# למידת מכונה הרצאה 8

למדנו שקיים המושג למידה – שאפשר על ידי מדגם שמייצג את העולם, להגיע למסקנות לגבי העולם. תיארנו את רמת הפשטות של חוק על ידי מימד-VC (VC dimension). ראינו שככל שהחוק מסובך יותר (בדרך כלל מדויק יותר על מדגם מסויים), הטעות שלו עולה (הוא פחות מדויק על שאר העולם). זה נקרא התאמת יתר – overfitting.

באלגוריתמים, התחלנו באלגוריתמים שמוצאים מישור מפריד (בעולם עם מימד $d$, מישור ממימד $d-1$). וראינו שמישור במימד נמוך הוא טוב. במימד גבוה, המימד-VC עולה, אבל ראינו שאפשר לחסום את המימד-VC יותר אם למישור יש מרווח.
האלגוריתמים perceptron, adaboost שראינו הם אלגוריתמים כאלה.

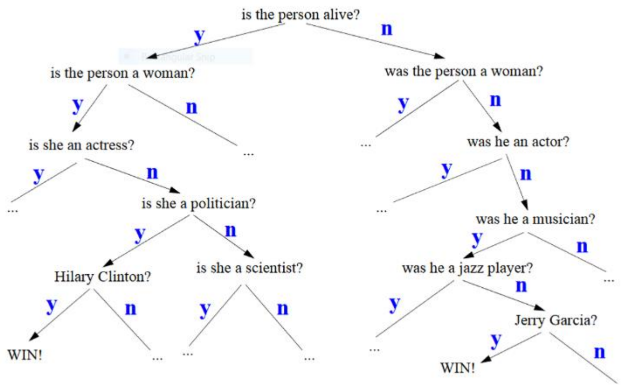
ראינו עוד שיטה – שכן קרוב. שיכול לפעול לא רק על דברים לינאריים, אלא בעצם על כל קבוצת אובייקטים שמוגדרת עבורה פונקציית מרחק.

מזווגים שמוצאים מישור הם שימושיים בנקודות במימד אוקלידי (כל ה-SVM שראינו). אנחנו בנושא של מזווגים לא-מישוריים שהם יחסית פשוטים – בעלי מימד-VC נמוך.

## עצי החלטה

מזווג לא לינארי, מאוד פופולרי. מבוסס על **תורת האינפורמציה – information theory**:

המשחק 20 שאלות – אחד חושב על אדם כלשהו, והשני צריך לנחש מי הוא. השאלות הכי טובות הן כאלה שחוצות את האפשרויות לחצי.
דוגמה חלקית לעץ אסטרטגיה:




לעץ בינארי עם 20 רמות יש: $2^{20} = 1048576$ עלים.

כלומר אם יש 1048576 אופציות, וכל פעם אנחנו חוצים לחצי, תמיד נגיע לתשובה.

### משחק מסוג אחר

מה אם המטרה שלנו היא לשאול כמה שפחות שאלות בממוצע? אז אם יש אנשים שיש הסתברות יותר גדולה שיחשבו עליהם, נרצה לשאול שאלות מתאימות לזה. מספר השאלות המצופה (התוחלת בעצם):

$$\sum_i P(x_i) \cdot \text{number\_of\_guesses}(x_i)$$

הסכום על כל האנשים האפשריים של: ההסתברות שחושבים על אדם מסויים, כפול מספר השאלות שצריך כדי לנחש אותו.

### דחיסה

לדוגמה, תמונות. Bitmap זה הפורמט היחיד שלא דחוס (ממש שומר כל פיקסל ואת הצבע שלו). כל פורמט אחר משתמש בדחיסה.
נרצה לשאול, עד כמה אפשר לדחוס קובץ מסויים? כמה ביטים צריך כדי לשמור את הקובץ?

נתחיל במקרה הפשוט – טקסט בלבד. נתבונן בפריט מסויים – תו מתוך הטקסט.

התיאוריה שייסדה את תורת המספרים (Shannon, 1948): אם לתו $i$ יש תדירות $w_i$ בקובץ, אז הדחיסה האופטימלית של קובץ באורך $m$ היא לפחות בגודל (בביטים):

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

הנחות: כל תו מקודד כמחרוזת סופית של ביטים. הייצוג של כל תו לא משתנה בזמן הקידוד.

לדוגמה, מסמך באורך 50 אותיות שהם רק $a, b$. $a$ מופיע 20 פעמים, $b$ מופיע 30. אז אפשר לדחוס:

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

הנוסחה לעיל היא **פונקציית האנטרופיה.** מדד של האקראיות של הדאטא. הפונקציה יכולה להיות מוגדרת על כמה מקרים, והיא:

$$\sum_i P(e_i)\log_2\left(\frac{1}{P(e_i)}\right)$$

לדוגמה, עבור מטבע הגון:

$$P(\mathrm{h})\log_2\left(\frac{1}{P(\mathrm{h})}\right)+P(\mathrm{t})\log_2\left(\frac{1}{P(\mathrm{t})}\right)=\frac{1}{2}\log_2\left(\frac{1}{1/2}\right)+\frac{1}{2}\log_2\left(\frac{1}{1/2}\right)=\log_2(2)=1$$

שזה בעצם אנטרופיה מקסימלית.

אם המטבע תמיד יוצא עץ:

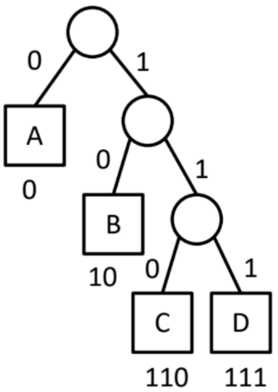
$$P(\mathrm{h})\log_2\left(\frac{1}{P(\mathrm{h})}\right)+P(\mathrm{t})\log_2\left(\frac{1}{P(\mathrm{t})}\right)=1\log_2(1)+0\log_2\left(\frac{1}{0}\right)=0+0=0$$

באופן כללי, הפונקציה עבור מאורעות בינאריים (רק שני מקרים):

$$P(e_1)\log_2\left(\frac{1}{P(e_1)}\right)+\left(1-P(e_1)\right)\log_2\left(\frac{1}{1-P(e_1)}\right)$$

משמעות המשפט: ככל שהמידע יותר אקראי, פחות אפשר לדחוס אותו.

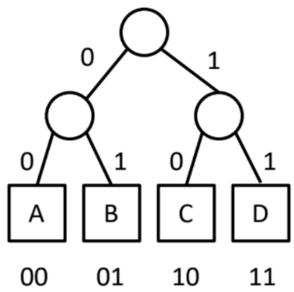
החסם של Shannon הוא חסם תחתון, והוא לא נותן אלגוריתם כדי להשיג את החסם הזה.
האלגוריתם של הופמן (Huffman, 1952) מתאר אלגוריתם שמשיג את החסם האופטימלי – בנייה מלמטה-למעלה:



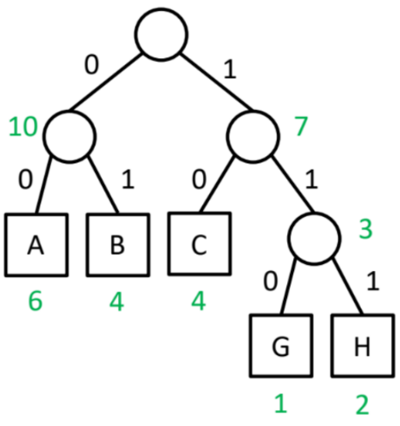

עבור המחרוזת BAABAC:

עץ מאוזן – לא הכי אופטימלי. 010000010010

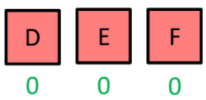
עץ אופטימלי: 1000100110




לדוגמה עבור תווים: A, B, C, D, E, F, G, H, המחרוזת AAAAABBAHHBCBGCCC.
אם נבנה עץ מאוזן, עבור 8 תווים כל תו 3 ביטים, אורך המחרוזת יהיה $3\cdot 17=51$ ביטים. אם נסתכל על התדירויות: $\{A: 6, B: 4, C: 4, D: 0, E: 0, F: 0, G: 1, H: 2\}$.



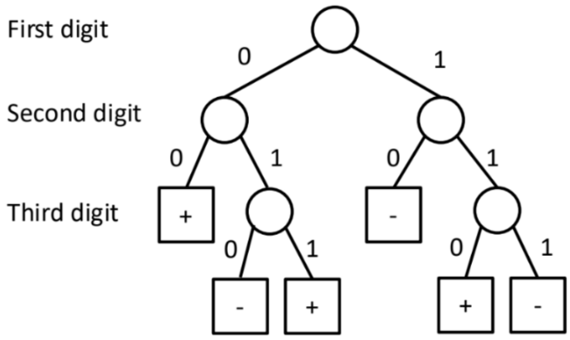

נבנה את העץ מלמטה למעלה, וכל פעם נחבר את שני הקודקודים הכי קלים:



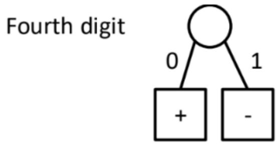
| | |
|---|---|
| 0000 | + |
| 0010 | + |
| 0101 | - |
| 0110 | + |
| 1001 | - |
| 1011 | - |
| 1100 | + |
| 1111 | - |

**למידה בעזרת עצי החלטה**

לדוגמה, מדגם של וקטורים בינאריים. החוק שלנו יהיה לפי הערך בכל מימד:




העץ הזה עקבי לגמרי על המדגם שלנו. אבל נשים לב שגם העץ הקטן הוא עקבי:




אינטואיטיבית, העץ הגדול יכול להשיג יותר תיוגים. מה המימד-VC של עץ על וקטורים בינאריים באורך $d$?

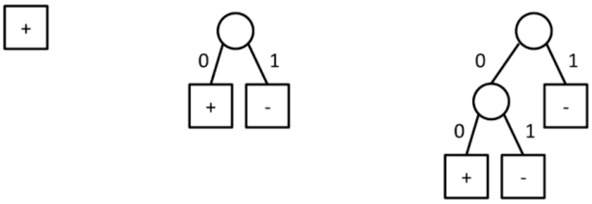
$T(k) =$ משפחת עצי ההחלטה עם $k$ קודקודים. בכל קודקוד, נבחר מימד לבחון, אן נסמן את הקודקוד בתור עלה $(+,-)$. סה"כ $d+2$ אפשרויות. כלומר $T(k) \le (d+2)^k$. לפי הלמה של סאואר: $\text{VC-dim}(T(k)) = O(k \log(d))$.

אז אנחנו נרצה $k$ קטן יותר. הבעיה: איך למצוא עץ שאחיד (או כמעט) על המדגם, עם כמה שפחות קודקודים. זה NP-hard. במקום זה, נשתמש בהיוריסטיקה: *ID3*, *pruning*.

## יער רנדומי – *Random forest*

אלגוריתם ID3:

- נתחיל עם שורש שהוא עלה $+$.
- פונקציית מחיר מודדת את איכות הפיתרון.
- נחלק את העלה עם חוק שממזער את המחיר.
- נחזור עד שהעץ עקבי על המדגם.




מהי פונקציית מחיר טובה?

כזאת שממזערת את האנטרופיה של העלים. בהינתן קבוצה S, נבחר חוק שמחלק (*partition*) את S ל- $A, B \subseteq S$ שממזער את:

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

כי אם האנטרופיה של עלה נמוכה, זה אומר שהוא יותר אחיד. אם יש עלה שבו כל הנקודות אותו תיוג, יש לו אנטרופיה 0.

האלגוריתם הזה לא נותן לנו הבטחה לגבי גודל העץ.

## גיזום - *pruning*

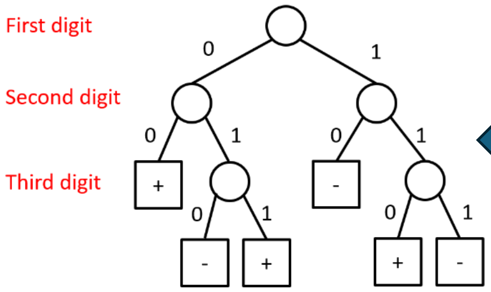
דרך לקבל עץ קטן. אפשר לעשות אחרי אלגוריתם brute-force או ID3.

נתחיל מלמטה, ונמחק עלים. האפשרויות:

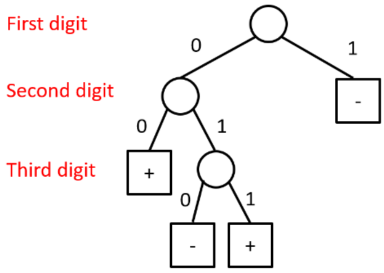
1. לא לשנות כלום.
2. להפוך קודקוד פנימי לעלה.
3. להחליף את הקודקוד בתת-עץ.

דוגמה:

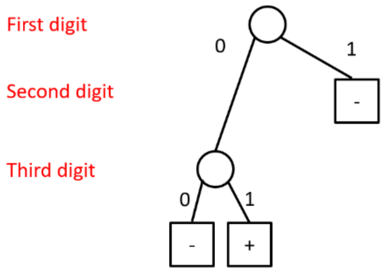
נתחיל מלמטה. עבור העלים
והצמתים ברמה לפני
האחרונה, לא נשנה כלום (כי
החלוקה שווה, אנטרופיה
מינימלית בכל עלה)




נתבונן בבן הימני של השורש:
נשים לב שיש מתחתיו יותר –
מאשר +. אז נהפוך אותו
לעלה –




נתבונן בבן השמאלי של
השורש. נוכל להחליט שתת
העץ הימני שלו יותר
משמעותי, ולשמור רק אותו




**יער אקראי**

מודל מאוד חזק, שעובד בצורה טובה בהרבה מקרים. נחשב המודל state-of-the-art.

מייצר הרבה עצים בצורה רנדומית, ולכל וקטור, לוקח את הבחירה לפי הרוב. זו שיטה טובה לסנן רעשים.