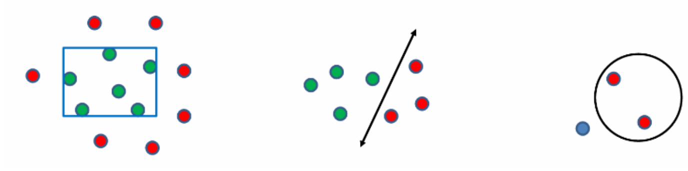
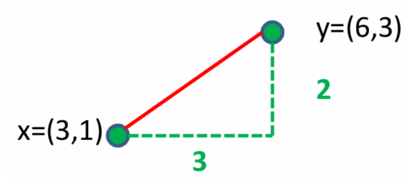
למידת מכונה הרצאה 7



ראינו מסווגים מסוגים שונים.

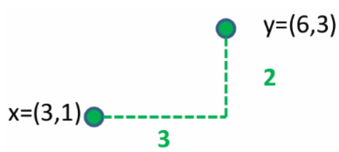
כולם מניחים שהנקודות הן ווקטורים ושהמרחק הוא אוקלידי.

זו לא הדרך היחידה למדוד מרחק!



מרחק אוקלידי: נקרא גם ℓ_2. המרחק בין ווקטורים x, y:

$$\|x-y\|_2 = \left(\sum_{i=1}^{d} |x_i - y_i|^2\right)^{1/2}$$



מרחק מנהטן: גם "מרחק מוניות". המרחק מעל ℓ_1. המרחק בין שני ווקטורים x, y:

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

מרחב ℓ_p: מוגדר לכל $p \geq 0$. המרחק בין שני ווקטורים x, y:

$$\|x-y\|_p = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{1/p}$$

מרחק פרישה (*frechet distance*): מסומן גם ℓ_∞. המרחק בין שני ווקטורים x, y:

$$\|x-y\|_\infty = \left(\sum_{i=1}^{d} |x_i - y_i|^\infty\right)^{1/\infty} = \max_i |x_i - y_i|$$

כי בתוך הסכום, הכי משמעותי זה המרחק הכי גדול בחזקת אינסוף. אז מתייחסים רק אליו. ואז זה בעצם ההפרש הזה, בחזקת אינסוף ואז בחזקת אחד חלקי אינסוף – זה מצטמצם.

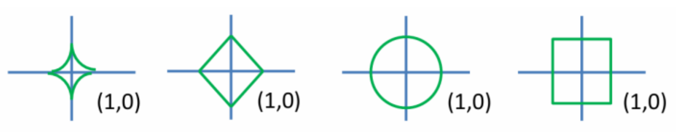
מה היחס בין מרחבי ℓ_p שונים? בדוגמה שראינו, עבור $x = (3,1),\ y = (6,3)$:

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

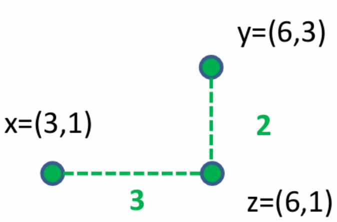
כאשר p גדל, המרחק לא גדל (יכול להישאר אותו דבר). קל לראות את ההבדל בין ℓ_1 ל- ℓ_∞: הראשון הסכום של ההפרשים בכל הקורדינטות, והשני זה רק ההפרש הכי גדול.

דוגמה עבור $x = (0,0),\ y = (0,1)$ – המרחק לא משתנה. $\|x-y\|_p = 1$ לכל $p \geq 0$.

נתבונן בכדור היחידה: מוגדר לפי כל הנקודות שהמרחק מהן לראשית הצירים הוא 1. במרחק אוקלידי (ℓ_2) זה כדור. במרחבי ℓ_p שונים:



$\ell_{1/2}$, ℓ_1, ℓ_2, ℓ_∞



מרחבי ℓ_p בשברים: איך ℓ_p מתנהג כאשר $p<1$?

עבור $x=(3,1),\ y=(6,3),\ z=(6,1)$:

$$\|x-y\|_{1/2}=\left(3^{1/2}+2^{1/2}\right)^{\frac{1}{1/2}}\approx 9.9,$$
$$\|x-z\|_{1/2}+\|z-y\|_{1/2}=3+2=5$$

לא מקיים את אי – שוויון המשולש! "מעגל היחידה" קעור.

מה זה ℓ_0? המרחק בין שני וקטורים x,y:

$$\|x-y\|_0=\left(\sum_{i=1}^{d}|x_i-y_i|^0\right)^{1/0}$$

אי אפשר לחשב את החזקה $1/0$, אז נתייחס ל $\lim_{p\to0}\left(\sum_{i=1}^{d}|x_i-y_i|^p\right)^{1/p}$. כל קורדינטה שבה ההפרש הוא לא 0, ההפרש בחזקת 0 הוא 1. כלומר הסכום הוא בעצם **בכמה קואורדינטות** יש הבדל בין x,y.

הרבה פעמים מגדירים את ℓ_0 להיות בדיוק הדבר הזה – כלומר מתעלמים מהחזקה.

מטריקה

מטריקה צריכה לקיים:

1. סימטריה ($symmetric$): $d(x,y)=d(y,x)$,
2. אי – שוויון המשולש: $d(x,y)\le d(x,z)+d(z,y)$.

סמי – מטריקה (*semi – metric*) מקיימת רק סימטריה.

לדוגמה: ℓ_p עבור $p<1$. מחירי טיסות – יותר זול לטוס עם קונקשן מאשר טיסה ישירה.

קוואזי – מטריקה (*quasi - metric*) מקיימת רק את אי"ש המשולש.

לדוגמה: זמני נסיעה (פקקים) או הליכה (עליות).

עוד פונקציות מרחק:

מרחק עריכה – *edit distance*: הניסיון להתאים מחרוזות לשכן הכי קרוב אליה – חיפושי גוגל, או השלמה אוטומטית.

יש 3 פעולות: מחיקת תו, הוספת תו (לכל מיקום), החלפת תו (מחיקת תו והוספת תו אחר באותו מקום).

```
Levenshtein(string s, string t)
      Int[][] a = array[s.length + 1][t.length + 1]
      int count_s = 0, count_t = 0
      for (int i=0; i ≤ s.length; i++) a[i][0] = count_s++
      for (int j=0; j ≤ t.length; j++) a[0][j] = count_t++
      for (int i=1; i ≤ s.length; i++)
            for (int j=1; j ≤ t.length; j++)
                  if(s.charAt(i-1) == t.charAt(j-1))
                        a[i][j] = min(a[i-1][j-1], a[i][j-1]+1, a[i-1][j]+1)
                  else
                        a[i][j] = min(a[i-1][j-1]+1, a[i][j-1]+1, a[i-1][j]+1)
      return a[s.length()][t.length()]
```

מרחק לוינשטיין ($Levenshtein\ distance$) הוא מספר הפעולות המינימלי שצריך כדי להגיע ממחרוזת אחת לשנייה.

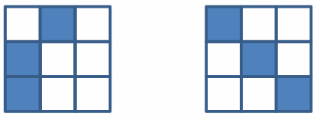
מחיקה והוספה שווים 1. החלפה שווה 1 או 2 (במקרה שמתייחסים לזה בתור מחיקה והוספה).

לדוגמה – DNA. כדי למדוד עד כמה שני אורגניזמים קרובים, אפשר לבדוק מה מרחק העריכה.

אפשר לחשב ע"י תכנות דינאמי – בעצם מחשב את המרחקים בין כל רישא של s, t. זמן ריצה $O(|s| \cdot |t|)$.

תמונות:

איך מודדים את המרחק בין תמונות? מרחק ווקטורי לא עובד – נכשל כאשר מזיזים תמונה בעמודה אחת.



מרחק $earth\ mover's$ – המחיר של להזיז פיקסל מתמונה אחת לשנייה.
קובעים את המחיר של תנועה אנכית, אופקית, אלכסון.

איך יודעים מה הדרך הכי יעילה להזיז פיקסלים כדי לעבור מתמונה אחת לשנייה?

בעיית השמה – $\boldsymbol{assignment\ problem}$.

Task / Worker	Clean bathroom	Sweep floors	Wash windows
Alice	**$8**	$4	$7
Bob	$5	$2	**$3**
Carol	$9	**$4**	$8

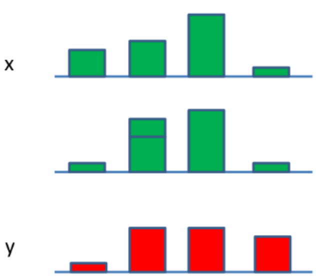
קלט: N עובדים, N משימות.

מטריצה $N \times N$. המחיר של כל עובד שעושה כל משימה:

פלט: השמה חד – חד – ערכית של עובד למשימה. כל פועל חייב לקבל בדיוק משימה אחת. רוצים למזער את המחיר הכולל.

במקרה שלנו – "לייצר" פיקסל בתמונה השנייה זו משימה, והפיקסל מהתמונה הראשונה זה פועל. רוצים להתאים פועל למשימה – כל פיקסל, לאן הוא יזוז כדי לייצר את התמונה השנייה (ורוצים למזער את המרחק הכולל שהפיקסלים יזוזו).

בעיית $min\ sum\ total\ matching$. פתרון – "אלגוריתם הונגרי"



עוד דוגמה ל $earth\ mover$ – מעבר מווקטור אחד לשני:

$$x = (3,4,7,1) \xrightarrow{move\ 2\ from\ x_1 \to x_2} (1,6,7,1) \longrightarrow$$
$$\xrightarrow{move\ 1\ from\ x_2 \to x_1} (1,5,8,1) \xrightarrow{move\ 3\ from\ x_3 \to x_4} (1,5,5,4) = y$$

בסה"כ: $2 + 1 + 3 = 6$.

מסווג "שכן הכי קרוב" – $\boldsymbol{nearest\ neighbor\ classifier}$**:**

נרצה קודם כל להגדיר איך מודדים מרחק. המרחק האוקלידי הוא לא הדרך היחידה (אבל עדיין כן הדרך הכי נפוצה).

איך אפשר ללמוד במרחבים לא אוקלידיים? אין מישור.

גישה אחת: ניקח קבוצת בסיס (יכול להיות כל הנקודות). לכל נקודה ניתן תיוג לפי הנקודה שהכי קרובה אליה.
נרצה לשאול – האם זה מזווג טוב או לא. החסמים שראינו – לפי טעות אמפירית נמוכה, **ומימד-**$\boldsymbol{VC}$ **נמוך.**

הבעיה – ש לזה מימד-VC אינסופי! הוכחה:

נאמר שהמימד-VC הוא אינסוף אם לכל קבוצת נקודות, לכל תיוג אפשרי, יש מזווג שמשיג את התיוג הזה. וזה מתקיים באופן טריוויאלי עבור שכן קרוב – כי לכל קבוצת נקודות, ניקח אותה בתור הבסיס. ואז כל נקודה היא הכי קרובה לעצמה.

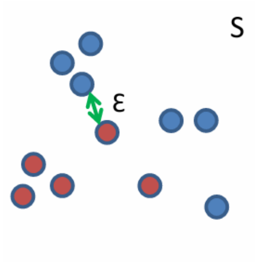
חסמי הכללה:

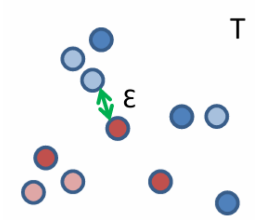
***k* – שכנים קרובים (*k – nearest neighbors*).** נסתכל על ה-k שכנים הכי קרובים וניקח את ההסכמה של הרוב.

- פותר הרבה בעיות.
- בפרט, שימושי בשביל להוריד רעש.

***Condensing*:** נגביל את הבסיס ע"י גודל s כלשהו. ואז אין את הבעיה שיש עם מימד-*VC* אינסופי. כי אי אפשר לקחת כל קבוצה בתור הבסיס. אז מספר התיוגים האפשריים (במדגם בגודל n): $O(n^s)$. (s בחירות מתוך n). והמימד-*VC*: $O(s)$.

נשאל, איך לבחור את הבסיס.





דרך אחת לבחור היא לקחת "רשת אפסילון":

בהינתן S קבוצת *train*, יהי ε המרחק המינימלי בין נקודה כחולה לאדומה. כלומר, המרחק בין כל שתי נקודות מצבעים שונים הוא לפחות אפסילון. עכשיו אפשר לקחת רשת על המדגם. T היא הרשת של S ומקיימת:

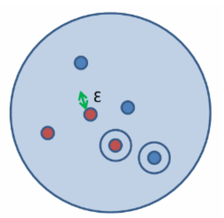
לכל שתי נקודות ברשת, המרחק ביניהם הוא לפחות אפסילון: $\forall p,q \in T : d(p,q) \geq \varepsilon$.

לכל נקודה שלא ברשת, המרחק ממנה לנקודה כלשהי ברשת הוא לכל היותר אפסילון:
$\forall p \in S : \exists q \in T : d(p,q) < \varepsilon$. או פשוט: $\forall p \in S : d(p,T) < \varepsilon$.

בניית T באלגוריתם חמדן:

1. נאתחל את $T = \phi$.
2. לכל $p \in S$, אם $d(p,T) > \varepsilon$, נוסיף את p ל-T.

ניקח את T בתור בסיס ל-*nearest neighbor*. עבור *1-NN* הוא עקבי על כל S. למה? כי אם נקודה כחולה (בה"כ), זה אומר שבתוך T יש או את נקודה עצמה, או נקודה אחרת במרחק פחות מאפסילון. ואנחנו יודעים שהנקודה האדומה שהכי קרובה אליה היא במרחק לפחות אפסילון.



מה המימד-*VC* של הרשת? נחשב עבור מרחב אוקלידי.

כל הנקודות מוכלות בתוך כדור היחידה.

הנקודות של T הן במרחק לפחות אפסילון אחת מהשנייה.

אז מסביב לכל נקודה נוכל לשים כדור ברדיוס $\varepsilon/2$, והם לא נחתכים.

אז נחשב חסם עליון לגודל של T, לפי הניתוח: כמה כדורים כאלה אפשר להכניס בתוך כדור היחידה. זה חישוב של נפח. ניקח את נפח כדור היחידה, חלקי נפח של כדורי קטן. משוואת מפה כדור במרחב:

$$V_d(R) = \frac{\pi^{d/2} \cdot R^d}{\Gamma\left(\frac{d}{2}+1\right)}$$

כאשר Γ היא פונקציית גאמא של אוילר (הכללה של עצרת עבור מספרים לא שלמים). הפונקציה שווה $n!$ לכל n טבעי.

המכנה זהה בנפח של כדור היחידה והכדורים הקטנים אז הוא מצטמצם. וגם R^d. אז יוצא:

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