex Cover:

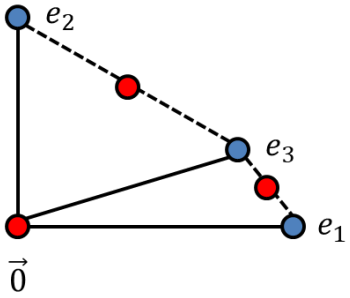
- קלט: $G = (V, E)$.
- פלט: תת הקבוצה הקטנה ביותר של V שמכסים את כל הקשתות.
- כדי לכסות קשה צריך שיהיה צומת אחד מכל צלע בתת הקבוצה הזאת.

לדוגמה, אם ניקח את תת האוסף עם הצמתים הסגולים אז כיסינו את כל הקשתות בגרף:



לא ידוע האלגוריתם שימצא את האוסף הכי קטן שמכסה את כל הקשתות.
אנחנו יכולים לקחת את הבעיה הזאת ולתרגם אותה בלתרגם את המישור שטועה על הכי מעט נקודות וכמעט אי אפשר את הבעיה הזאת ולכן אי אפשר לפתור את הבעיה של המישור.

ציור מישור מפריד

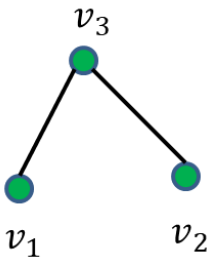


<u>*רדוקציה*</u>: בהינתן $G = (V, E)$ של $Vertex\ Cover$.
ניצור מופע S של בעיית שגיאת היפר-מישור מינימלית:

נבנה דוגמה של נקודות רב מימדיות שנרצה לפרש אותם על ידי מישור.

- לכן בראשית הצירים $\vec{0}$ אנחנו נשים נקודה אדומה עם משקל n (יש בו n נקודות) ולא נרצה לטעות בנקודה הזאת כי אז נטעה בכל ה-n נקודות.

ציור ה-$Vertex\ Cover$



- בשביל כל אחד מהצמתים בבעיה של $Vertex\ Cover$ כלומר לכל $v_i \in V$ אנחנו יוצרים נקודה כחולה במישור המפריד שהיא וקטור הבסיס e_i ונתייג אותם כ-"שלילי".
- אם יש שני צמתים $v_i, v_j \in V$ שיש ביניהם קשת, אז במישור המפריד נשים ביניהם נקודה אדומה בין e_i לבין e_j ולא נרצה לטעות על הנקודות האלה.

הרעיון - כל הנקודות האדומות הם n נקודות ולא נרצה לטעות בהם, הנקודות הכחולות הם אולי נקודה אחת ולכן נרצה לטעות דווקא בהם כדי למזער את הנזק של השגיאות שלנו. נרצה למזער על כמה נקודות כחולות אנחנו טועים.
יש מקבילות בין הבעיה ב-vertex לבין החיתוך של המישור - לא נרצה לשים גם e_1 וגם את e_3 בתוך החיתוך, לפחות אחד מהם חייב להישאר בחוץ בדומה לבעיה של vector.

טענה

ברדוקציה, הדברים הבאים שקולים:

- מישור-על (Hyperplane) טועה על $k < n$ נקודות ב-P.
- ה-$Vertex\ Cover$ ב-G הוא לכל היותר k.

הפרדה רכה - Soft Margin

Vapnik הציע SVM שמוצא עבור נקודות שלא ניתנות להפרדה.

נרצה לקחת את העיקרון שהצגנו:

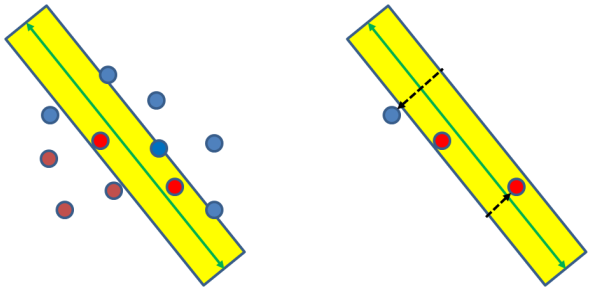
- נקבע מה המרחק הכי קטן $\gamma = 1$ ש- $\left|w \cdot x_i\right| \geq 1$ כלומר נקבע

ולמזער את הנורמה של w אך שהפעם נרצה גם עבור מקרה שיש נקודות שיפלו במרווח (זה נקודות טועות).

עקרון ה-Soft Margin בעצם אומר לנו שמותר לנו לעשות שגיאות:

על כל נקודה x_i שנמצאת בצד הלא נכון (נקודה שטועה) אנחנו נשלם ב-ζ_i ש בעצם מודד כמה הנקודה רחוקה מהמרווח.

- אנחנו נרצה לצמצם את סכום כל הטעויות שלנו.



נציג בציור את הצד השמאלי שמראה מי הם הנקודות שטועות ביחס לקבוצה המתאימה להם, ובצד הימני שמראה עד כמה הנקודות האדומות טועות מהמרווח הנכון, אנחנו לא רק נרצה שהנקודות הטועות יפלו בצד הנכון אלא נרצה שהם יפלו גם במרווח הטעות.

<u>בציור השמאלי</u>

נשים לב לציור למעלה, בצד שמאל שהנקודות הכחולות מתאימות לצד הימני של המישור והאדומות לצד השמאלי אבל אפשר לראות כי יש כמה נקודות שחורגות מהתחומים שלהם, להן אנחנו משלמים טעות בגודל ζ_i.

<u>בציור הימני</u>

אפשר לראות את המשקל של הטעות ה-ζ_i לכל נקודה x_i בגרף.

תוכנית עבודה עם Soft Margin

ראינו בעצם שיש לנו טעויות בנקודות מהמישור לכן הפעם נתעסק עם:

$y_i(w \cdot x_i) \geq 1 - \zeta_i$ לכל i וגם $\zeta_i \geq 0$.

כלומר, הפעם אם אנחנו בצד הלא נכון של המישור אז נשלם עליו ζ_i .

- אם בעבר רצינו למזער את $||w||$ כי ככל שהוא יותר קטן אז המרחק שלנו יותר גדול.
- כעת נרצה למזער שתי דברים בו-זמנית: את $||w||^2$ ואת $\Sigma_i\zeta_i$ שזה סכום הטעויות שלנו.

נרצה למזער את:

$$\frac{\Sigma_i\zeta_i}{n} + \lambda||w||^2$$

הגדרה

ה-Hinge Loss הוא ה-$\frac{\Sigma_i\zeta_i}{n}$ כאשר ה-hinge הוא הקו (המרחק או הזווית) מנקודת הטעות אל המישור.

<u>מה זה λ ?</u>

התפקיד שלו נועד למזער שתי דברים בו-זמנית, הוא מאפשר פשרה בין שני הדברים האלה.

ה-λ הוא האיבר שאמור למזער גם את הנורמה של w וגם את הסכום הנל.

<u>איך נבחר את λ ?</u>

נשתמש באלגוריתם $cross - validation$ - זה טכניקה לבחירת פרמטרים לאלגוריתם:

- ניקח את קבוצת המדגם שלנו ונחלק אותו ל-2 (בערך) ונקבל:
  - לחלק ראשון - נקרא לו ה-Training.
  - חלק שני - נקרא לו ה-Validation.
- נריץ את SVM על ה-Training עם כל מיני ערכים שונים של λ.
- ואז ננסה את λ על ה-Validation עד שנמצא את ה-λ ההכי טוב.

יש לנו מדגם אך אנו לא יכולים להסתכל על שאר העולם, אלא על סמך המדגם להחליט מה יהיה ה"איזון" בין סכום $\Sigma_i\zeta_i$ לבין ה-w המנורמל. אנחנו ננסה כל מיני ערכים של λ וננסה אותו על המישור עד שנמצא אחד שמתאים לנו ושיהיה הטוב ביותר מכולם.

אז מה המטרות שלנו עבור מקרה Soft Margin?

- נרצה למזער את סכום הטעויות שלנו ואת $||w||$ בעזרת בחירת λ הטוב ביותר:

$$MIN\ \frac{1}{n}\Sigma_i\zeta_i + \lambda||w||^2$$

- נרצה למזער את הנורמה של w, כלומר נמקסם את המרחק של הנקודות מהמישור אך יש לנו אפשרות לחרוג מהמרווח או אפילו להשים נקודות בצד הלא נכון ולשלם על זה $\zeta_i \geq 0$ כלומר לכל i :

$$y_i(w \cdot x_i) \geq 1 - \zeta_i$$

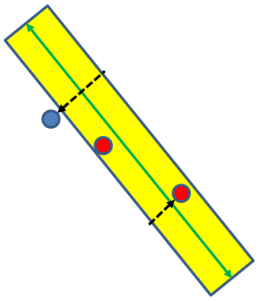
חסם הכללה

הוכחנו שאי אפשר למזער את מספר השגיאות באופן ישיר כי זה בעיה NP-Hard אבל זה כן נכון שסכום הטעויות חוסם מלמעלה את הטעות האמפירית (כלומר בכמה טעויות טעינו בהם):

$$hinge\ loss\ \geq\ empirical\ error$$

למה זה נכון? -

$$\zeta_i \geq \gamma\left[h\left(x_i\right) \neq y_i\right] \geq \left[h\left(x_i\right) \neq y_i\right]$$



אם טעינו על נקודה x_i כך שהיא נפלה ממש בצד הלא נכון אז זה טעות אמפירית ושילמנו עליה לפחות γ.

כי אם ניקח את המרווח γ ונכפיל אותו במספר השגיאות $h\left(x_i\right) \neq y_i$ (כלומר הנקודה נפלה בצד הלא נכון של המישור) אז נשלם על זה לפחות γ.

זה אומר שזה נכון למזער את כל ה-ζ_i כי כך אנחנו נמזער את מספר השגיאות ביחד.

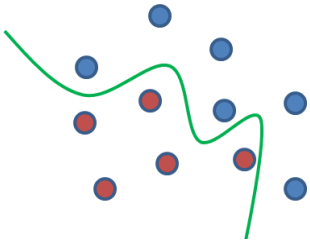
מה הטעות האמיתית של SVM?

זה נכון שאם w הוא המישור של ה-SVM אז עם הסתברות של לפחות $1 - \delta$ נקבל:

$$e(h) \leq \sum_i \zeta_i + \frac{4||w||}{\sqrt{n}} + (1 + 2||w||)\sqrt{\frac{2\ln\left(\frac{4||w||}{\delta}\right)}{n}}$$

אם h השיג על המדגם מרווח מסוים, אז הטעות האמיתית שלו על כל העולם (האם בצד הנכון או לא נכון על כל העולם) חסום ע"י הסכום של ה-ζ-ות (ראינו שהסכום חוסם את הטעות האמפירית) יחד עם $||w||$ ולכן נרצה למזער את שני הפרמטרים האלה כי שניהם נמצאים בתוך החסם הכללה.

הפרדה לא לינארית - Kernel SVM



- מה אם נרצה מסווג שהוא לא מישור (לא לינארי)?
- אולי הנתונים שלנו נראים הכי טוב כאשר יש לנו קו שהוא לא לינארי?
- אפשר די בקלות להפוך את SVM שמסווג באופן לינארי למצב של מסווג שהוא לא לינארי.

ה-$Soft\ Margin$ של SVM יכול להיכתב באופן הבא:

אנחנו נרצה למזער את סכום ה-c_i וגם את המכפלה הפנימית $(x_i \cdot x_j)$ של כל זוג וקטורים:

$$\min\ \Sigma_{i=1}^{n} c_i - \frac{1}{2}\Sigma_{i=1}^{n}\Sigma_{j=1}^{n} y_i c_i (x_i \cdot x_j) y_j c_j$$

כך שצריך להתקיים:

$$\Sigma_{i=1}^{n} c_i y_i = 0$$
$$0 \le c_i \le \frac{1}{2n\lambda}$$

הביטוי $(x_i \cdot x_j)$ נקרא ה-**Kernel**:

ניקח את המכפלה הפנימית של כל זוג של נקודות וכך ניצור מסווג יותר ויותר מסובך.

נחליף את $(x_i \cdot x_j)$ בפונקציות אחרות, הנפוצות ביניהם כיום הם:

- פונקציית Polynomial Kernel המוגדרת: $(x_i \cdot x_j + 1)^d$.
- פונקציית Radial Basis Function - RBF המוגדרת: $e^{-||x_i - x_j|| / 2\sigma^2}$.

אנחנו לא נתעסק בהם בקורס - רק הכרנו את הנושא באופן כללי.

חיפוש שכן קרוב: k-NN לסיווג

מרחק אוקלידי

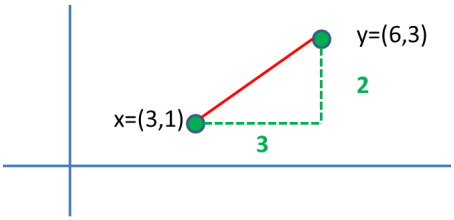
ידוע כמרחק l_2.

המרחק בין שני וקטורים $x,\ y$ יהיה:

$$||x - y||_2 = \left[\sum_{i=1}^{d} |x_i - y_i|^2\right]^{\frac{1}{2}}$$

למשל:

$$\sqrt{3^2 + 2^2} \approx 3.6$$



מרחק מנהטן

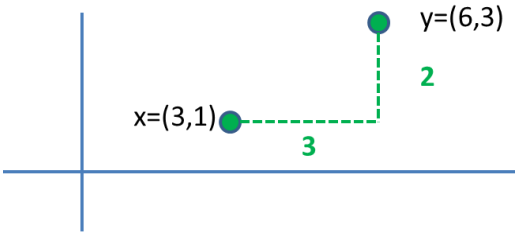
ידוע גם בתור מרחק Taxicab או l_1.

המרחק בין הוקטורים x, y הוא:

$$||x - y||_1 = \sum_{i=1}^{d} |x_i - y_i|$$

למשל:

$$3 + 2 = 5$$



מרחב l_p

זה הכללה של l_2, l_1.

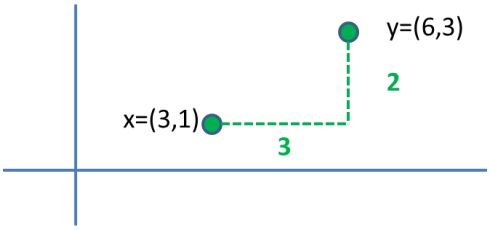
נגדיר לכל $p \geq 0$.

המרחק בין הווקטורים x, y יהיה:

$$||x - y||_p = \left[\sum_{i=1}^{d} |x_i - y_i|^p\right]^{\frac{1}{p}}$$

למשל:

$$\sqrt[p]{3^p + 2^p}$$



מרחק פרשט - Frechet

מה קורה כאשר $p = 0$?

כאשר p נעשה יותר ויותר גדול אז הקואורדינטה נעשית חשובה ו-p אחר שקטן ממנו מוזנח.

כלומר כאשר $p \to \infty$ אז הקואורדינטה הכי גדולה היא הקובעת וכל מה שקטן ממנו לא רלוונטי.

לכן, מרחק פרשט הוא הקואורדינטה עם ההפרש הכי גדול שיש.

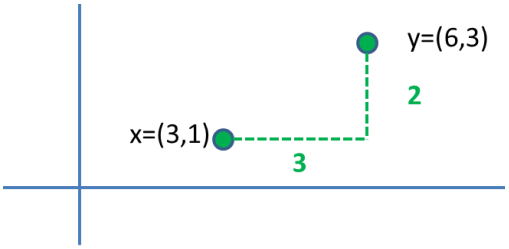
ידוע בתור l_∞.

המרחק בין הוקטורים x, y הוא:

$$||x - y||_\infty = \left[\sum_{i=1}^{d} |x_i - y_i|^\infty\right]^{\frac{1}{\infty}} = \max_i |x_i - y_i|$$

למשל:

$$||x - y||_\infty = max(3, 2) = 3$$



מסקנה

נשים לב עבור הדוגמה של $(3,2)$ קיבלנו:

- עבור l_1 מרחק מנהטן קיבלנו את המרחק 5.
- עבור l_2 מרחק אוקלידי קיבלנו את המרחק 3.6.
- עבור l_∞ מרחק פרשט קיבלנו את המרחק 3.

כלומר, ככל שנשאיף את $p \to \infty$ אז המרחק שלנו יקטן.

השוואת מרחבים של l_p

כיצד משתווים הנורמות השונות של l_p?

עבור $x = (3,1)$ וגם $y = (6,3)$ נקבל:

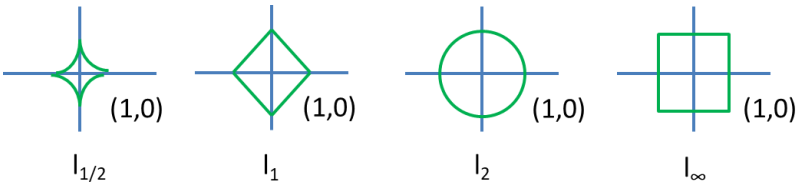
$$||x-y||_\infty = 3 \qquad ||x-y||_2 = 3.6 \qquad ||x-y||_1 = 5$$

עבור $x = (0,0)$ וגם $y = (0,1)$ נקבל:

$$||x-y||_p = 1 \quad \forall p$$

הסבר להקטנת המרחקים:

כדור יחידה תחת ל-p שונים



מרחב של l_p בשברים

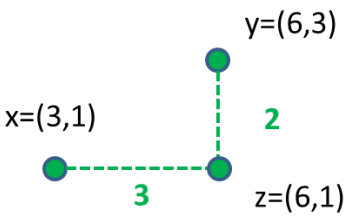
איך מתנהג l_p כאשר $p < 1$?

עבור $x = (3,1),\ y = (6,3),\ z = (6,1)$ נקבל:

$$||x-y||_{\frac{1}{2}} = \left(3^{\frac{1}{2}} + 2^{\frac{1}{2}}\right)^2 \approx 9.9$$

$$||x-z||_{\frac{1}{2}} + ||z-y||_{\frac{1}{2}} = 3 + 2 = 5$$

מפר את אי-שוויון המשולש.



מקרה ה- l_0

המרחק בין הוקטורים x, y הוא:

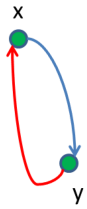
$$||x - y||_0 = \left[\sum_{i=1}^{d} |x_i - y_i|^0\right]^{\frac{1}{0}}$$

בכמה קואורדינטות הוקטורים x, y שונים? - (כלומר כמה מקרים (a, b) שונים יש?):

- אם $x \neq y$ אז כל מספר קבוע שההפרש שלו הוא לא אפס הוא יהיה אחד כלומר $|x - y|^0 = 1$.
  משמש לציון מספר הערכים שאינם אפס בוקטור $x - y$.
- אם $x - y = (1, 1)$ אז:

$$||x - y||_0 = \infty$$

מטריקת מרחק



המטריקה נדרשת:

- סימטריות: $d(x, y) = d(y, x)$.
- אי שוויון המשולש: $d(x, y) \leq d(x, z) + d(z, y)$.

הסמי-מטריקה נדרשת:

- סימטרית בלבד.
- לא מקיימת את אי-שוויון המשולש.

ה-quasi-metric דורשת:

- לא סימטרית.
- מקיימת את אי-שוויון המשולש בלבד.
- כמו למשל המרחק שונה אם עולים על הר או הולכים במישור.

פונקציות מרחק נוספות

להתאים מחרוזת למחרוזת אחרת, המרחק נמדד במספר הצעדים הדרושים כדי להתאים את המחרוזות.
למשל ה-Autocomplete שיש ב-Google Search, עבור המחרוזת "Hello World" אפשר לבצע:

- מחיקה: "Helo World"
- הכנסה: "otHello World"
- החלפה: "Hallo World"

מרחק לוינשטיין

נקרא גם edit distance.

וקטור המרחקים (l_p) לא יכול להתמודד עם מקרים של הכנסות ומחיקות.

למשל:

x=(1,2,3,4,5,7,8,9,10) y=(0,1,2,3,4,5,6,7,8,9)

כלומר, בשביל להשוות בין x, y צריך לעשות הכנסה אחת ומחיקה אחת.

אבל $||x - y||_1 = 10$.

מרחק לוינשטיין במחרוזות:

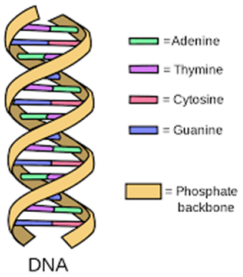
נשאל - כמה הוספות, מחיקות והחלפות צריך לעשות כדי לעבור ממחרוזת x למחרוזת y?

- הוספה ומחיקה עולה לנו ב-1.
- החלפה עולה 1 או 2 (כלומר כמו הוספה + מחיקה).

לדוגמה, עבור מרחק לבנשטיין הוא מחרוזת ה-**DNA**.

יש לו 4 חומצות אמינו בסיסיות:

$$Adenine, Cytosine, Guanine, Thymine$$



בכל פעם שה-DNA מחליף למחרוזת חדשה, יכולים להיות שגיאות למשל:

למשל אם נרצה לעבור ממחרוזת $GGCTAG$ ל-$GCGCAA$.

מה המרחק בין המחרוזות האלה?

GGCTAG → GCGCTAG → GCGCAG → GCGCAA

כלומר המרחק לוינשטיין בין המחרוזות הוא שלוש כי:

הכנסה + מחיקה+ החלפה = 3

ניתן לחישוב בתכנות דינאמי:

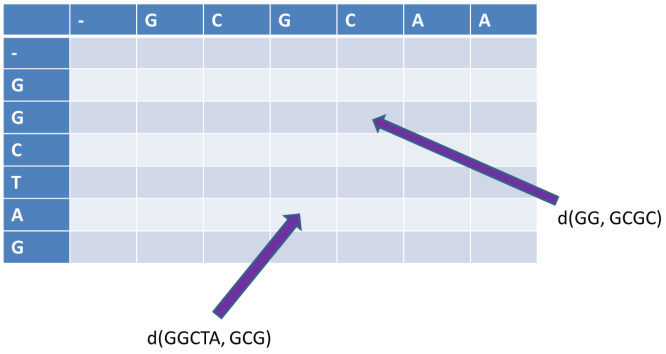
מחשב למעשה מרחקים בין כל הקידומת של s,t

זמן הריצה:

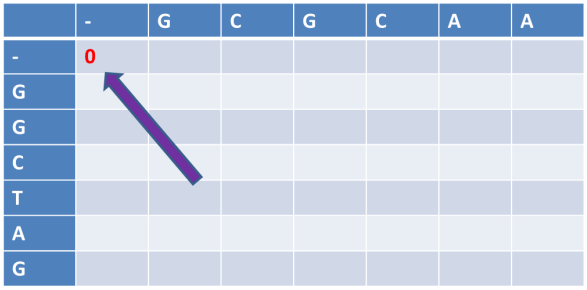
$$O(|S| \cdot |T|)$$

מרחק לוינשטיין בתכנות דינאמי

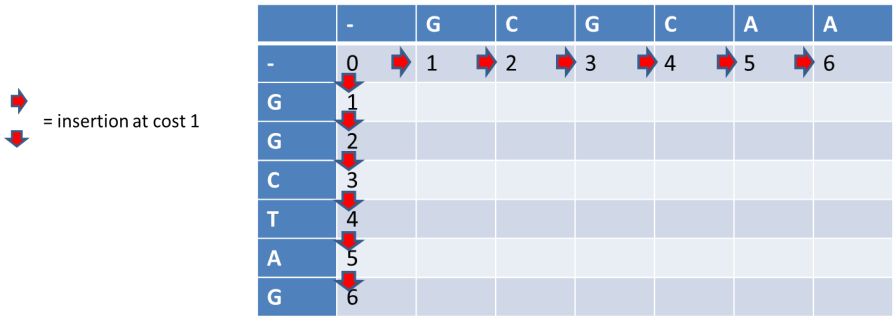
נעזר בדוגמה של ה-DNA שהוצגה מקודם, למשל אם נרצה לעבור ממחרוזת $GGCTAG$ ל-$GCGCAA$.
ראינו שהמרחק הוא 3, נציג בטבלה / מערך דו-מימדי את המחרוזות ונפעל בתכנות דינאמי:



המרחק בין 2 מחרוזות ריקות הוא 0 כי הם שוות.



הוספת השורה הראשונה זה להשוות מחרוזת ריקה למחרוזת שלמה ולכן נבצע n הוספות כאשר n הוא גודל המחרוזת השלמה (גם עבור העמודה הראשונה):



= insertion of G to both at cost 0

אם ב-2 המחרוזות במיקום ה-i יש אותו תו אז הפעולה עולה לנו פשוט 0.
ולאחר מכן, כל תו זר אחר יעלה לנו בהוספה של 1+ בכל תא.

	-	G	C	G	C	A	A
-	0	1	2	3	4	5	6
G	1	0	1	2	3	4	5
G	2						
C	3						
T	4						
A	5						
G	6						

עבור המקרה GC מול GG ראינו בשלב הקודם כי שניהם שווים בתו הראשון וזה עלה לנו 0.
כעת בתו השני אפשר לראות כי הם לא שווים ולכן נרצה להחליף וזה עולה לנו 1+.
וכך הלאה...

	-	G	C	G	C	A	A
-	0	1	2	3	4	5	6
G	1	0	1	2	3	4	5
G	2	1	1	1	2	3	4
C	3						
T	4						
A	5						
G	6						

	-	G	C	G	C	A	A
-	0	1	2	3	4	5	6
G	1	0	1	2	3	4	5
G	2	1	1	1	2	3	4
C	3	2	1	2	1	2	3
T	4						
A	5						
G	6						

	-	G	C	G	C	A	A
-	0	1	2	3	4	5	6
G	1	0	1	2	3	4	5
G	2	1	1	1	2	3	4
C	3	2	1	2	1	2	3
T	4	3	2	2	2	2	3
A	5						
G	6						

	-	G	C	G	C	A	A
-	0	1	2	3	4	5	6
G	1	0	1	2	3	4	5
G	2	1	1	1	2	3	4
C	3	2	1	2	1	2	3
T	4	3	2	2	2	2	3
A	5	4	3	3	3	2	2
G	6						

	-	G	C	G	C	A	A
-	0	1	2	3	4	5	6
G	1	0	1	2	3	4	5
G	2	1	1	1	2	3	4
C	3	2	1	2	1	2	3
T	4	3	2	2	2	2	3
A	5	4	3	3	3	2	2
G	6	5	4	3	4	3	3

Final result

נציג את אלגוריתם לוינשטיין:

Levenshtein Algorithm

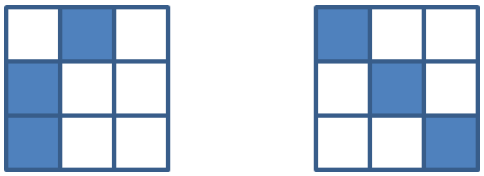
```
1:  Levenshtein(String s, String t)
2:        int[][] a = new array[s.length()+1][t.length()+1]
3:        int count_s = 0, count_t = 0
4:        for (int i=0; i ≤ s.length(); i++)  a[i][0] = count_s++  \\ first column
5:        for (int j=0; j ≤ t.length(); j++)  a[0][j] = count_t++  \\ first row
6:
7:        for (int i=1; i ≤ s.length(); i++)
8:              for (int j=1; j ≤ t.length(); j++)
9:                    if(s.charAt(i-1) == t.charAt(j-1))
10:                         a[i][j] = min(a[i-1][j-1], a[i][j-1]+1, a[i-1][j]+1)
11:                   else  \\ substitute cost = 1
12:                         a[i][j]=min(a[i-1][j-1]+1, a[i][j-1]+1, a[i-1][j]+1)
13: return a[s.length()][t.length()]
```

מדידת מרחק בין תמונות (מרחק EMD)

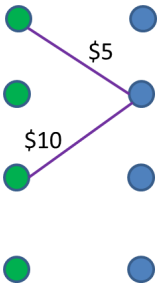
איך אנחנו מודדים מרחקים בין תמונות?

עבור מרחק בין וקטורים לא נגיע לפתרון - נופל כאשר התמונות זזות בשורת פיקסל אחד.

נפעל בשיטת ה-EMD כאשר הרעיון הכללי הוא להגדיר את העלות להעביר פיקסלים מתמונה אחת לאחרת.



העלות הוא המדד למרחק וצריך להגדיר את העלות של תנועה אנכית, אופקית ואלכסונית.



בעיית ההשמה

קלט: N עובדים, N משימות ומטריצה N*N המתארת את העלות של כל עובד לבצע את המשימה.

פלט: העלות המינימלית הכוללת.

כדי לפתור את סך ההתאמות המינימליות נשתמש **באלגוריתם Hungarian**.

דוגמה

עבור וקטורים ממימד 1 לפי EMD, נחשב למשל את המרחק מ- $x = (3, 4, 7, 1)$ ל- $y = (1, 5, 5, 4)$.

נפעל בשיטה הבאה, עבור וקטור x:

$\Rightarrow$ x = (3, 4, 7, 1)

נוריד **2** מהתא הראשון ונוסיף אותו לתא השני:

$\Rightarrow$ (1, 6, 7, 1) $\Rightarrow$

נוריד **1** מהתא השני ונוסיף אותו לתא השלישי:

$\Rightarrow$ (1, 5, 8, 1) $\Rightarrow$

נוריד **3** מהתא השלישי ונוסיף אותו לתא הרביעי:

$\Rightarrow$ (1, 5, 5, 4) = y

והגענו מ-x ל-y כך שהעלות הכוללת היא:

2 + **1** + **3** = **6**

קצת על Hungarian Algorithm - (לא נלמד בהרצאה)

האלגוריתם ההונגרי הוא אלגוריתם אופטימיזציה שפותר את בעיית ההשמה בזמן פולינומי.

בניסוח כמטריצה אנו מקבלים מטריצת nxn לא שלילית, כאשר האלמנט בשורה i ועמודה j מייצג את עלות הקצאת משימה j לעובד i.

המטרה היא לשבץ את העובדים למשימות כשלכל משימה משובץ עובד אחד וכל עובד משובץ למשימה אחת, כך שהעלות הכוללת של ביצוע כלל המשימות היא מינימלית.

לדוגמה:

בדוגמה פשוטה זו ישנם שלושה אחים: אפרים, מאיר וישראל. אחד מהם צריך לנקות את חדר השינה, שני לטאטא את החצר ושלישי לשטוף את החלונות. כל אחד מהם דורש תשלום שונה עבור כל משימה. המטרה היא למצוא את השיבוץ שייתן את העלות הנמוכה ביותר לביצוע שלוש המשימות. את הבעיה ניתן לייצג במטריצה של עלויות העובדים לביצוע המשימות: ←

	לנקות חדר	לטאטא חצר	לשטוף חלונות
אפרים	2	3	3
מאיר	3	2	3
ישראל	3	3	2

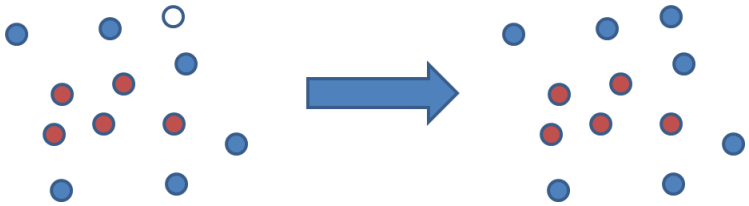
יישום האלגוריתם ההונגרי על המטריצה ייתן את העלות המינימלית שהיא **6**, עלות זו מושגת על ידי שיבוצו של אפריים לניקוי החדר, של מאיר לטיאטוא החצר ושל ישראל לשטיפת החלונות.

סיווג שכן קרוב

איך אפשר ללמוד במרחבים לא אוקלידים?

- ניקח קבוצת בסיס (**Base**).
- לכל נקודה חדשה, ניתן לו את אותו התיוג כמו שיש לשכן הכי קרוב אליו מקבוצת הבסיס.

למשל באיור הבא, הנקודה הלבנה החדשה שנכנסה למדגם תסווג לכחול כי השכן הכי קרוב אליו גם כחול:



בעיה

במקרה הזה $VC_{Dim} = \infty$.

הוכחה

עבור כל קבוצה של נקודות שניתן, אפשר להשיג בשבילם כל תיוג אפשרי, כי יכול להיות באוסף הבסיס אותם נקודות עם איזה תיוג שנרצה וגם לאוסף אחר ניתן איזה תיוג שנרצה ולכן יכול להיות במדגם שלנו אותם נקודות עם אותו תיוג.

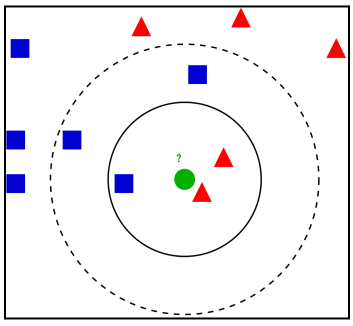
נציג 2 פתרונות לבעיה הזאת:

- k-NN
- Condensing (עיבוי).

שיטת k-NN

ניקח את K השכנים הקרובים ביותר לנקודה החדשה ונסווג את הנקודה לפי השכנים עם הרוב של אותו סיווג.

- זה טוב למניעת רעשים.



לדוגמה, **נקודה חדשה** במדגם שלנו (ירוק) ועבור $k = 3$,
נחפש את את k השכנים הקרובים ביותר לנקודה החדשה.
נסווג את הנקודה החדשה לפי הסיווג המשותף הגדול ביותר בין השכנים.
ולכן, הנקודה הירוקה החדשה תסווג להיות **משולש אדום** כמו רוב שכניה.

שיטת העיבוי - Condensing

מספר האיברים בקבוצת הבסיס Base חסומה על ידי **s** כלשהו.
נבצע NN בקבוצת הבסיס.
לדוגמה: עבור מדגם S בגודל n, נשמור לנו תת-מדגם $S' \subset S$ שיהיה קבוצת הבסיס מגודל s.

לכן, עבור מדגם בגודל n מתקיים:

- מספר התיוגים הוא $O(n^s)$
- $VC_{Dim} = O(s)$

<u>הסבר יותר מובן</u> - (הסבר לא מההרצאה):
אלגוריתם Condensed Nearest Neighbors או CNN עוזר לצמצם את מערך הנתונים S עבור סיווג k-NN.
הוא בונה תת-קבוצה S' של דגימות המסוגלות לסווג נכון את מערך הנתונים המקורי S.
הוא מחזיר לא את מערך הנקודות שסווגו בצורה שגויה, אלא תת-קבוצה S' של מערך הנתונים S.

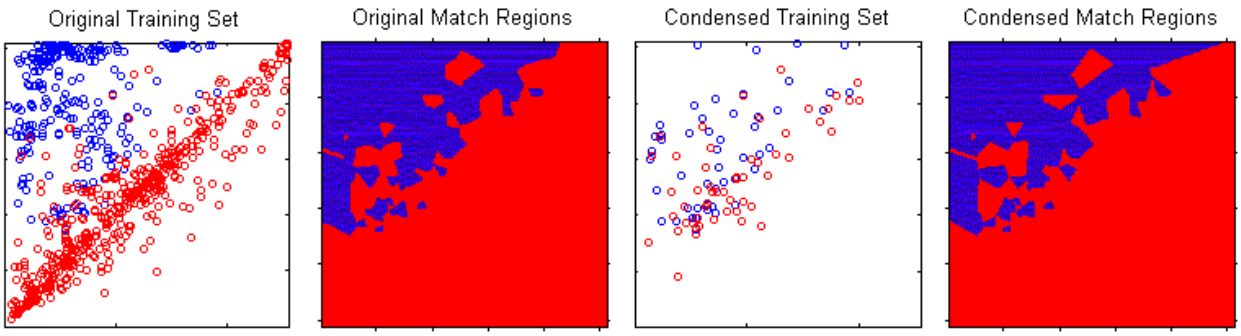
אלגוריתם CNN עובד כך:

1. סרוק את כל הנקודות של S, וחפש נקודה p שלשכן הקרוב שלו מ-S' יש תיוג שונה מזו של p.
2. הסר את p מ-S והוסף אותו ל-S'.
3. חזור על הסריקה עד שלא יתווספו עוד ל-S'.

וכעת S' בשימוש במקום S עבור סיווג k-NN.
נשים לב ש $|S'| = s$ וגם $|S| = n$ כאשר $n > s$ ולכן היתרון בשיטה זו הוא קיצור זמן-ביצוע והפחתת הנפח.

לפי האיור הבא:

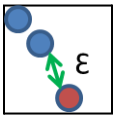
- קבוצת ה-Original Training Set כלומר הקבוצה S הוא המדגם כולו.
- תת-הקבוצה Condensed Training Set כלומר הקבוצה S' הוא מדגם חלקי שנוצר לפי השלבים.
- הפלט של שתי הקבוצות האלה הוא אותו הפלט.
- לכן, ברור שניקח את הקבוצה S' שתחסוך לנו זמן חישוב יקר ותחסוך לנו בנפח מקום זיכרון.



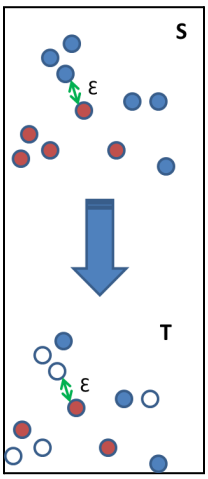
דחיסת מדגם - CNN

נרחיב על אלגוריתם Condensing Nearest Neighbor - CNN.

1. בהינתן קבוצת מדגם S, יהא ε המרווח בין הנקודות (המרחק המינימלי בין 2 נקודות בצבעים שונים). נרצה לזרוק הרבה נקודות ורק לשמור כמה מהם.



2. ניקח מ-S רשת (נקרא לו $net - \varepsilon$) הוא תת קבוצה T כך ש:
   a. אריזה: לכל $p, q \in T$ המרחק שלהם $d(p,q) \geq \varepsilon$.
   b. כיסוי: לכל $p \in S$ המרחק שלו מהשכן הכי קרוב אליו ב-T הוא $d(p,T) < \varepsilon$.
   c. בנייה חמדנית:
      i. נאתחל $T = \emptyset$.
      ii. לכל $p \in S$, אם $d(p,T) > \varepsilon$ אז נוסיף את p ל-T.



כל 2 נקודות שלקחנו ל-T המרחק ביניהם הוא לפחות ε.

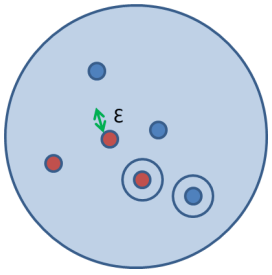
3. ניקח T כקבוצת הבסיס עבור השכנים הקרובים.
   a. אז $NN - 1$ עקבי על S ללא שום טעות אמפירית.
   b. כמה גדול T ומה ה-$VC_{Dim}(T) =$?.

מה הגודל של T?

עבור כדור ברדיוס של $R = 1$ המכיל את כל הנקודות.

הנקודות בקבוצה T הם במרחקים גדולים יותר מ- $\varepsilon < 1$.

לכן, נגדיר כדור מעטפת סביב כל נקודה ב-T, כדורים ברדיוס $R = \frac{\varepsilon}{2}$ מסביב לנקודות של T אינם מצטלבים.



הנפח של כדור d-מימדי ברדיוס R הוא:

$$Vol_d(R) = \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)} \cdot R^d$$

$\Gamma(k)$ הוא פונקציית הגמא של אוילר, ושווה ל- $(k-1)!$ לכל k חיובי שלם.

מאחר ו-R לא תלות נגדיר $R = 1$.

לכן, הגודל של T חסום:

$$T < \frac{Vol_d(1)}{Vol_d(\frac{\varepsilon}{2})} = \left(\frac{2}{\varepsilon}\right)^d$$

עצי החלטה

מבוא (לא מההרצאה)

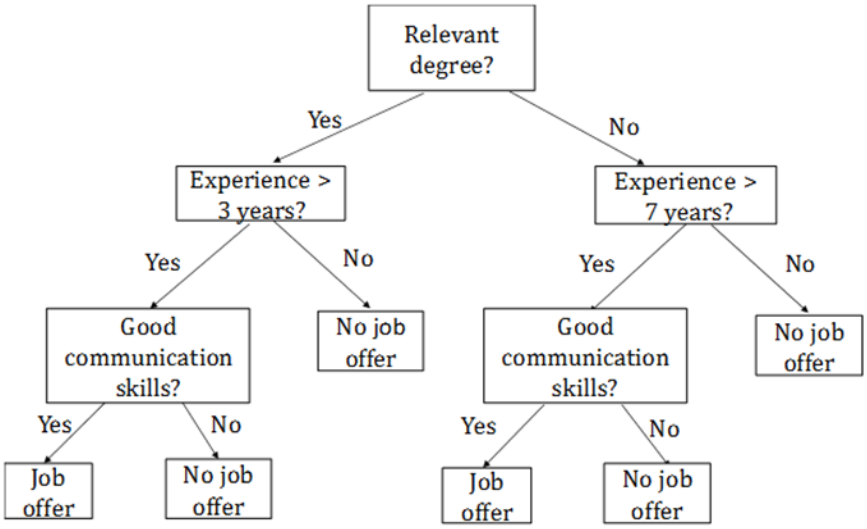
עצי החלטה הוא אלגוריתם חיזוי השייך לתחום הלמידה

- עצי החלטה תואמים לאופן שבו מרבית בני האדם חושבים על בעיה מסוימת והם פשוטים להסבר למי שאינם מומחים.
- אין שום דרישה שהקשר בין היעד לבין המאפיינים יהיה לינארי.
- העץ בוחר אוטומטית במאפיינים הטובים ביותר על מנת לבצע את החיזוי.
- עץ החלטה רגיש פחות לתצפיות חריגות.

עץ החלטה מציג תהליך של שלב-אחר-שלב לביצוע חיזוי. לשם הפשטות נתחיל עם דוגמא פשוטה הנוגעת לסיווג מועמדים לתפקיד מסוים לשתי קטגוריות:

1. כאלו שצריכים לקבל הצעת עבודה (job offer);
2. כאלו שצריכים לומר להם "תודה אבל לא תודה" (no job offer).

הדוגמא מתארת את אחד המאפיינים העיקריים של עצי החלטה. ההחלטה מתקבלת על ידי הבאה בחשבון של המאפיינים אחד בכל פעם במקום כולם בבת אחד. תחילה, נלקח בחשבון המאפיין החשוב ביותר (תואר), לאחר מכן נלקחים שנות הניסיון, וכך הלאה. להלן דוגמה לעץ החלטה לקביעת קריטריון להעסקת עובד:



חשוב להבין שמעסיק יכול בתת-מודע שלו להשתמש בעץ החלטה שכזה, מבלי לרשום אותו.

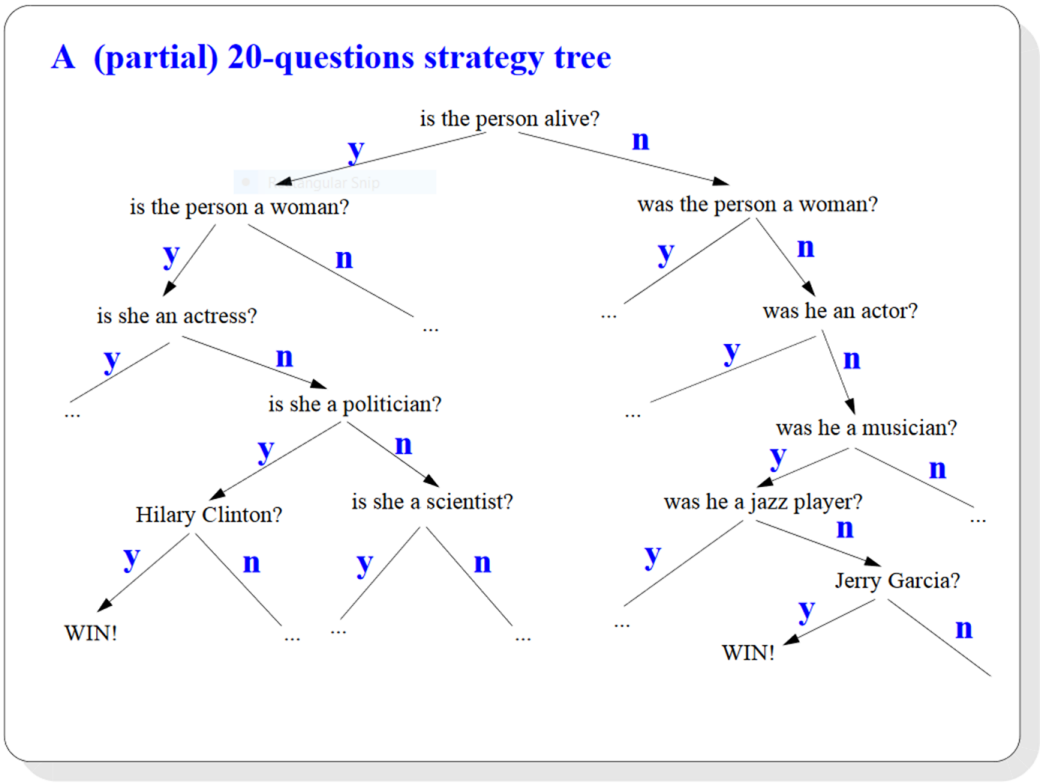
כאשר עצי החלטה משמשים ככלי של למידת מכונה, אז העץ נבנה מתוך נתונים היסטוריים באמצעות אלגוריתם, כפי שנסביר בהמשך.

אם לאחר חישוב העץ ניתן לראות כי ברור שקיים רווח מהמידע האם למועמד מסוים יש תואר רלוונטי או אם לאו, אז ברור שנרצה לקצר את דרך ההחלטה בעץ עבור מועמד מסוג זה.

את הדרך לעשות זאת נפתור בעזרת פונקצית האנטרופיה הנפוצה בעולם דחיסת נתונים.

שאלה

מרצה חושב על אישיות מוכרת ומבקש מהסטודנטים לתת עד 20 ניחושים מי הבן אדם הזה.
המרצה יכול לתת רק תשובה של "כן/לא" לכל ניחוש.



מספר העלים בעץ בינארי עם 20 רמות הוא נורא גבוהה.

כלומר מדובר ב: $2^{20} = 1,048,576$ עלים.

יש לנו את היכולת לנצח תמיד - אך השאלה היא מהי הגישה הנכונה בשביל שננצח כל פעם?

אסטרטגיה שתמיד תנצח:

יהיה לנו רשימה של כמה אנשים מוכרים בעולם (כמה עשרות אלפים שכאלה), נעשה חיפוש בינארי על השמות של האנשים.

נרצה למזער את מספר הניחושים, ההסתברות שאני חושב על בן אדם מסוים X_I הוא $\Pr(X_i)$ ואנחנו נרצה שמספר הניחושים $number\_of\_guesses[X_i]$ יהיה נמוך כלומר:

$$\Pr(X_i) \cdot number\_of\_guesses[X_i]$$

Pr  > Pr 

שאלה

כמה אפשר לדחוס קובץ?

- **תורת האינפורמציה** מתעסקת בשאלה הזאת ונותנת תשובה ברורה לכמה זיכרון צריך בשביל זה.

שאנון 1948

הסמל i מופיע בתדירות w_i אז הדחיסה האופטימלית להודעה באורך m היא בגודל של ביטים:

$$m \cdot \sum_i w_i \cdot \left\lceil \log_2\left(\frac{1}{w_i}\right) \right\rceil = \quad - m \cdot \sum_i w_i \cdot \left\lceil \log_2(w_i) \right\rceil$$

הנוסחה הזאת זה חסם תחתון על כמה ביטים צריך לאחסן את המסמך הזה.

<u>לדוגמה</u>

- מסמך באורך $m = 50$
- האות 'a' מופיעה 20 פעמים ו-'b' מופיעה 30 פעמים.
- הדחיסה האפשרית היא:

$$50 \cdot \left(\frac{20}{50} \cdot \log_2\left(\frac{50}{20}\right) + \frac{30}{50} \cdot \log_2\left(\frac{50}{30}\right)\right) = 48.54\ bits$$

לפי משפט של שאנון,

- צריך לקחת כל אות ולתת לה ייצוג של ביטים.
- לכל אות אסור לשנות את הייצוג שלה בזמן הדחיסה.

אנטרופיה

האנטרופיה מודדת כמה "מאוזן" חלוקת הערכים במשתנה המקרי.
פונקציית האנטרופיה עבור פריט/מאורע e_i מוגדרת כך:

$$\sum_i \Pr(e_i) \cdot \log_2 \left\lceil \frac{1}{\Pr(e_i)} \right\rceil$$

לדוגמה, למטבע הוגן יש אנטרופיה של 1, ומטבע לא הוגן (למשל תמיד נקבל עץ) אז האנטרופיה שלו היא 0.
לכן, האנטרופיה בעצם מודדת כמה מאוזן החלוקה של התוצאות של המשתנה המקרי.

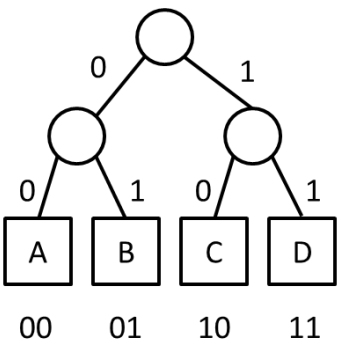
פונקציית האנטרופיה הבינארית לסימנים/מאורעות $e_1,\ e_2$ מוגדרת כך:

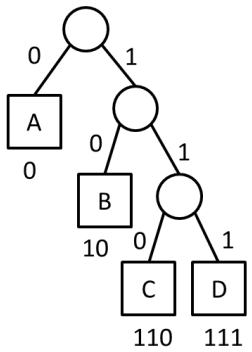
$$\Pr(e_1) \cdot \log_2\left(\frac{1}{\Pr(e_1)}\right) + \Pr(e_2) \cdot \log_2\left(\frac{1}{\Pr(e_2)}\right) =$$
$$= \Pr(e_1) \cdot \log_2\left(\frac{1}{\Pr(e_1)}\right) + \left(1 - \Pr(e_1)\right) \cdot \log_2\left(\frac{1}{1-\Pr(e_1)}\right)$$

שאנון אמר שצריך מספר ביטים מסוים כדי לאחסן את המשפט.
אך נרצה לדעת איזה סוג של דחיסה נרצה להשיג כדי לקיים את זה.
ארבע שנים אחרי המצאת שאנון, המציאו טכניקה להשיג את החסם של שאנון שנקרא הופמן (למדנו באלגוריתמים ובדחיסת נתונים).

אלגוריתם האפמן

הוא אמר שאם אנחנו מתחילים את הקידוד מלמטה (של העץ) ללמעלה אז אנחנו נקבל את הקידוד האופטימלי.
למשל עבור הדוגמה של העץ השמאלי: $BAABAC = 010000010010$
ועבור הדוגמה של העץ הימני: $BAABAC = 1000100110$





נשים לב שבעץ השמאלי הצלחנו לשחזר את ההודעה שלנו באורך של 12 ביטים.
נשים לב שבעץ הימני הצלחנו גם לשחזר את ההודעה שלנו באורך של 10 ביטים.
לכן אפשר לראות כי ניתן לקודד ולפענח במספר ביטים קטן יותר אם נפעל בשיטה המתאימה.

<u>דוגמה</u>

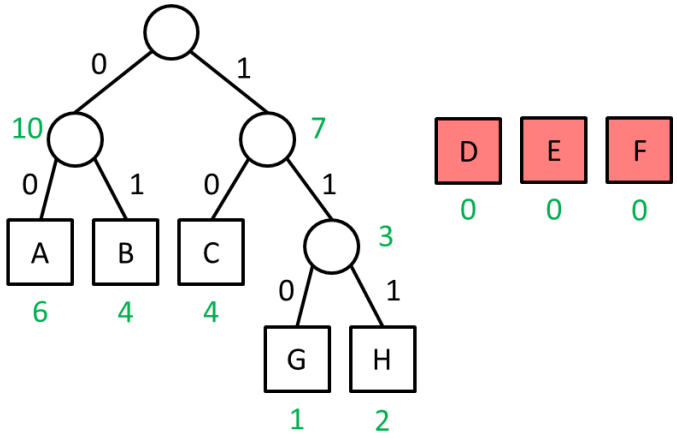
נתונים האותיות (סימנים): $A,\ B,\ C,\ D,\ E,\ F,\ G,\ H$.
ההודעה: $AAAAABBAHHBCBGCCC$
ניתן לבנות עץ מאוזן לקידוד.

- 8 סימנים.
- כל אחד מקודד ל-3 ביטים.
- גודל ההודעה יהיה $3 \cdot 17 = 51$

נשים לב שניתן לשפר את זה לפי התייחסות למספר המופעים של כל אות בהודעה שלנו:

$A: 6,\ B: 4,\ C: 4,\ D: 0,\ E: 0,\ F: 0,\ G: 1,\ H: 2$

אנחנו נבנה עץ מלמטה ללמעלה לפי סדר משקלי התדירויות של כל אות בהתאם:
אז את ההודעה $AAAAABBAHHBCBGCCC$
נקודד ל: $37\ bits$ – 0000000000010100111111011001110101010
ונקבל:

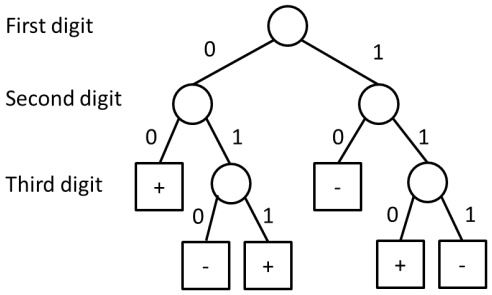


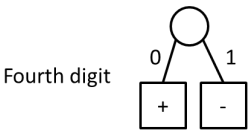
מה היתרון בבניה כזאת? - נשים לב שמאחר ו-A היה הכי נפוץ, נרצה לגשת אליו בזמן הכי מהיר מן הסתם ולכן הרמה של A תיהיה כמה שיותר קרובה לשורש בעץ כדי שנוכל לגשת אליו באופן יעיל ככל האפשר.

למידה עם עצי החלטה

0000	+
0010	+
0101	-
0110	+
1001	-
1011	-
1100	+
1111	-

נרצה ללמוד עם עצי החלטה.
לדוגמה: דגימה של וקטורים בינארים (נראה בצד שמאל).
עבורו נציג שני עצי החלטה שונים:





נשים לב שבאופן אינטואיטיבי, נרצה לבחור בעץ הימני ולא בשמאלי.
העץ הימני בעצם אומר לנו שאם בספרה הרביעית שלנו היא 0 אז נקבל + ואחרת נקבל -.
מן הסתם, שנרצה עץ החלטה קל ומהיר כמו הימני, לכן, כדי להגיע למצב שכזה, נרצה למצוא עץ החלטה עם מספר k הצמתים הקטן ביותר (בדומה לעץ הימני בדוגמה הזאת).

<u>מה ה-VC_{Dim} של עץ החלטה בוקטור בינארי מ-d מימדים?</u>

נסמן $T(k)$ - משפחה על עצי החלטה עם k צמתים.
בכל צומת, אני או בוחר איזה קואורדינטה לבדוק או שזה עלה (ולבדוק האם זה + או -) לכן $d + 2$ החלטות.
לכן מספר העצים חסום כך:

$$T(k) \leq (d + 2)^k$$

לפי הלמה של סאואר:

$$VC_{Dim}(T(k)) = O(k \cdot \log(d))$$

אנחנו נרצה למצוא חסם עבור מספר הצמתים k, כלומר נרצה מספר k קטן ככל הניתן.

בעיה

צריך למצוא עץ החלטה שעקבי על הנתונים שלנו ובעל מספר הצמתים הקטן ביותר.

- זו בעיה $NP - Hard$.

לכן, נשתמש בהיוריסטיקות ב-2 שיטות:

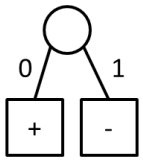
1. שיטת $ID3$.
2. שיטת $Pruning$.

אלגוריתם ID3

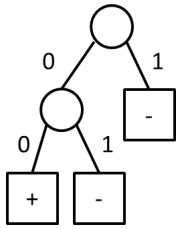
זה אלגוריתם שפועל מלמעלה למטה.

1. נתחיל עם השורש כעלה שמסומן ב-'+'.

2. פונקציית העלות **Cost** מודדת את איכות הפתרון.
3. נפצל את העלה עם חוק שממזער את **Cost**.



4. נחזור על השלבים עד שנגיע למצב של עקביות על הנתונים.



*<u>איך נגדיר פונקצית **Cost** טובה?</u>*

- נמזער את סכום האנטרופיה של העלים.
- בהינתן קבוצה S, נבחר חוק שמפצל את S ל- $A, B \subseteq S$ שממזער בצורה הבאה:

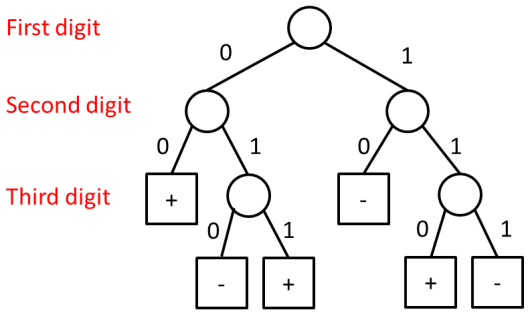
$$\sum_{e_i \in A} \Pr(e_i) \cdot \log_2\left(\frac{1}{\Pr(e_i)}\right) + \sum_{e_i \in B} \Pr(e_i) \cdot \log_2\left(\frac{1}{\Pr(e_i)}\right)$$

נזכר שהאנטרופיה נמוכה כאשר "המטבע לא הוגן - כלומר שנקבל רק עץ", לכן, נרצה דווקא את האנטרופיה הנמוכה ביותר, כלומר בקבוצה A יהיו לנו למשל "רק +" וב-B למשל "רק -".
אז נרצה למצוא חוק שכזה שיפצל באופן הכי טוב שאפשר ולבסוף אם סכום האנטרופיה של העצים נמוך, זה פשוט אומר שהצלחנו להפריד בין התיוגים באופן טוב.

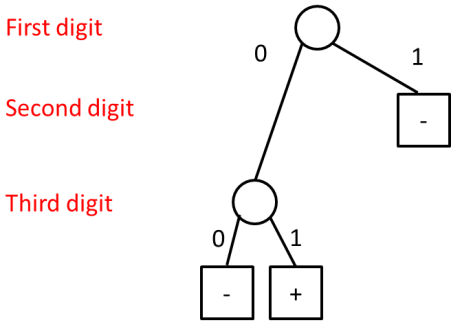
אלגוריתם Pruning

דרך נוספת להשיג עץ קטן - לא עקבי. אפשר לעשות את זה אחרי brute-force או ID3.
באלגוריתם זה לא בהכרח נקבל משהו עקבי (גם ID3 ככה, אך ב-Pruning זה יותר משמעותי).
האלגוריתם מקבל עץ ומתחיל לגזום אותו מלמטה ויש לו כמה אפשרויות:

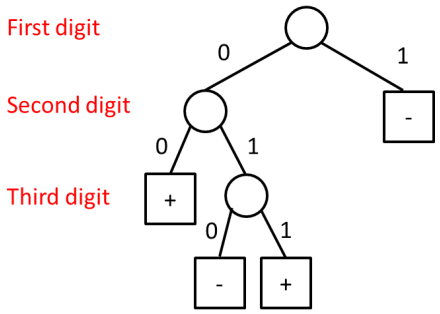
1. לא לעשות כלום



2. לקחת צומת פנימית ולהפוך אותו לעלה ב-$\{0, 1\}$.



3. להחליף אותו עם תת-עץ אחר שקיים כבר בעץ.



יער אקראי

הטכניקה הכי חזקה בעצי החלטה עבור וקטורים.

- בונים הרבה עצי החלטה שונים למשל עם ID3 יחד עם Pruning.
- שואל כל עץ החלטה איך הוא ביחס לוקטור מסוים.

מקבוץ - Clustering

מבוא (לא נלמד בהרצאה)

מקבוץ או **ניתוח אשכולות** מתייחס למשימה של קיבוץ אובייקטים לקבוצות כך שהאובייקטים הנמצאים באותה קבוצה דומים זה לזה יותר מאשר לאובייקטים השייכים לקבוצות אחרות.

אשכול (Cluster) - אוסף של אובייקטים, כאשר מה שמאפיין את האובייקטים הנמצאים באותו אשכול – זה שיש דמיון רב ביניהם, והם שונים מהאובייקטים הנמצאים באשכולות אחרים.

מטרה של הניתוח - אנו מנסים למצוא דמיון בין האובייקטים (פיזיים או אבסטרקטיים), בהתאם למאפיינים של אותם אובייקטים. אז *נקבץ* את האובייקטים הדומים לאותו אשכול, כאשר אובייקטים שונים ישולחו לאשכולות שונים.

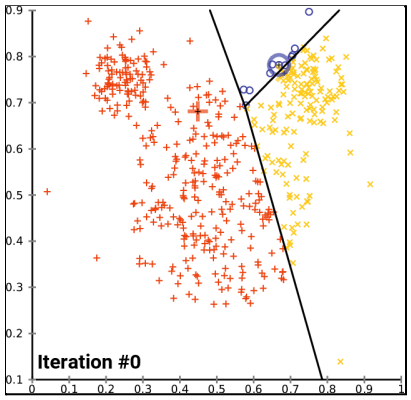
יישומים לניתוח - באופן כללי, יש 2 אפשרויות:

1. אנו משתמשים בניתוח כדי להבין טוב יותר את הנתונים שיש לנו.
2. אנו משתמשים בניתוח כשלב מקדים לאלגוריתמים אחרים (למשל, לאחר החלוקה, אנו יכולים להפעיל אלגוריתם של סיווג על-מנת להבין מה מאפיין כל אשכול ואשכול).

אלגוריתם K-Means -

אנו לוקחים בסיס הנתונים שיש בו n אובייקטים (או רשומות) ומחלקים אובייקטים אלה ל-k אשכולות.
שואפים להביא למינימום את סכום ריבועי הסטיות, כאשר בתוך הסכום אנו מחשבים עבור כל אשכול ואשכול את סכום הריבועים של המרחקים בין מרכז הכובד של אשכול (centroid) לבין אובייקטים שנמצאים בתוך האשכול.
זה אומר שאנו שואפים למצוא את החלוקה שתמקם את האובייקטים דומים (למרכז, ואחד לשני) בתוך אותו אשכול. ולעומת זאת, אם אובייקטים רחוקים מדי ממרכז הכובד של האשכול שלהם – זה, כנראה, אומר שצריך לשייך אותם לשכול אחר.

האנימציה משמאל היא דוגמה למספר איטרציות של המודל תוך כדי מזעור המרחקים בתוך האשכול:

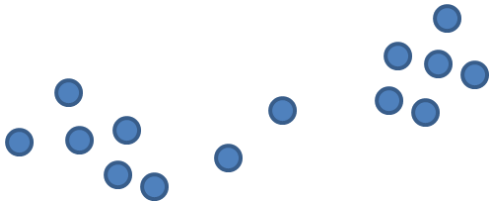


מקבוץ

למידה מפוקחת: מדגם של נקודות עם תיוגים.

למידה לא מפוקחת: ללא תיוגים!

- עדיין אפשרי ללמוד ללא תיוג.
- למשל, למקבץ את הנקודות הבאות ל-2 קבוצות:



מטרות המקבוץ

אנחנו דוגמים מאגר עם תיוג ורוצים בהינתן נקודות בלי תיוג לנחש.

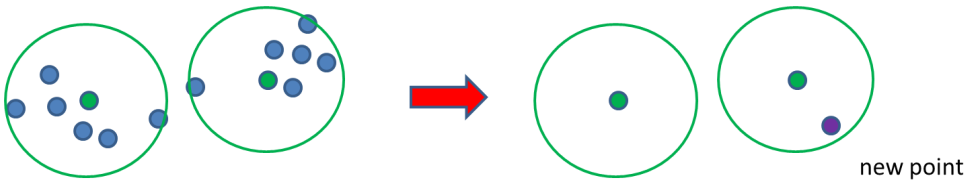
למידה: נקודות חדשות יכולות להשתייך לקבוצה.

דחיסה: יכול להיות שצריך רק לשמור על K, לא S.

כלומר, ניתן לשמור מספר קטן של נקודות המייצגות כל קבוצה.

- חסם הכללה טוב יותר.

זמן ריצה: חיפוש השכנים הקרובים בקבוצת הבסיס K, לא S.



באיור למעלה, בחרנו נקודה שהיא המרכז ועשינו סביבה עיגול, מכאן כל נקודה חדשה שתיכנס תהיה שייכת לקבוצה המוגדרת לפי העיגול בה היא נמצאת.

K-Clustering - חסם הכללה בדחיסה

מה ה- VC_{Dim} ללמידה (סיווג) בעזרת $k - clustering$?

נבחר קבוצה מרכזית K מגודל k מתוך הקבוצה S כאשר כמות האפשרויות הוא בערך $|S|^k$.

אם ניתן לנפץ d נקודות אז:

$$d^k \geq 2^d$$
$$k\log(d) \geq d$$
$$d = O(k\log(k))$$

בסך הכל לקחנו $|S|$ נקודות, סיווגנו אותם לפי k מרכזים ואז בהינתן נקודה חדשה, סיווגנו אותה כמו השכן הקרוב שלה.

אלגוריתמי מקבוץ

בעיה: בהינתן מדגם S עם n נקודות ופרמטר k.

- מקבץ את S ל-k קבוצות.
- שאלה של מידה: מהו מקבוץ טוב?
- מדדים שונים: k-center, k-median, k-means

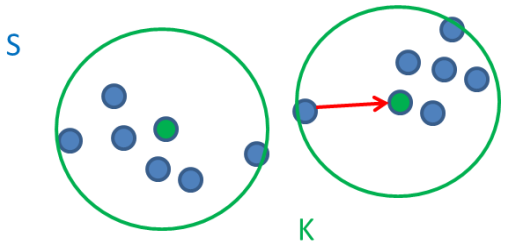
מקבוץ K-Center

בעיה: בהינתן מדגם S עם n נקודות ופרמטר k, צריך לקבץ את S ל-k קבוצות.

מדד: נבחר קבוצה של k מרכזים $K \subset S$ כך שהוא ממזער: $\max_{v \in S} d(v, K)$.

כלומר, נרצה למזער את הנקודות שהכי רחוקות מהמדד שלהן.

המרחק מכל $v \in S$ לנקודה הקרובה ביותר בתת הקבוצה K.



למשל בציור למעלה, הנקודה הכחולה שיוצא ממנה חץ אדום היא הנקודה המשפיעה על המדד כי היא הכי רחוקה מהנקודה הירוקה.

גרסה אחרת של מקבוץ k-center

הצגנו גרסה של מרכזים $K \subset S$, זה הגרסה הדיסקרטית (K הוא תת-קבוצה של S).

כעת נציג גרסה לא-דיסקרטית: מרכזים שנבחרו בסביבת המרחב האוקלידי.

כלומר, זה יהיה כדורים שהמרכז שלהם לא יהיה אחד מן הנקודות.

אלגוריתם מדויק עבור הגרסה הדיסקרטית

- שיטת $Brute\ Force$: ננסה את כל תתי-הקבוצות עם k נקודות מתוך הקבוצה S.
- זמן הריצה יהיה $O(|S|^k)$.
- האם אפשר לעשות את זה יעיל יותר? - בלתי אפשרי.

מציאת מקבוץ $k-center$ אופטימלי היא בעיית $NP-Hard$ כאשר k מספר גדול.
בנוסף, הערכת הרדיוס בתוך הפקטור $2-\varepsilon$ היא בעיית $NP-Hard$.

<u>הוכחה ברדוקציה (עבור שתי הגרסאות)</u>:
רדוקציה מתוך $Minimum\ Dominating\ Set$.
בהינתן גרף $G=(V,E)$, נמצא את תת-קבוצה בגודל המינימלי ביותר $V' \subset V$
כך שלכל נקודה $v \in V-V'$ הוא שכן של נקודה כלשהי מתוך הקבוצה המינימלית V'.
וזו בעיה $NP-Hard$.

אלגוריתמי קירוב עבור k-center

בעיית k-center היא $NP-Hard$ כאשר נרצה לפתור אותה **במדויק**.
ישנם שני אלגוריתמי קירוב בזמן ריצה פולינומי עבור הגרסה הדיסקרטית.

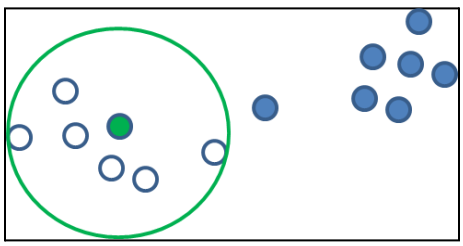
1. רדיוס אופטימלי, אבל $k(\ln(n))$ מרכזים.
   זה $NP-Hard$ כדי לקבל פחות מ-$k(\ln(n))$ מרכזים.
2. מספר k מרכזים, אבל רדיוס גדול.
   זה $NP-Hard$ כדי לעשות יותר מאשר פי שניים מהרדיוס.

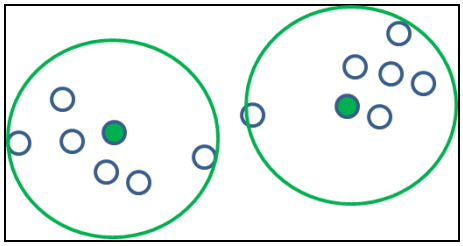
אלגוריתם 1 - חמדני

רדיוס אופטימלי, אבל $k(\ln(n))$ מרכזים.

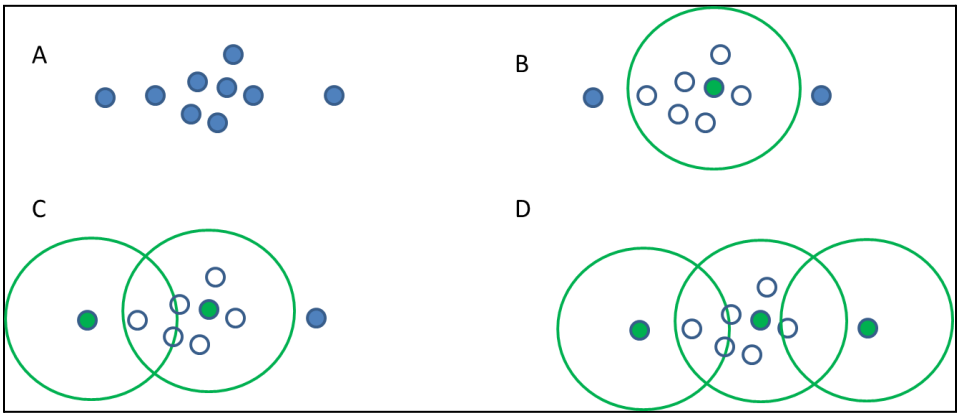
- נניח שהרדיוס האופטימלי r ידוע עם n^2 ניחושים.
- ניקח את המרכז שהכדור שלו מכסה הכי הרבה נקודות.
- נוציא את הנקודות הללו שמכוסות ונמשיך שוב...

בציור הבא, הנקודה הירוקה מכסה הכי הרבה נקודות:





האלגוריתם החמדני הוא לא האופטימלי, יצא לנו 3 מרכזים:



בפועל, היינו יכולים לקבל פחות מ-3 מרכזים.

<u>ניתוח אלגוריתם קירוב חמדני עבור בעיית k-center</u>:

- יש לנו k מרכזים שמכסים את הקבוצה S וגם לכל תת-קבוצה S ישנם מרכזים שמכסים $\frac{1}{k}$ מהנקודות.
- לכל צעד, אנחנו מכסים $\frac{1}{k}$ נקודות.
- לאחר i צעדים, מספר הנקודות שישארו לנו חסום על ידי:

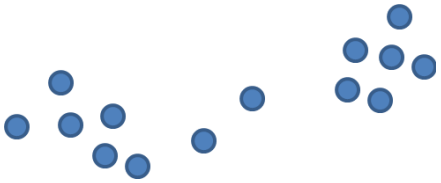
$$n \cdot \left(1 - \frac{1}{k}\right)^i \leq n \cdot e^{-\frac{1}{k}}$$

- ה-i המקסימלי ביותר יהיה $i = k(\ln(n))$ צעדים.
  לכן יהיה לנו $k(\ln(n))$ מרכזים במקום k מרכזים.

אלגוריתם 2 - חמדני

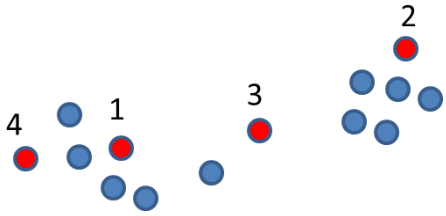
מספר k מרכזים, אבל רדיוס גדול.

נבחר את הנקודה הרחוקה ביותר שלא מכוסה.

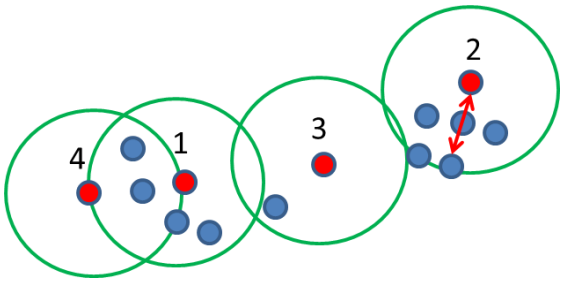


לדוגמה:

$k = 4$ מרכזים.



כעת נבחר את **הרדיוס** המינימלי (האופטימלי):



כלומר, הנקודה שהכי רחוקה מכל אחד מהמרכזים תקבע את הרדיוס.

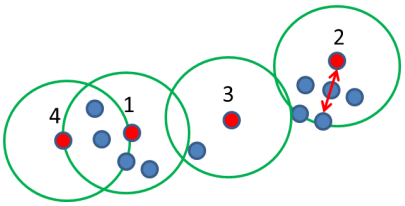
טענה:

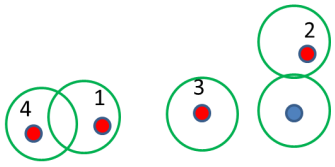
הרדיוס שאנחנו משיגים הוא לכל היותר פי 2 מהרדיוס האופטימלי.

הרדיוס הסופי r הוא לכל היותר $2\bar{r}$ כאשר $\bar{r}$ הוא הרדיוס האופטימלי ביותר.

דרושים לפחות $k + 1$ מעגלים ברדיוס קטן מ-$\frac{r}{2}$ כדי לכסות את כל הנקודות.

לכן, $\frac{r}{2} \leq \bar{r} \leq r$.





למשל במקרה של $k = 4$ אנחנו צריכים לפחות דרושים לפחות $k + 1 = 5$ מעגלים ברדיוס קטן מ-$\frac{r}{2}$ כדי לכסות את כל הנקודות (כפי שאפשר לראות בציור למעלה).

סיכום K-Center

נבחר תת-קבוצה של k מרכזים $K \subset S$ שממזער $\max_{v \in S} d(v, K)$.

המרחק מ-v לנקודה הקרובה ביותר ב-K.

מה הבעיה פה? - לא עמיד בפני חריגים, למשל אולי יש 2 נקודות שממש רחוקות.

מדדים נוספים:

K-Median: נבחר K שממזער $\frac{1}{|S|}\Sigma_{v \in S} d(v, K)$.

כלומר, יש לנו מרכזים וכל מרכז לוקח את הנקודות הכי קרובות אליו וננסה למזער את הממוצע של הנקודות מהמרכז הכי קרוב.

K-Means: נבחר K שממזער $\frac{1}{|S|}\Sigma_{v \in S} d^2(v, K)$.

כלומר, דומה ל-Median רק שזה מרחקים בריבוע ולכן נקודה שהיא קצת יותר רחוקה יש לה השפעה יותר טובה.

מדדי מקבוץ - מרחקים

עבור K-Center זה $max(l_\infty)$, בעזרת מעגלים בממוצע $Voronoi$.

עבור K-Median זה בממוצע (l_1) בעזרת דיאגרמת $Voronoi$.

עבור K-Means זה בממוצע-בריבוע (l_2)בעזרת דיאגרמת $Voronoi$.

מדד K-Median

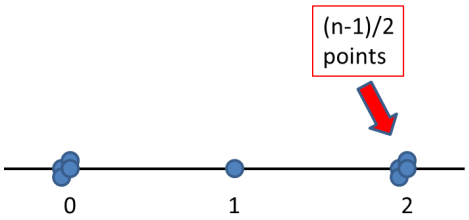
נציג אלגוריתם חמדני עבור K-Median:

1. $K \leftarrow \emptyset$
2. $while\ |K| < k$
3. $Find\ p \in S\ for\ which\ cost(K \cup \{p\})\ is\ minimal$
4. $\quad K \leftarrow K \cup \{p\}$

כלומר, מתחילים עם 0 מרכזים, מוצאים את המרכז שהעלות של המרחקים ממנה לשאר הנקודות היא מינימלית. מוצאים עוד מרכז שעם שניהם נמזער את כל הפתרון שוב וכך הלאה..

טענה

הפתרון של אלגוריתם חמדני הזה ממש גרוע.



הוכחה

יש לנו n נקודות, נבחר $k = 2$.

כלומר העלות היא $\frac{n-1}{2}$ באלגוריתם החמדני הזה.

ובאופטימלי: העלות היא 1.

לדוגמה, לפי הציור משמאל, כמעט חצי מהנקודות נופלות בצד ימין (2) וכמעט חצי בצד שמאל (0) אבל המרכז הכי טוב נמצא בכלל ב-(1).

אלגוריתם חמדני-הפוך עבור K-Median

1. $K_n \leftarrow S$
2. $For\ i = n, \ldots, k+1$
3. $Find\ p \in K_i\ minimizing\ cost\left(K_i \setminus \{p\}\right)$
4. $K_{i-1} \leftarrow K_i \cup \{p\}$

כלומר, נתחיל עם כל הנקודות כמרכזים, נוריד כל פעם את המרכזים כך שנמצא את המרכז שאם נוריד אותו העלות תגדל בהכי מעט שאפשר.

הקדמה

נניח שקיימים הקבוצות $K,\ M$ וגם יהי $Q \subset K$ השכנים הקרובים ביותר של M מ-K.

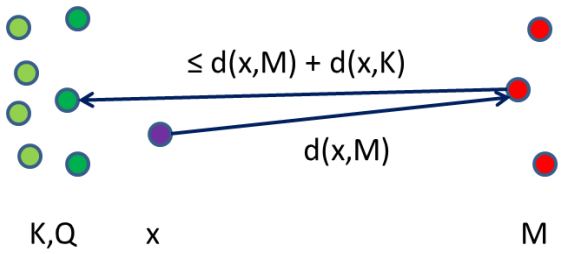
טענה

לכל x מתקיים:

$$d(x,\ Q) \leq 2d(x, M) + d(x, K)$$

הקדמה להוכחה

לפי אי שוויון המשולש:



נרצה לחסום את המרחק מ-x לנקודה הכי קרובה (האדומה) ב-M שזה $d(x,M)$ ומהנקודה האדומה הזאת לנקודה הכי קרובה אליה מהקבוצה Q (ירוק כהה) שזה $d(x,M) + d(x,K)$ וסה"כ נקבל:

$$d(x,M) + d(x,M) + d(x,K) = 2d(x,M) + d(x,K)$$

הוכחה

יהא M הפתרון האופטימלי.

יהא $Q_i \subset X_i$ להיות תת-הקבוצה שהיא אוסף כל הנקודות הקרובות ביותר מ-M ל-K_i.

העלות של הפתרון בכל שלב יהיה:

$$Cost(K_{i-1}) - Cost(K_i) =$$

$$= \left[\min_{r\in K_i} Cost\left(K_i \backslash \{r\}\right)\right] - Cost(K_i) \leq$$

$$\leq \left[\min_{r\in\{K_i \backslash Q_i\}} Cost\left(K_i \backslash \{r\}\right)\right] - Cost(K_i) \leq$$

$$\leq \frac{1}{\left|\{K_i \backslash Q_i\}\right|}\Sigma_{r\in\{K_i \backslash Q_i\}}\left[Cost\left(K_i \backslash \{r\}\right) - Cost\left(K_i\right)\right] \leq$$

$$\leq \frac{1}{i-k}\Sigma_{r\in\{K_i \backslash Q_i\}}\left[Cost\left(K_i \backslash \{r\}\right) - Cost\left(K_i\right)\right] \leq$$

$$\leq \frac{1}{i-k}\left[Cost\left(Q_i\right) - Cost\left(K_i\right)\right] \leq$$

$$\leq \frac{2}{i-k} \cdot Cost(M)$$

שורה 1: העלות של שלב ה-i.

שורה 2: שוויון לפי הגדרה.

שורה 3: נבחר קודקוד שהוא לא r כלומר שהוא לא מקבוצת הנקודות הקרובות ביותר ולכן העלות גדלה.

שורה 4: נבחר לעומת זאת, נקודה שאומנם היא לא הקרובה ביותר אך בממוצע במרחק.

הערה

נשים לב שהסכום של העלויות גדל בסדרה הרמונית:

$$\frac{1}{n} + \frac{1}{n-1} + \ldots = O(\log(n))$$

ולכן יש לנו קירוב של $log(n)$ עבור K-Median.

אלגוריתם Local Search עבור K-Median

זה האלגוריתם הפופולארי ביותר עבור בעיית ה-K-Median.

1. $Start\ with\ arbitrary\ set\ K$
2. $Swap\ some\ p \in K\ with\ some\ q \in S \backslash K\ which\ maximizes\ reduction\ in\ cost$
3. Repeat 2 until no improvement possible

כלומר, נתחיל עם אוסף שרירותי של מרכזים.
נעשה החלפה של מרכז אחד עם נקודה אחת שלא במרכז שאם נחליף אותה, העלות משתפרת.
נעשה שוב עד שנגיע לפתרון כשלא נצטרך עוד החלפות.

ה-Local המינימלי הוא 5-בקירוב. **????**
זמן הריצה הוא בעייתי כי נוכל לעשות Swap עד אינסוף. לכן נוכל לעצור כאשר השיפור כבר לא ממש משתנה.

אלגוריתם K-Means

אנחנו לא נמזער את הממוצע כמו ב-K-Median אלא המרחק בממוצע יהיה ממוצע-בריבוע.
אלגוריתם לא-דיסקרטי k-means שהוא גרסה שונה של $Lloyd's\ Algorithm$.

1. נבחר k מרכזים באופן שרירותי.
2. נייעד כל נקודה ב-S למרכז הקרוב ביותר אליו.
3. לכל מקבוץ C, נחשב את ה-$centroid$ כך ש:

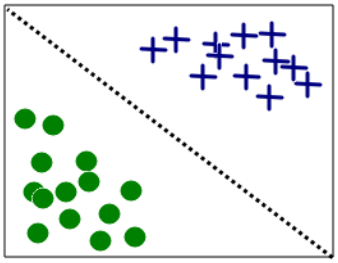
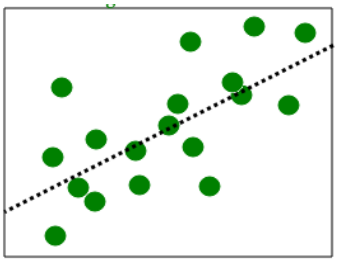
$$centroid = \frac{1}{|C|}\Sigma_{x \in C}\, x$$

4. ה-$centroids$ הם המרכזים החדשים: נייעד לכל נקודה ב-S את ה-$centroid$ הקרוב ביותר.
5. נחזור על שלבים 3 ו-4 עד שלא יהיו עוד שינויים נוספים.

כלומר, נתחיל מאיזשהו אוסף של מרכזים, כל נקודה הולכת לאיזשהו מרכז קרוב.
נמצא את האמצע $centroid$ של כל קבוצה (האמצע זה לא הנקודה שהגדרנו כמרכז) וה-$centroid$ שמצאנו יהפכו להיות המרכזים.
מה האלגוריתם הזה מבטיח לנו?

- מספר האיטרציות במקרה הגרוע הוא $O(2^{\sqrt{n}})$.
  בפועל, המקרה בממוצע הרבה יותר טוב.

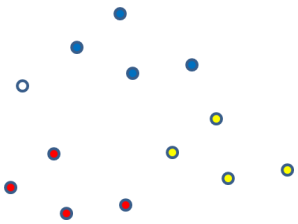
רגרסייה

הבדלים בין קלסיפיקציה לבין רגרסייה		
	קלסיפיקציה	**רגרסייה**
כרוך בחיזוי של	ערכים בדידים	ערכים רציפים
אופי הנתונים החזויים	לא מסודר	מסודר
שיטת חישוב	על ידי מדידת דיוק	על ידי מדידת טעות ממוצע בשורש ריבועי
שאלות לדוגמה	• האם המייל הזה ספאם או לא? • האם זה תמונה של כלב או חתול?	• מה הערך של בית בתל אביב? • מה ההסתברות שהמשתמש ילחץ על המודעה הזאת?
השוואה גרפית		

מולטיקלאס

עד עכשיו, עשינו סיווג בינארי כלומר, 2 מחלקות שונות למשל אדום או כחול.
במולטיקלאס יש לנו K מחלקות שונות למשל אדום, כחול וצהוב.

איך נחשב במקרה שכזה?
אחד הגישות היא להפחית לבעיות בינאריות של אחד מול אחד.



- האם הנקודה החדשה היא כחולה או צהובה?
- האם הנקודה החדשה היא צהובה או אדומה?
- האם הנקודה החדשה היא כחולה או אדומה?

זה נותן לנו רק $O(K)$ תתי-בעיות.

- בעיה שונה: כאשר המאגר לא מחולק בשווה ל-2 מחלקות.
  למשל מחלקות בהם יש 10% עבור '+' וגם 90% עבור '-'.

מה הבעיה כאן? שברגע שנאמן את המאגר הוא תמיד ינחש את '-' עד לרמה של 99.99% אחוזים.

- פתרון אפשרי: איך מתמודדים עם המצב הזה? - אחד הגישות הפופולאריות זה בעזרת הדגימה, אפילו שהמאגר שלנו לא מאוזן, כשנדגום מהמאגר נדגום יותר מה-'+' פי 9 מאשר מה-'-'.

מה זה רגרסיה?

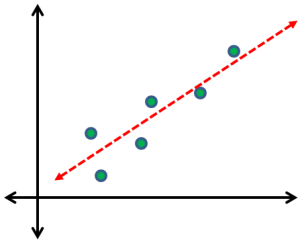
תיוגים בטווחים רציפים.

למשל: $[0,1],\ [-1,1],\ 0-120\ kph,\ 0-200\ kg$

באופן כללי יותר: $\{0,\ M\}$ או $\{-M, M\}$ עבור M כלשהו.

המטרה

בהינתן נקודה חדשה נרצה לדעת מה התיוג הרציף שלה.



בעיה

חיזוי התיוגים לא ידוע עבור נקודות עם תיוגים רציפים, לדוגמה עבור $[0, M]$.

סימונים

פונקציית הטעות לכל $p \geq 1$:

$$L\left(h\left(x_i\right)\right) = \left|y_i - h\left(x_i\right)\right|^p$$

הסיכון האמפירי:

$$\bar{r}(h) = \frac{1}{n}\Sigma_i L\left(h\left(x_i\right)\right)$$

הריסק האמפירי שזה הממוצע של ה-$Loss$ על הנקודות.

בטעות אימפרית בקלסיפיקציה היה הממוצע, כאן שואלים בכמה נקודות טעינו.

הסיכון האמיתי:

$$r(h) = E\left[L\left(h\left(x_i\right)\right)\right]$$

כאשר $x \sim D$.

המטרה: למזער את הסיכון האמיתי.

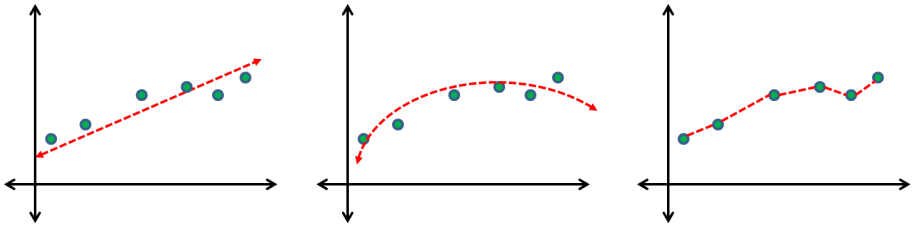
ה-Trade Off ברגרסיה

אנחנו רוצים להתאים פונקציה לנקודות.

אנחנו מחפשים איזשהו חוק שהוא פשוט על המדגם שלנו.

ראינו בסיווג שאפשר לקחת חוק יותר ויותר מסובך שיותר טוב למדגם (כמו ב-AdaBoost).

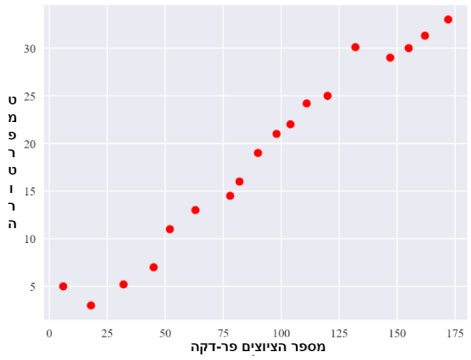
- פונקציה פשוטה נותן מימד VC נמוך.
- פונקציה מורכבת יותר מתאימה לנתונים באופן טוב יותר - אך VC_{Dim} גבוהה.



רגרסיה לינארית

נתחיל מדוגמה (מומלץ לקרוא אותה כדי להבין את ההגדרות הפורמליות לאחר מכן):
זה זמן רב ידוע שצרצרים (זן חרקים) מצייצים בתדירות גבוהה יותר בימים חמים יותר מאשר בימים קרירים יותר.
במשך עשרות שנים מקטלגים מדענים מקצועיים וחובבים נתונים על ציוצים לדקה וטמפרטורה.
כמתנת יום הולדתך, דודתך מעניקה לך את מאגר הצרצרים שלה ומבקשת ממך ללמוד מודל לחיזוי הקשר הזה.
באמצעות נתונים אלה, אתה רוצה לחקור את הקשר הזה.

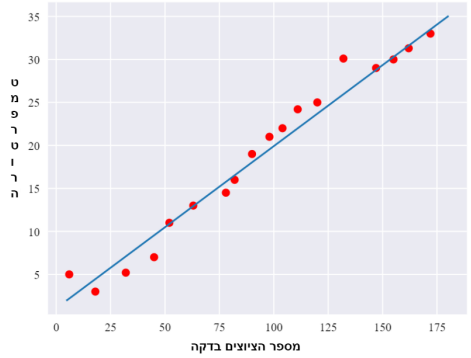
נתבונן בגרף הבא:



]

כצפוי, העלילה מראה את הטמפרטורה עולה עם מספר הציוצים

האם הקשר הזה בין ציוצים לטמפרטורה הוא **ליניארי**?
כן, אתה יכול לשרטט קו ישר אחד כמו הבא כדי לערוך יחס זה:



נכון, הקו לא עובר בכל נקודה, אך הקו מראה בבירור את הקשר בין ציוצים לטמפרטורה.
באמצעות המשוואה לקו ישר, תוכל לרשום את הקשר הזה באופן הבא:

$$y = mx + b$$

כאשר:

- y הוא הטמפרטורה - הערך שאנחנו מנסים לנבא.
- m הוא שיפוע הקו.
- x הוא מספר הציוצים בכל דקה - הערך של תכונת (פיצ'ר) הקלט שלנו.
- bהוא יירוט ה-y.

פונקציית הניבוי (לא מההרצאה)

לפי מוסכמות בלימוד מכונה, נכתוב את המשוואה למודל באופן קצת שונה, הפונקציה נקראת פונקציית הניבוי:

$$y' = h(x) = w \cdot x + b$$

כאשר:

- $h(x)$- הוא הלייבל החזוי (פלט רצוי).
- b - היא ההטיה "bias" (יירוט ה- y), המכונה לפעמים w_0.
- w- הוא המשקל של הפיצ'ר x. משקל הוא אותו מושג כמו "השיפוע" m במשוואה המסורתית של קו.
- x- הוא הפיצ'ר (קלט ידוע).

כדי לחזות את הטמפרטורה $h(x)$ לערך חדש של ציוצים לדקה x , אז פשוט נמיר את הערך x במודל זה.

למרות שמודל זה משתמש בפיצ'ר אחת בלבד, דגם מתוחכם יותר עשוי להסתמך על מספר פיצ'רים שלכל אחת מהן משקל נפרד. לדוגמה, מודל הנשען על שלוש פיצ'רים עשוי להיראות כך:

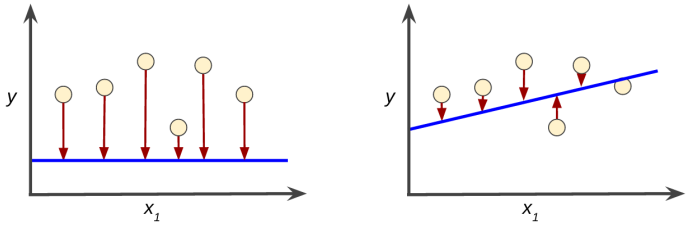
$$y' = w_1x_1 + w_2x_2 + w_3x_3 + b$$

גודל ומזעור הטעות (לא מההרצאה)

הכשרת מודל פירושה פשוט ללמוד (לקבוע) ערכים טובים לכל המשקולות וההטיה מדוגמאות הלייבלים.
אלגוריתם למידת מכונה בונה מודל על ידי בחינת דוגמאות רבות וניסיון למצוא מודל שממזער אובדן/טעות;
תהליך זה נקרא **מזעור טעות**.

הטעות הוא העונש לחיזוי רע. כלומר, טעות הוא **מספר** המציין עד כמה חיזוי המודל היה בדוגמה אחת.
אם חיזוי המודל מושלם, ההפסד הוא אפס; אחרת, ההפסד גדול יותר.
מטרת הכשרת המודל היא למצוא סט של משקולות והטיות בעלות אובדן נמוך, בממוצע, בכל הדוגמאות.
לדוגמא, האיור הבא מציג מודל בטעות גבוה משמאל ומודל בטעות נמוכה מימין:

- החץ האדום מייצג את **גודל הטעות**.
- הקו הכחול מייצג את הניבוי שלנו.



שימו לב שהחצים בעלילה השמאלית ארוכים בהרבה מעמיתיהם בעלילה הימנית.
ברור שהקו בעלילה הימנית הוא מודל ניבוי טוב בהרבה מהקו בעלילה השמאלית.

פונקציית הטעות/ההפסד (לא מההרצאה)

נשאל האם נוכל ליצור פונקציה מתמטית - **פונקציית הטעות** - שתצבור את הטעויות האישיות בצורה משמעותית.
מודלי הרגרסיה הליניארית שנבחן כאן משתמשים בפונקציית הטעות הנקראת **פונקציית הטעות בריבוע** המחשבת את הטעות הממוצעת בריבוע לכל דוגמא לאורך כל מערך הנתונים.
כדי לחשב אותה נסכום את כל הטעויות בריבוע עבור דוגמאות בודדות ואז נחלק במספר הדוגמאות הכולל שלנו:

$$\frac{1}{N} \cdot \Sigma_{(x,y)\in D} \left(h(x) \ - \ y \right)^2$$

כאשר:

- (x, y)היא דוגמה כך ש:
  - x הוא אוסף של פיצ'רים (למשל ציוצים לדקה, מין , גיל) שהמודל משתמש כדי ליצור ניבויים.
  - yהוא הדוגמה של הלייבל (למשל, הטמפרטורה).
- $h(x)$היא פונקציית הניבוי של המשקולות וההטיות בשילוב עם מכלול הפיצ'רים.
- Nהוא מספר הדוגמאות בקבוצה D.
- Dהוא קבוצת נתונים המכילה דוגמאות לייבל רבות, שהן זוגות (x, y).
  כלומר עבור הנתונים שלנו: $X = \{x_1, x_2,...., x_N\}, \ Y = \{y_1, \ y_2,...., y_N\}$
  הקבוצה Dהיא בעצם: $D = \{(x_1, y_1),..., (x_N, \ y_N)\}$

רגרסייה לינארית

נתונים לנו אוסף של נקודות X.
וגם אוסף של התיוגים של הנקודות Y.

אנחנו רוצים מסווג שיקח את הנקודות שלנו ב-d מימדים ויתן את הערך R כלומר $H: R^d \rightarrow R$.
מרחב ההיפוטזות ממפה את הנקודות לתיוגים.
הקבוצה H היא קבוצה אינסופית של היפותזות ו-h_w מגדירה היפותזה אחת.
h_w מוגדר על ידי הוקטור w. (מישור אפשר לייצג על ידי וקטור נורמאלי).

המטרה

נרצה למצוא מישור שימזער את הטעות (Loss ***Least-Squares***)

$$argmin_{w\in R^d} \sum_{i=1}^{n} \left(w \cdot x_i - y_i\right)^2$$

חסם הכללה

חסם הכללה עבור **Least Squares**.

<u>משפט - עבור קבוצה סופית</u>

יהי H קבוצה סופית של היפותזות וטווח התיוגים $[0, M]$.

יהי S קבוצת מדגם בגודל n.

אז לכל $0 < \delta < 1$ עם הסתברות של לפחות $1 - \delta$ מתקיים שלכל $h \in H$:

$$r(h) \leq \bar{r}(h) + M\sqrt{\frac{\log|H| + \log(2/\delta)}{2n}}$$

ה-M זה נרמול, אם נחלק ב-M אז כל התיוגים שלנו הם בין 0 ל-1.

ללא החילוק זה נותן לנו את ה-Scale של העולם.

<u>משפט - עבור קבוצה אינסופית</u>

יהי H קבוצה היפותזות עם VC_{Dim} שנסמנו d וטווח התיוגים $[0, M]$.

יהי S קבוצת מדגם בגודל n.

אז לכל $0 < \delta < 1$ עם הסתברות של לפחות $1 - \delta$ מתקיים שלכל $h \in H$:

$$r(h) \leq \bar{r}(h) + M\sqrt{\frac{O(d\log n + \log(1/\delta)}{n}}$$

נרצה למזער את הסיכון האמפירי.

החסם הכללה מניח שיש מספר סופי של חוקים.

אם יש לנו מספר אינסופי של חוקים, ראינו בקלסיפיקציה, לפי התיאוריה של VC רוב החוקים יעשו אותו הדבר - לכן אנחנו לא צריכים להתעסק עם כל החוקים אלא לקחת הרבה פחות חוקים שהם כן שונים ולעבוד עליהם.

אז אפילו אם H אינסופי אך עם VC_{Dim} קבוע אז אפשר להגיד שהסיכון האמיתי חסום על ידי הסיכון האמפירי פלוס הביטוי בשורש.

זה למה רגרסיה לינארית זה רעיון טוב, כי אם מישור במספר מימדים נמוך אז טוב להפריד את הנקודות האלה.

אתחול הבעיה

יש לנו מטריצה X של נקודות dxn

ניקח את התיוגים של הנקודות ונשים אותם בוקטור $Y \in R^n$.

פונקציית המטרה שלנו היא:

$$f(w) = \left|\left|X^T \cdot w - Y\right|\right|^2$$

כאשר $X^T \cdot w$ זה וקטור הניחושים ולכן Y זה ההפרש בין הניחוש לאמת שלנו.

כלומר $f(w)$ מסדר את הנקודות במטריצה וכל מקום בוקטור נחסיר ממנה את הוקטור Y של התיוגים האמיתיים והמכפלה הפנימית מייצגת את $argmin_{w \in R^d} \sum_{i=1}^{n} \left(w \cdot x_i - y_i\right)^2$.

איך נוכל למזער את זה? בעזרת נגזרת:

$$f'(w) = 2X\left(X^T w - Y\right) = 0$$

ונקבל:

$$XX^T w = XY$$

לכן אם ניקח את XX^T נקבל שזה 1 (למדנו באלגברה לינארית).

קיבלנו:

$$w = \left(XX^T\right)^{-1} XY$$

בעיה

הבעיה היא ש $\left(XX^T\right)^{-1}$ לא תמיד קיים, למדנו באלגברה לינארית שלא תמיד נוכל למצוא את המטריצה ההופכית. - מה נעשה במקרה שכזה?

נשתמש ב-**Pseudo-Inverse** שזה ביטוי שמאוד קרוב להופכית.

נסמן A^+

ומקיים את התכונות הבאות:

- $AA^+A = A$
- $A^+AA^+ = A^+$
- $(AA^+)^T = AA^+$
- $(A^+A)^T = (A^+A)$

מה זה A^+? - נגדיר כך:

$$A^+ = \lim_{\lambda \downarrow 0} \left(A^T A + \lambda I\right)^{-1} A^T = \lim_{\lambda \downarrow 0} A^T \left(AA^T + \lambda I\right)^{-1}$$

זה ביטוי מתמטי שהוא מאוד קרוב להיפוך של A.

תכונות

- $A^T A + \lambda I$ וגם $AA^T + \lambda I$ הם תמיד ניתנות להיפוך.
- הגבול קיים אפילו אם לא ניתן להיפוך.

התוצאה

צריך לפתור $XX^T w = XY$.

אם XX^T ניתן להפוך אז $w = (XX^T)^{-1} XY$

אחרת, $w = (XX^T)^+ XY$.

שני המקרים $(XX^T)^+$ וגם $(XX^T)^{-1}$ ניתן לחשב ב-$O(d^3)$.

לכן זמן הריצה הכולל יהיה $O(nd + d^3)$.

חסם הכללה

בהסתברות גבוהה, לכל $h \in H$ מתקיים:

$$r(h) \leq \bar{r}(h) + O\left(\sqrt{\frac{d}{n}}\right)$$

כאשר $M = 1$ וגם δ הוא קבוע.

מה קורה אם המימד גדול?

נשתמש ב-Margin בדומה לשיטה שעשינו בקלסיפיקציה.

חסם של מרווח גדול

אם כל השלושה מתקיימים:

- נקודות: $||x|| \leq R$ לכל $x \in X$
- תיוגים: $|y| \leq BR$ לכל $y \in Y$
- היפותזה: $h_w \in H$ מקיים $||w|| \leq B$.

אז בהסתברות גבוהה לכל $h_w \in H$ מתקיים:

$$r(h) \leq \bar{r}(h) + O\left(\frac{B^2R^2}{\sqrt{n}}\right)$$

זה נקרא **רגולציה** - מתמודד עם הבעיה של VC_{Dim} גבוהה.

רגרסיית רכס -Ridge

רגולציית Least Squares (נקרא גם Large-Margin).

נרצה למזער את סכום של ה-$Loss$ של הסיכון האמפרי וגם של הוקטור.

מתאים באופן אנלוגי למקרה של מרווח גדול שלמדנו ב-SVM.

$$Min_{w \in R^d}\left[\lambda||w||^2 + \sum_{i=1}^{n}\left(w \cdot x_i - y_i\right)^2\right]$$

בצורה מטריציונית

$$f(w) = \lambda||w||^2 + \left||X^T w - Y\right||^2$$
$$f'(w) = 2\lambda w + 2X\left(X^T w - Y\right) = 0$$
$$\left(XX^T + \lambda I\right)w = XY$$

פתרון:

$$w = \left(XX^T + \lambda I\right)^{-1} XY$$

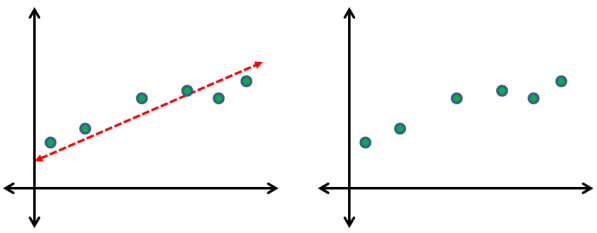
תמיד ניתנת להיפוך.

מה אם נרצה מסווג לא לינארי?

נרצה פונקציה לא לינארית.

כמו שעשינו ב-SVM שהכנסנו פונקצית Kernel , נעשה בדומה גם כאן.

$$Min_{w \in R^d}\left[\lambda||w||^2 + \sum_{i=1}^{n}\left(w \cdot \phi\left(x_i\right) - y_i\right)^2\right]$$



שימוש ב-LASSO

במקום למזער את הנורמה, נמזער את הסכום של הקואורדינטות של הוקטור.

אין את האפשרות להכניס פונקציות יותר מסובכות - קשה לפתור ופחות גמיש.

מה שטוב - עלול להוציא פתרון עם הרבה אפסים - זה פתרון יותר פשוט, כלומר אם יש לנו וקטור עם הרבה אפסים אז הוא בא ממשפחה פשוטה יותר.

אינטואיציה: אם ננסה למזער ב-L1 בדר"כ הנקודות יפלו בזוויות של החוק

ה-LASSO לוקח את הפונקציה שרצינו למזער ובמקום למזער את הנורמה של הוקטור:

$$Min_{w \in R^d} \left[\lambda ||w||_1 + \sum_{i=1}^{n} \left(w \cdot x_i - y_i \right)^2 \right]$$

רגרסיה לוגיסטית

מבוא (לא מההרצאה)

רגרסיה לוגיסטית הוא מודל נוסף לקלסיפיקציה.
במקום לחזות סיווג של בדיוק 0 או 1, רגרסיה לוגיסטית מייצרת הסתברות - ערך בין 0 ל -1.

<u>דוגמה</u> -
זיהוי דואר זבל. אם המודל מסיק להודעת דוא"ל מסוימת ערך של 0.932, זה מרמז על הסתברות של 93.2% שהודעת הדוא"ל היא ספאם. ליתר דיוק, פירוש הדבר שמכלול הדוגמאות שלגביהן המודל מנבא 0.932 יהיה למעשה דואר זבל 93.2% מהזמן ושאר 6.8% לא.

רגרסיה לוגיסטית היא מנגנון יעיל במיוחד לחישוב הסתברויות.
אנחנו יכול להשתמש בהסתברות שהוחזרה כהמרה לקטגוריה בינארית.
מטרת הסיווג בינארי היא לחזות נכון אחת משתי תוויות אפשריות (למשל, "דואר זבל" או "לא דואר זבל").

כיצד מודל רגרסיה לוגיסטית יכול להבטיח תפוקה שתמיד נופלת בין 0 ל -1.
הפונקציה הבאה מייצרת תפוקה עם אותם מאפיינים:

הפונקציה הלוגיסטית (לא מההרצאה)

- האוספים:

$$z = w_i x_i + b$$

- פונקציית הניבוי:

$$h(x) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(wx+b)}}$$

רגרסיה לוגיסטית

הוא לא באמת עושה רגרסיה אלא משהו דומה.
אם אנחנו רוצים קלסיפיקציה 0 או 1 אז יש איזשהו וקטור שמייצג מישור עם מכפלה פנימית של הנקודות ואם קיבלנו חיובי אז נקבל 1 ואם חיובי נקבל 1-.
ראינו ברגרסיה לא אכפת לנו באיזה צד מהמישור אנחנו נמצאים אלא מה המרחק שלנו מהמישור.
עכשיו אנחנו רוצים דבר אחר - מהי ההסתברות שהתיוג של הנקודה היא 1 או 0.
דוגמה של אלגוריתם שעושה את זה הוא רגרסיה לוגיסטית.

רגרסיה לוגיסטית עושה סיווג אבל משתמש במושג של רגרסיה כלומר במרחק מהמישור.
פונקציית רגרסיה לינארית: $w \cdot x$.

פונקציית רגרסיה לוגיסטית: $g(w \cdot x)$.

נראה כך:

$$g(z) = \frac{1}{1+e^{-z}}$$

כלומר הוא לוקח את המרחק מהמישור ומתרגם אותו לפונקציה g.

למשל אם אנחנו מאוד רחוק בצד של 1 מהמישור אז נגיד שב-99% אחוז שאנחנו 1.

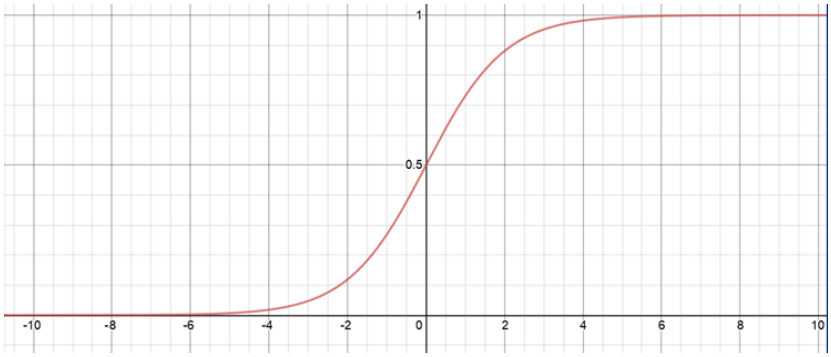
התכונות של הפונקציה g:

- פונקציה חלקה ומונוטונית.
- הטווח שלו חסום, כלומר כל z ממופה לערך בין 0 ל-1.
- אם ניקח את הנגזרת של g אז כש-z מספר קטן הביטוי e^{-z} זניח ולכן יצא לנו קו לינארי.

הוא ימצא מישור ובהינתן נקודה חדשה הוא יגיד לנו מה ההסתברות שהיא שייכת ל-1 או 0.

אם הנקודה היא על המישור אז המרחק שלה מהמישור הוא 0 ולכן ההסתברות היא 50% עבור 1 או 0.

אם הנקודה מאוד רחוק מהמישור לכיוון של 1 אז ב-99% הוא יגיד שהוא שייך ל-1 - באופן אנלוגי גם עבור 0.



למה הפונקציה מעניינת?

פונקציית ה-Odds לוקח הסתברות וממפה אותו לקו אינסופי:

$$\frac{\Pr(x)}{1-\Pr(x)}$$

פונקציית ה-Log-Odds לוקח טווח שבין אינסוף למינוס אינסוף וממפה אותו בין 0 ל-1:

$$\log\left(\frac{\Pr(x)}{1-\Pr(x)}\right)$$

כי אם ההסתברות להתקיים היא גדולה אז דבר גדול חלקי דבר קטן שואף לאינסוף.

אנחנו לא רוצים לקחת את ההסתברות ולמפה אותו לאינסוף אלא להפך ה-Log-Odds

איך נבחר את המישור ברגרסיה לוגיסטית?

נרצה לבחור את המישור ולמקסם את ההסתברות של המודל שיתן לכל נקודה את התיוג האמיתי שלה.
כלומר, נרצה לעשות:

$$\prod_{x_i:y_i=1} g(wx_i) \prod_{x_i:y_i=0} \left[1 - g(wx_i)\right]$$

זה פשוט של המכפלה שההסתברות של כל התיוגים של 1 וגם המכפלה של כל התיוגים של 0.

נמקסם בעזרת ה-Log באופן הבא:

$$\sum_{x_i:y_i=1} \log(g(w \cdot x_i)) + \sum_{x_i:y_i=0} \log(1 - g(w \cdot x_i)) =$$

$$= \sum_{x_i} \log\left[\left(1 - y_i\right) + \left(2y_i - 1\right) \cdot g\left(w \cdot x_i\right)\right]$$

הורדת מימד

הקללה של מימד גבוהה

בהינתן התפלגות נרצה לדעת עליה תכונות כמו צפיפות, שונות וכו'...
כשהמימד של הנקודות גדל, אז ההסתברות הופכת להיות נורא קשה לחישוב.

- הפונקציה של ההסתברות הופכת ליותר ויותר מורכבת.
- דורש יותר זיכרון.
- זמני הריצה והזיכרון גדל באופן אקספוננציאלי עם המימד.

ייתכן שמימד גבוהה זה מיותר, למשל עבור מיליון קואורדינטות נשמע כמו מקרה בו באמת לא רלוונטי.

משפט ג'ונסון ולינדנשטראוס

בהינתן n נקודות במימד גדול, אפשר להוריד את המימד ב-$\log(n)$ מבלי לאבד את היחס בין הנקודות (עם שינוי קטן).

למה JL

נתון אוסף V של n וקטורים מ-d מימדים.
קיימת פונקציה לינארית $f: V \to R^k$ עבור $k = \left\lceil \frac{8\ln(n)}{\varepsilon^2} \right\rceil$ כך שלכל $x, y \in V$ וגם $0 < \varepsilon \leq 1$
חסום על ידי:

$$(1 - \varepsilon)||x - y||_2 \leq ||f(x) - f(y)||_2 \leq (1 + \varepsilon)||x - y||_2$$

טרנספורמצית JL

אנחנו ניתן פונקציה אקראית לינארית שמקיימת את JL.

- f הוא מכפלת המטריצה F מגודל $k \cdot d$.
  - כאשר k הוא מספר העמודות ו-d הוא מספר השורות.
- F_{ij} הוא משתנה מקרי המתפלג ברנולי עבור הערכים $\{-1, 1\}$ בהסתברות $\Pr(F_{ij}) = \frac{1}{2}$.
  - בהמשך ננרמל את F ב- $\frac{1}{\sqrt{k}}$.

מבוא להוכחה

נוכיח ש-f משמר את המרחקים בין הנקודות.
בהינתן S אוסף נקודות, כל זוג נקודות $x, y \in S$ נרצה להראות את שאכן נשמר המרחק ביניהם.
יהא $w = x - y$.
נרצה להראות ש:

$$||f(x) - f(y)||_2 \approx ||x - y||_2$$

נכון גם עבור:

$$||f(w)||_2 \approx ||w||_2$$

- לפי הלינאריות: $f(x - y) = f(x) - f(y)$.

וגם עבור:

$$||f(w)||_2^2 \approx ||w||_2^2$$

אז אנחנו צריכים להראות שהטרנספורמציה משמרת את הנורמה של וקטורי המרחק w.

הוכחה
נתחיל עם מטריצה F שהיא רק עמודה אחת.

w	F	f(w)
(8, 1, 6, 3)	-1 1 1 -1	= (-4)

לפי הבניה:

$$f(w)^2 = \left(\sum_j F_j w_j\right)^2$$

לדוגמה, ב-4 מימדים:

$$f(w)^2 = \left(F_1 w_1 + F_2 w_2 + F_3 w_3 + F_4 w_4\right)^2 = \sum_{i=1}^{4} \left(F_i w_i\right)^2 + 2 \sum_{i<j} \left(F_i w_i \cdot F_j w_j\right)$$

ולכן במימד d נקבל:

$$f(w)^2 = \left(F_1 w_1 + ... + F_d w_d\right)^2 = \sum_{i=1}^{d} \left(F_i w_i\right)^2 + 2\sum_{i<j}\left(F_i w_i \cdot F_j w_j\right)$$

לפי לינאריות התוחלת נקבל:

$$E\left[f(w)^2\right] = E\left[\left(\sum_{j=1}^{d} F_j w_j\right)^2\right] = E\left[\sum_{j=1}^{d} \left(F_j w_j\right)^2 + 2 \cdot \sum_{i<j} F_i w_i \cdot F_j w_j\right] =$$
$$= \sum_{j=1}^{d} E\left[\left(F_j w_j\right)^2\right] + 2 \cdot \sum_{i<j} E\left[F_i w_i \cdot F_j w_j\right] =$$
$$= \sum_{j=1}^{d} w_j^2 = ||w||_2^2$$

כעת נניח מטריצה עם k עמודות במקום עמודה אחת.
יהא F_{ij} להיות השורה ה-j והעמודה ה-i.
לפי הבניה:

$$||f(w)||_2^2 = \sum_{i=1}^{k} \left(\sum_{j=1}^{d} F_{ij} w_j\right)^2$$

ראינו ש:

$$E\left[\left(\sum_{j=1}^{d} F_j w_j\right)^2\right] = ||w||_2^2$$

אז יש לנו:

$$E\left[||f(w)||_2^2\right] = E\left[\sum_{i=1}^{k} \left(\sum_{j=1}^{d} F_{ij} w_j\right)^2\right] = \sum_{i=1}^{k} E\left[\left(\sum_{j=1}^{d} F_{ij} w_j\right)^2\right] =$$
$$= \sum_{i=1}^{k} ||w||_2^2 = k||w||_2^2$$

ואם ננרמל את המטריצה F ב- $\frac{1}{\sqrt{k}}$ אז:

$$E\left[||f(w)||_2^2\right] = w_j^2$$

∎

- אז מה ראינו עד עכשיו:

לכל $x, y \in S$ כל קואורדינטה מקיימת בתוחלת:

$$\left|\left|w_j\right|\right|_2^2 = ||x - y||_2^2$$

מה באמת נרצה:

לכל $x, y \in S$ הפונקציות $f(x),\ f(y)$ משמרות $||w||_2^2$ עד לפקטור של $(1 \pm \varepsilon)$ בהסתברות גבוהה.

אם ההסתברות גבוהה מספיק $\left(1 - \frac{1}{2n^2}\right)$, אז הוא משמר כל n^2 מרחקים בקבוצה V במכה אחת:

$$\Pr(fails\ for\ one\ given\ pair) \leq \frac{1}{2n^2}$$

$$\Pr(fails\ for\ any\ pair) \leq \frac{n^2}{2n^2} = \frac{1}{2}$$

אפשר לראות את זה לפי חסם צ'רנוף:

לכל וקטור w, מתקיים:

$$\Pr\left[||f(w)||_2^2 > (1 + \varepsilon)||w||\right] = \Pr\left[||f(w)||_2^2 > (1 + \varepsilon)E||f(w)||^2\right] < e^{-\frac{k}{2}\left(\frac{\varepsilon^2}{2} - \frac{\varepsilon^3}{3}\right)}$$

$$\Pr\left[||f(w)||_2^2 < (1 - \varepsilon)||w||\right] = \Pr\left[||f(w)||_2^2 < (1 - \varepsilon)E||f(w)||^2\right] < e^{-\frac{k}{2}\left(\frac{\varepsilon^2}{2} - \frac{\varepsilon^3}{3}\right)}$$

לכן ניקח $k \sim \frac{\ln(n)}{\varepsilon}$.

דוגמה נוספת

פונקציה לינארית נוספת שמוסיפה את התכונות של JL.

יש לנו את פונקציה f שהיא מכפלת מטריצה F מגודל $k \cdot d$.

וגם F_{ij} הוא $(0, 1)$ משתנה מקרי המתפלג נורמלי.

<u>הוכחה</u>

נניח ש: F יש רק עמודה אחת.

לכן, $f(w) = \sum_i F_i w_i$ וגם $f(w)^2 = \left(\sum_i F_i w_i\right)^2$.

התפלגות נורמלית יש את התכונות הבאות:

- $a \cdot N(0, 1) = N(0, a^2)$
- $N(0, c) + N(0, d) = N(0,\ c + d)$
- אז: $a \cdot N(0, 1) + b \cdot N(0, 1) = N(0, a^2 + b^2) = \sqrt{a^2 + b^2}$.

נובע מכך ש:

$$f(x) = \sum_i F_i w_i \sim \sum_i w_i \cdot N(0,1) = ||w||_2 \cdot N(0,1)$$

$$f^2(x) = ||w||_2^2 \cdot N^2(0,1)$$

הוכחנו ש: $f(x)^2 \sim ||w||_2^2 \cdot N^2(0,1)$.

מה זה $N^2(0,1)$? - זה נקרא התפלגות $Chi - square$ עם דרגה אחת של חופש.

אם במטריצה F יש k עמודות אז:

$$||f(w)||_2^2 \sim \sum_{i=1}^{k} ||w||^2 \cdot N^2(0,1) = ||w||^2 \cdot \sum_{i=1}^{k} N^2(0,1)$$

מה זה $\sum_{i=1}^{k} N^2(0,1)$? - נקרא התפלגות $Chi - square$ עם k דרגות חופש.

משפט

יהא משתנה מקרי X מתפלג $Chi - squared$ עם k דרגות חופש ותוחלת $E(X) = 1$.

אז:

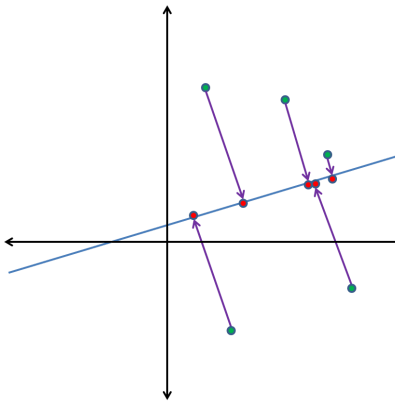
$$\Pr(X > (1+\varepsilon) \cdot E(X)) < e^{-\frac{k}{2}(\frac{\varepsilon^2}{2} - \frac{\varepsilon^3}{3})}$$

ניקח $k \geq \frac{4\ln(n)}{\frac{\varepsilon^2}{2} - \frac{\varepsilon^3}{3}}$, ההסתברות שזוג נקודות x, y תשתבש היא פחות מ: $\frac{1}{n^2}$.

מתקיים לכל הנקודות.

אם נשתמש ב-JL לעבור מ-2 מימדים למימד אחד.

מה שמטריצת המכפלה עושה זה לשקם נקודות ב-2 מימדים לתוך קו כלשהו:



חסם תחתון של JL

ה-JL מוריד מימד הכי טוב ל-$k = \frac{8\ln(n)}{\varepsilon^2}$.

האם אפשר לעשות את זה יותר הדוק?

קל להוכיח שאי אפשר להוריד יותר מזה.

ניתוח גורמים ראשיים - PCA

הורדת מימד

ראינו שמטרת הלמה JL היתה להוריד מימד ולשמור על מרחק בין הנקודות.

ניתוח גורמים ראשיים או באנגלית **P**rinciple **C**omponent **A**nalysis כלומר **PCA** נועד כדי:

1. להוריד מימדים
2. למזער את סכום המרחקים של מימדים חדשים
3. היה המרכיב המרכזי בזכייה בפרס נטפליקס

הגדרה

בהינתן n נקודות $x_1, \ldots, x_n \; in \; R^d$ וגם מימד מטרה k.

מצא את תת המרחב ממימד k הטוב ביותר בקירוב לנתונים.

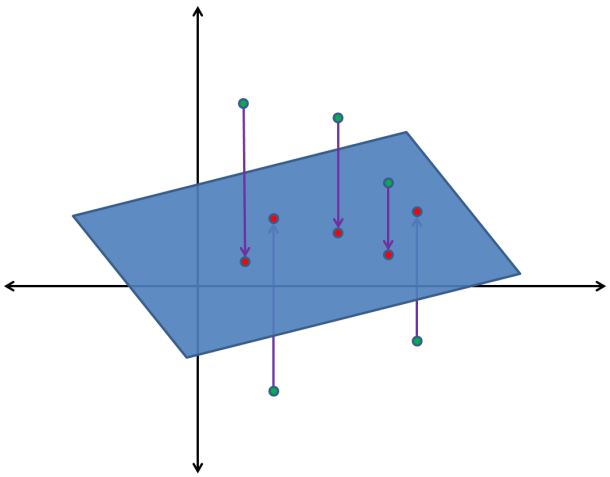
למה זה שימושי?

- מבטל רעש ממדים גבוהים
- מפחית נתונים

דוגמאות:

- פרס נטפליקס
- זיהוי פנים

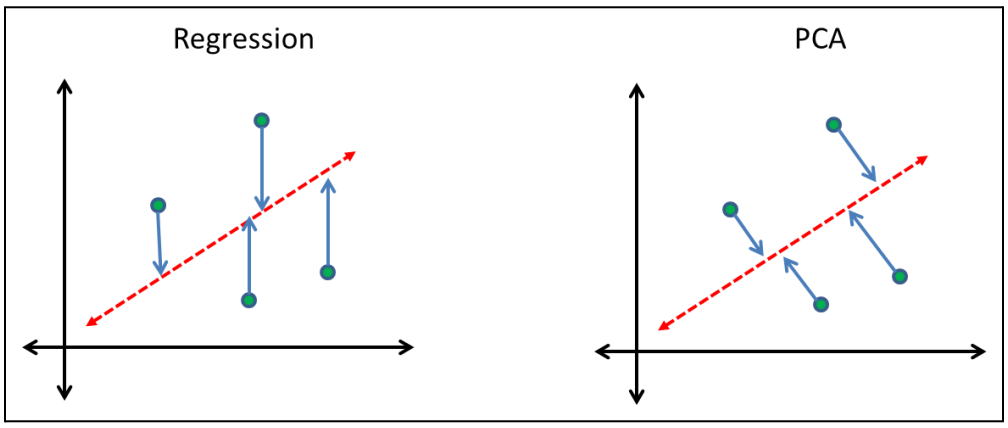
עקרון **PCA** ממזער את סכום האורכים (הסגולים) בציור:



ההבדל בין רגרסיה ל-PCA

רגרסיה: סכום המרחקים בריבוע למישור בכיוון y

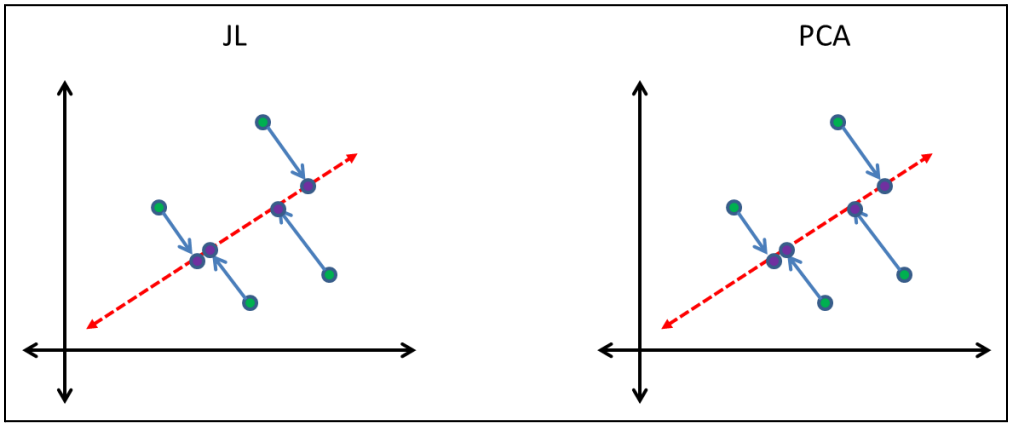
PCA: סכום המרחקים בריבוע של נקודות למישור



ההבדל בין JL ל- PCA

JL: עיוות במקרה הגרוע ביותר בין נקודות חדשות

PCA: סכום המרחקים בריבוע של נקודות מקוריות לנקודות חדשות



באופן פורמלי

בהינתן נקודות $x_1, \ldots, x_n \; in \; R^d$ ומימד מטרה $k << d$.

נמצא מטריצות $U \in R^{dxk}$ וגם $V \in R^{kxd}$ שממזערות את הפונקציה (הסכום) הבאה:

$$f(U, V) = \Sigma_{i=1}^{n} \left|\left|x_i - UVx_i\right|\right|_2^2$$

- מטריצת V הוא **המדחס** - הוא ממפה את הנקודות אל תת-המרחב ה-k-ממדי
- מטריצת U הוא **המפרק** - הוא ממפה את תת-המרחב ה-k-ממדי בחזרה למרחב ה-d-ממדי, כך שנוכל למדוד כמה הנקודה זזה.

טענה

בפתרון אופטימלי,

- $U = V^T$ שזה אומר שהמדחס = המפרק
- $U^T U = 1$ שזה אומר שהעמודות של U אורתונורמליות.

הוכחה

המפה UVx יש את הטווח R של k מימדים.

יהא $w_1,\ldots,w_k$ בסיס אורתונורמלי עבור R.

יהא W מטריצה עם העמודות האלו ($W^T W = I$).

אז לכל x_i יש z_i המקיים: $UVx_i = Wz_i$.

איזה z_i ממזערים את $\left\|x_i - UVz_i\right\|_2^2$?

$$f = \left\|x_i - UVz\right\|_2^2 = \left\|x_i\right\|_2^2 + ||z||_2^2 - 2zWx_i$$

$$0 = f' = 2z - 2Wx_i \quad \Rightarrow \quad z_i = W^T x_i$$

$$\Sigma_{i=1}^m \left\|x_i - UVx_i\right\|_2^2 = \Sigma_{i=1}^m \left\|x_i - Wz_i\right\|_2^2 = \Sigma_{i=1}^m \left\|x_i - WW^T x_i\right\|_2^2$$

בהינתן ש: $U = W,\ V = W^T = U^T$ וגם $U^T U = I$.

חידוש הבעיה:

בהינתם נקודות $x_1, \ldots, x_n$ ב-R^d ובמימד היעד k.

מצא מטריצה $U \in R^{dxk}$ עם $U^T U = I$.

הממעזר: $\Sigma_{i=1}^m \left\|x_i - WW^T x_i\right\|_2^2$.

מסתבר ש-U היא המטריצה עם k הווקטורים העצמיים הראשונים של מטריצת הנקודה X...

$$\|x_i - UU^T x_i\|_2^2 = \|x_i\|^2 - 2x_i^T UU^T x_i + x_i^T UU^T UU^T x_i$$
$$= \|x_i\|^2 - x_i^T UU^T x_i$$
$$= \|x_i\|^2 - trace(x_i^T UU^T x_i) \quad \text{scalar}$$
$$= \|x_i\|^2 - trace(U^T x_i x_i^T U) \quad \text{matrix}$$

העקבות של מטריצה מרובעת היא סכום האלמנטים האלכסוניים שלה:

- $trace(A) = \sum_i a_{ii}$.
- בלתי משתנה תחת משמרות מחזוריות: $trace(ABC) = trace(CAB) = trace(BCA)$.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} -1 & 0 & 3 \\ 11 & 5 & 2 \\ 6 & 12 & -5 \end{pmatrix}$$

הוכח ש:

$$\left\|x_i - UU^T x_i\right\|_2^2 = \left\|x_i\right\|^2 - trace\left(U^T x_i x_i^T U\right)$$

נרצה למצוא U שממזער:

$$\sum_i \left[\left\|x_i\right\|^2 - trace\left(U^T x_i x_i^T U\right)\right]$$

וגם למקסם את U:

$$\sum_i trace\left(U^T x_i x_i^T U\right)$$

סקירה של וקטורים עצמיים

נניח $A \in R^{nxn}$ היא סימטרית, $A = A^T$.

אם $Av = \lambda v$ עבור $v \in R^n$ וגם $\lambda \in R$.

אז v הוא וקטור עצמי של A עם הערך העצמי λ.

טענה

עבור A סימטרי, הוקטור העצמי עם ערכים עצמיים שונים הוא אורתוגונלי.

הוכחה

ניקח וקטורים u, v עם $Av = \lambda v$ וגם $Au = \beta u$ כאשר $\beta \neq \lambda$.

$$u^T Av = u^T(Av) = \lambda u \cdot v$$

$$u^T Av = (u^T A) \cdot v = (A^T u)v = (Au)v = \beta u \cdot v$$

מאחר והנחנו ש: $\beta \neq \lambda$ אז חייב להתקיים ש: $u \cdot v = 0$.

משפט

עבור סימטריות $A \in R^{nxn}$ קיימת מטריצה אורתוגונלית $V \in R^{nxn}$, $\left(V^T = V^{-1} \; so \; V^T V = I\right)$

כך ש: $V^T AV = \Lambda = diag(\lambda_1, \ldots, \lambda_n)$ וגם $\lambda_1 \geq \ldots\ldots \geq \lambda_n$.

נאמר ש: V **מלכסן** את A.

זה נקרא **פירוק וקטור עצמי**

- שורה v_i היא של V הוקטור העצמי של A המקביל ל- λ_i.
- השורות v_i פורסות את R^n.

תמונת מצב

- נזכור: רוצה למצוא U שממקסם את $\sum_i trace\left(U^T x_i x_i^T U\right)$

  הגדר $A = \sum_i x_i x_i^T$ נרצה למקסם $trace(U^T AU)$.

- לכסון A

  $A = V\Lambda V^T$ כאשר $V^T V = VV^T = I$ וגם $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$.

- המטרה: נרצה לבחור U כך ש:
  1. $U \in R^{dxk}$
  2. $U^T U = I$
  3. וגם U ממקסם את $trace\left(U^T V\Lambda V^T U\right)$.

לכל U חוקי נגדיר $B = V^T U$.

$$trace(U^T V \Lambda V^T U) = trace(B^T \Lambda B) = trace(\Lambda B B^T) = \sum_{j=1}^{d} \lambda_j \sum_{i=1}^{k} B_{ji}^2$$

כעת נגדיר $\beta_j = \sum_{i=1}^{k} B_{ji}^2$.

- β_j הוא הכניסה ה-j באלכסון של המטריצה BB^T.
- ממקסם את $\sum_{j=1}^{d} \lambda_j \beta_j$.

טענה

$$\sum_{j=1}^{d} \beta_j = k$$

הוכחה

$$\sum_{j=1}^{d} \beta_j = trace(BB^T) = trace(B^T B) = trace(U^T V V^T U) = trace(U^T U) = trace(I) = k$$

טענה

$$\beta_j \leq 1$$

הוכחה

נכון כי B הוא אורתוגונלי

טענה

$$\sum_{j=1}^{d} \lambda_j \beta_j \leq \sum_{j=1}^{k} \lambda_j$$

הוכחה

מכיוון שהערכים העצמיים יורדים, הסכום מוגדל על ידי נטילת $\beta_j = 1$ לכל $j \leq k$.

לסיכום

- מצאנו U שממקסם את $trace(U^T AU)$.
- $A = \sum_{i=1}^{n} x_i x_i^T$
- $A = V\Lambda V^T$ כאשר $V^T V = VV^T = I$ וגם $\Lambda = diag(\lambda_1, \dots, \lambda_n)$ כאשר $\lambda_1 \geq \dots \geq \lambda_n$.
- הראינו ש: $trace(U^T AU) \leq \sum_{j=1}^{k} \lambda_j$.
- אפשר לקיים את זה לשוויון: נגדיר עמודה i של U להיות וקטור עצמי של עמודה i של A.

חלאס.