

# סדנת Machine Learning

## מפגש שלישי - Classification and Tree-Based Methods

חלק מהשקרים נלקחו מהספר:

"An Introduction to Statistical Learning"

© אברהם עני



## בפעם הקודמת עסקנו בבעיות סיווג

- הציגנו את מודל הרגרסיה הלוגיסטיית
  - ניתחנו את המבחנים הסטטיסטיים והרחיבנו את המודל למגוון משתנים
- ראיינו כיצד לבחן את טיב המודל
  - דיברנו על ROC-Curve ואילו בעיות עלולות לצאת
- הציגנו את מודל Naïve Bayes
  - דיברנו על היתרונות והחסרונות שלו אל מול הרגרסיה הלוגיסטיית
- הציגנו את מודל KNN
  - דיברנו על נקודות חשובות בבניית המודל (בחירה K, משקל, יתרונות וחסרונות)

# Maximal Margin Classifier

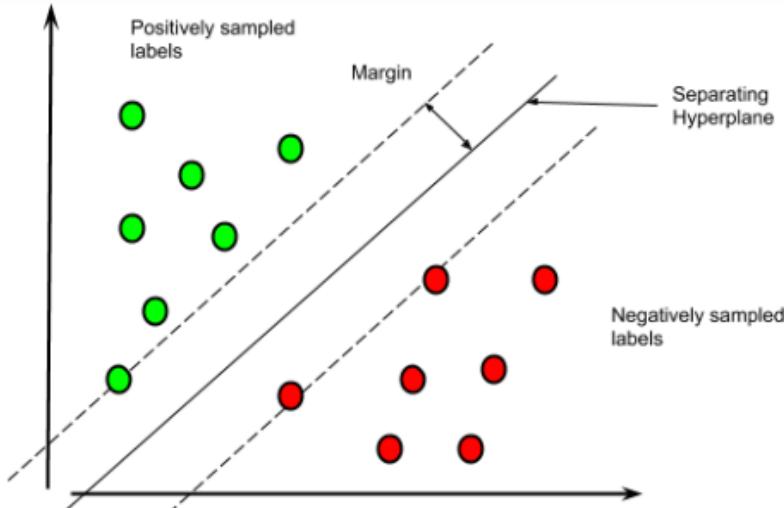


היגשה Maximal Margin Classifier היא גישה סיווג שהגיעה (בניגוד לרוב השיטות שראינו בקורס הקודם) מתחומי מדעי המחשב.

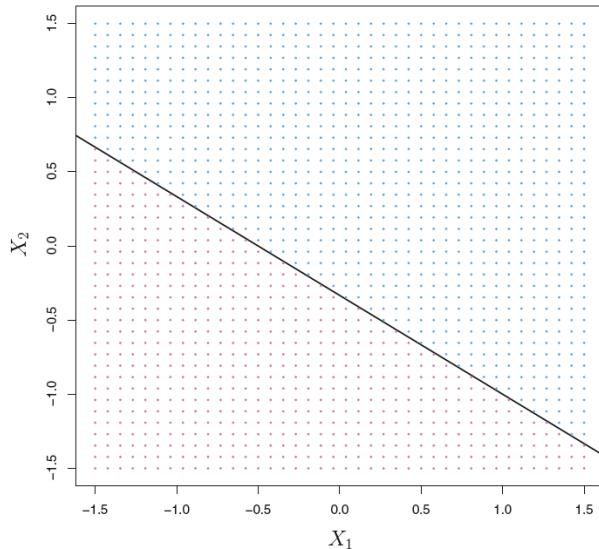
הגישה המוכללת והמוכרת של שיטה זו היא-SVM Support Vector Machines

MMC מבסס על הנחה פשוטה של הפרדה ביןארית בין הנתונים (הנחה שב"כ איןנה עובדת)

על מנת להבין את העיקרון העומד מאחורי הגישה זו נגידיר תחילה מספר דברים.



# Maximal Margin Classifier



זהו דוגמא לעל-מישור מפריד במרחב דו-ממדי.  
על הישר מקיימת:  $1 + 2X_1 + 3X_2 = 0$

נניח ואנו במרחב בעל  $P$  ממדים, על-מישור עבור מרחב זה יהיה בעל  $P-1$  ממדים.

למשל: עבור מרחב של 2 ממדים, על מישור עבור מרחב זה יהיה בעל מימד אחד, או במילים אחרות – קו.

ההגדירה המתמטית:  $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$

או עבור הגרסה המוכללת:  $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p = 0$

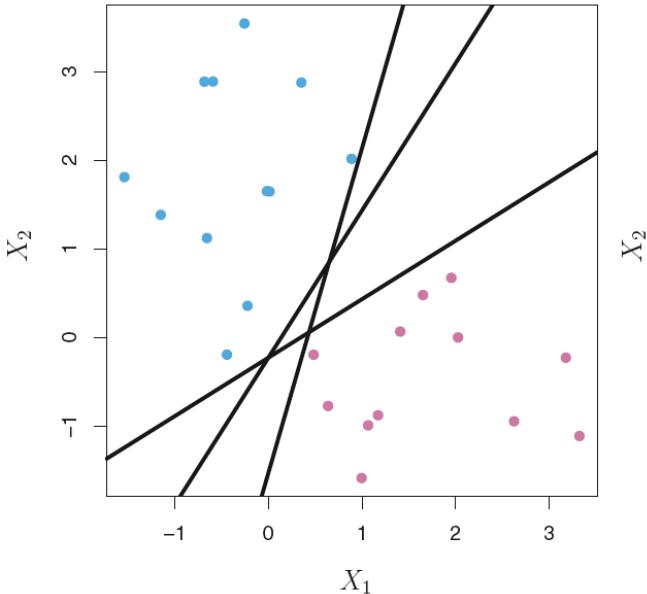
כאשר כל הנקודות המקיימות את המשוואה הנ"ל נמצאות על הקו.

כאשר הנקודה מקיימת –  $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p > 0$

אדיה נמצאת מעל הקו (ולהperf)

מסקנה:  
העל-מישור מחלק את המישור ל 2 חלקיים נפרדים כאשר לכל נקודה ניתן לחשב את הצד בו היא נמצאת

# Maximal Margin Classifier



על-מישור (Hyperplane)

כעת, נוכל להשתמש בתוכנה שמצאנו על מנת לבנות מסויים.

נניח ובידינו מטריצה של  $X$  של  $n$  דוגימות ו-  $m$  מאפיינים (פיצרים) כאשר כל דוגימה משתתפת לסיווג מסוים  $\{y_1, \dots, y_n\} \in \{-1, 1\}$ .

בנוסף, יש לנו דוגימות מבחן.

מה נרצה לעשות?

אנו ננסה לבנות על-סמן מודגם האימון שלנו, על-מישור שיפריד את הדוגימות החיוביות והשליליות.

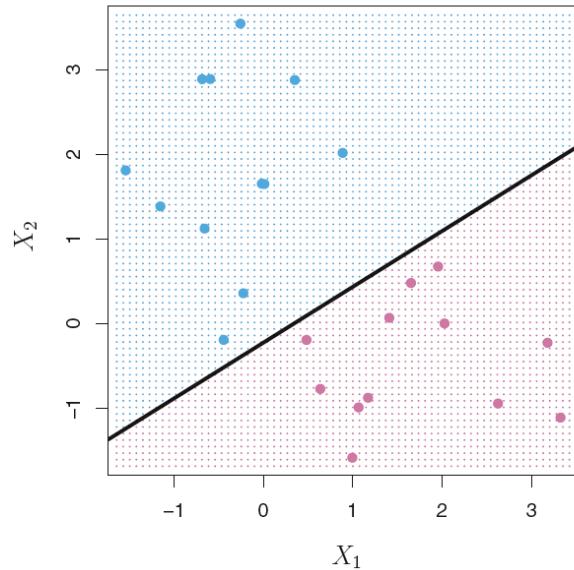
נסמן את הדוגימות החיוביות בכחול ואת השליליות באדום. נשים לב שאנו יכולים לקבל אין-סוף מישורי-על מפרידים כמו באIOR:

# Maximal Margin Classifier



המשר

הישר שלנו מקיים:



$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} > 0 \text{ if } y_i = 1$$

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} < 0 \text{ if } y_i = -1$$

או בצורה מוכללת:

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) > 0$$

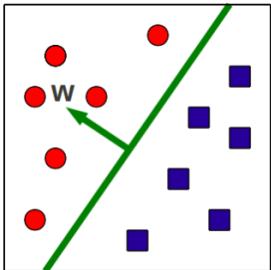
בהתאם לתיאור זה, בהגיע דוגמה חדשה נסואג אותה על סמך המאפיינים  
שליה לסייע המתאים עבורה.

ניתן גם להשתמש במרקח הדוגימה מן הישר על מנת לקבל רמת ביטחון  
בסיווג שלנו – ככל שהדוגימה יותר רוחקה מן הקו כך אנו בטוחים יותר  
בסיווג שלו

# Perceptron



כיצד נוכל למצוא ישר המקיים את הפרדה הליינארית?



## אלגוריתם ה-Perceptron:

- אחד האלגוריתמים הראשונים לסיווג בינארי (רוזנבלט, 1958)
- מבוסס על מציאת על-מישור מפריד בין הדגימות
- מובטח להתכנס לפתרון במידה וישנה הפרדה לינארית
- האלגוריתם הוא מסוג Online-עובר על דוגמה אחת בכל פעם ולא על כל הdataset
- עבד בצורה איטרטיבית:
  - משאיר את המשקלות במידה והסיווג של הדוגימה הנוכחית נכון
  - מעדכן את המשקלות במידה וישנה טעות בסיווג
- עובר על כל הנתונים עד להתכנסות

# Perceptron



פה השתמשנו בסימון מעט שונה:

$$y = \text{sign}(\mathbf{w}^T \mathbf{x} + b) = \text{sign}(\sum_{j=1}^D w_j x_j + b)$$

המשקלות מסומנים ב-w  
החותך מסומן ב- b

שים לב:  $\text{sign}(\mathbf{w}^T \mathbf{x}_n + b) \neq y_n$



$$y_n(\mathbf{w}^T \mathbf{x}_n + b) < 0$$

1. אנו מניחים שבידינו N דוגמאות מהצורה:  $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$

2. אנו מתחילה את המשקלות:  $\mathbf{w} = [0, \dots, 0]$

3. חזר עד להתקנות:

עבור  $n$  בתחום  $\{1..N\}$ :

אם:  $y_n(\mathbf{w}^T \mathbf{x}_n + b) < 0$  - כלומר, הדוגמא מסוגת בצורה לא נכונה

נעדכן את המשקלות בצורה הבאה:

$$\mathbf{w} = \mathbf{w} + y_n \mathbf{x}_n$$

$$b = b + y_n$$

אחרת: השאר את המשקלות הנוכחיים

תנאי עזירה

- כל הדוגמאות מסוגות בצורה נכון (Overfitting לא ייגרם כלל)

- עזירה באיטרציה מסויימת או כאשר תנאי מסויים מתקיימים (לדוגמא: מס' שינויים בעבר על כל הנתונים)

# Perceptron



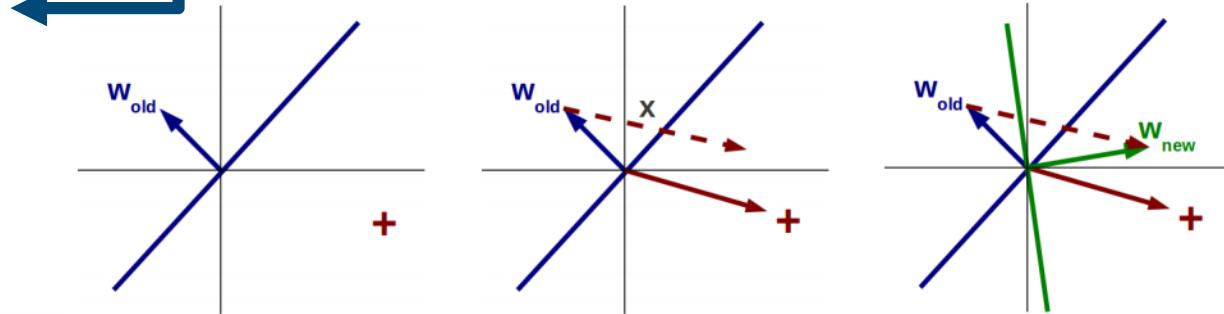
- איך זה עובד בפועל?

נניח ובידינו דוגמה שתוצאה חיובי:  $y_n = +1$  אך סיווגה באמצעות הuko הוא שלילי:  $\mathbf{w}_{old}^T \mathbf{x}_n + b_{old} < 0$   
העדכנים שנבעצע (בהתאם לשיקוף הקודם) הם:  $\mathbf{w}_{new} = \mathbf{w}_{old} + y_n \mathbf{x}_n = \mathbf{w}_{old} + \mathbf{x}_n$  ונשים לב שהנו מוסיפים מכיוון ש- $y$  חיובי!  
 $b_{new} = b_{old} + y_n = b_{old} + 1$

$$\begin{aligned}\mathbf{w}_{new}^T \mathbf{x}_n + b_{new} &= (\mathbf{w}_{old} + \mathbf{x}_n)^T \mathbf{x}_n + b_{old} + 1 \\ &= (\mathbf{w}_{old}^T \mathbf{x}_n + b_{old}) + \boxed{\mathbf{x}_n^T \mathbf{x}_n + 1}\end{aligned}$$

ניתן לראות שהbeitovi החדש שקיבלנו הוא פחות שלילי וכן יסוווג את הדוגינה "הבעייתית" כחיובי (שינויו את המשקولات כך שייתאים לדוגינה)

כמובן שהדברים עובדים בצורה דומה (פשטוט הפוכה)  
 גם בדוגמה שלילתית שתוצאה כחיובי



# Perceptron



• כאשר הדגימה מסווגת בצורה נכונה השגיאה היא – 0  
• כאשר הדגימה מסווגת בצורה שגויה השגיאה גדולה מ-0

$$E(\mathbf{w}, b) = \sum_{n=1}^N \max\{0, -y_n(\mathbf{w}^T \mathbf{x}_n + b)\}$$

• אם נבצע Stochastic Gradient Descent עם הפונקציה הנ"ל נקבל את אלגוריתם Perceptron

## • шиפורים

- האלגוריתם מייצר המון ווקטורי ש שונים לארך שלב האימון
- בזמן הבדיקה על נתונים המבחן, אנו משתמשים בווקטור הסופי
- יתכן ויש לנו ווקטורים מאד טובים באמצעות האימון וכן מאבדים אותם בשל טעות אחת (outlier)?
- אנו יכולים להשתמש בגרסאות המשמשות בווקטוריים אחרים
  - Voted Perceptron – אנו נסוויג על דגימה בהתחשב בסיווג של רוב הווקטוריים ואנו נסוויג דגימה בהתחשב בסיווג של ממוצע הווקטוריים
  - Averaged Perceptron
  - Multiclass Perceptron – שימוש בסיווג למספר מחלקות (יצירת מספר וקטורי A ש"יחתכו" אף בחרחבה)

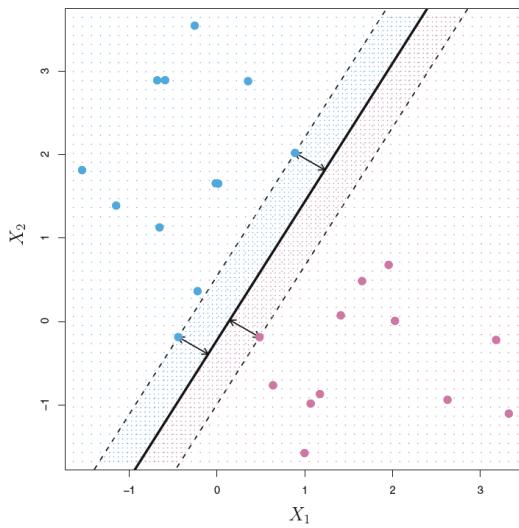
? averaged voted ?  
? ? ?

# Maximal Margin Classifier



ראינו כי ה-Perceptron מייצר לנו על-מישור אחד המפריד בין הדגימות השליליות והחיוביות. נזכיר כי ישנו איןסוף כאלו.

Maximal margin classifier



כיצד נבחר את על-המישור האופטימלי?

מבחן אינטואיטיבית נחפש את על-המישור שמזער ככל שנייתן את המרחק מכל הנקודות. כך קיבל מרוח (margin) מקסימלי מתוך הנחה שמרוח כזה יתאפשר יותר טובה לדגימות חדשות.

ניתן לשים לב שישנן 3 דגימות על "גדר" המרוחות. נקודות אלו חשובות וידועות בשם: **Support vectors**

למעשה, המישור-על האופטימלי נקבע בק באמצעות דגימות אלו, כך שאם למעשה "נviz" דגימות אחרות מישור-העל לא ישתנה.

זהו תכונה חשובה ונדון בה בהמשך.

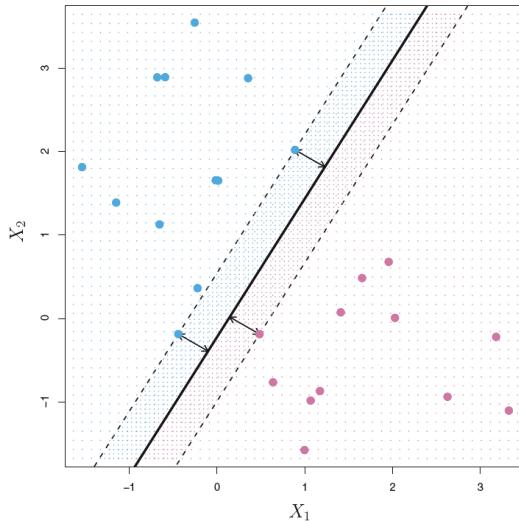
# Maximal Margin Classifier



איך נמצא את העל-מישור האופטימלי?

אופטימיזציה!

Maximal margin classifier



$$\underset{\beta_0, \beta_1, \dots, \beta_p, M}{\text{maximize}}$$

$$\text{subject to } \sum_{j=1}^p \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n$$

הסבר:

אנו מנסים לאפטם את מיקום העל-מישור כך שכל הדגימות ימוקמו מצד הנכון ובמרחק  $M$  (כמה שייותר רחוק) לפחות מן העל-מישור המפריד.

הגבלת הראשונה באה כדי לנרמל את המשקלים של המקדמים\* והגבלה השנייה באה כדי לחייב את כל הדגימות להיות במרחק  $M$  לפחות מעל-המישור.

בעיה זו ניתנת לפתרון אналיטי אך פתרונה מעבר לחומר הנלמד בסדנה

# Support Vector Machines



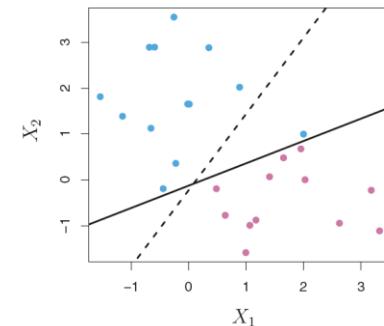
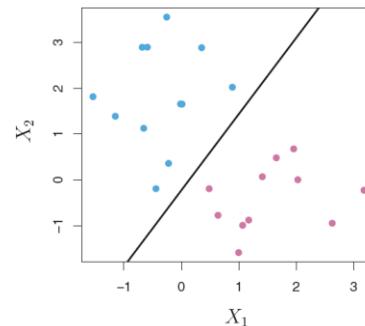
מה יקרה כאשר אין הפרדה לינארית בין הנתונים?

כאן, בעיית האופטימיזציה לא תפתר לנו בצורה ייעילה את הבעיה, ואילו ה-Perceptron כלל לא יוכל לפתורנו! במקרה זה, אנו נבצע הכללה לבעיית האופטימיזציה שראינו קודם.

נשים לב: גם במקרה בו הנתונים (במבחן האימון) ניתנים להפרדה אנו נבחר בדרך-כלל להשתמש בשיטות כלליות יותר על מנת ליצור מסווג "עמיד" ומכליל יותר.

במקרה הבא, מצד השמאלי קיבלנו קו מפריד אופטימי למדגם האימון, מצד הימני זהו אותו מדגם רק שהוספנו דגימה כחולה חדשה. ניתן לראות כיצד המסווג משתנה בצורה דרמטית עקב הוספה של **דגימה אחת**

זהוי כמובן תוכנה בעייתית שמצביעה על נטייה **Overfitting** וANO מעוניינים למנוע אותה!



לכן, נחפש פתרון שאומנם יכול "לטעות" בדגימות מסוימות, אך יספק לנו את תכונות הייציריות והגריללה

# Support Vector Machines



## Support Vector Classifier

זהו מסויוג שלא מנסה למצא ישר מפריד מושלם אלא מנסה לאזן בין סיווג נכון של רוב הדוגמאות לבין הכללה לדוגמאות חדשות. לעיתים הוא נקרא Soft Margin Classifier (מכיוון שהוא "מרשה" טעויות) בניגוד ל MMC שנקרא Hard Margin Classifier.

$$\underset{\beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n, M}{\text{maximize}} \quad M$$

subject to  $\sum_{j=1}^p \beta_j^2 = 1$ ,

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^n \epsilon_i \leq C,$$

התיאור שלו (בדומה לתיאור של MMC) הוא דרך בעיית אופטימציה:  
ניתן לראות הבעיה דומה מאוד לבעיית ה-C-MMC למעט השימוש ב-  
א. ו- C (שבהמ ניגע מיד)

גם כאן אנו מנסים למקסם את המרווח (Margin) בין שתי המחלקות אך אנו מתירים פה חריגות באמצעות המשתנים שהוזכרו.

לאחר שפתרנו את משווהת האופטימיציה, נקבל דגימה חדשה ולפי המשקولات שמצאנו נסוג איתה בהתאם לצד שלה ביחס לעל-מישור

# Support Vector Machines

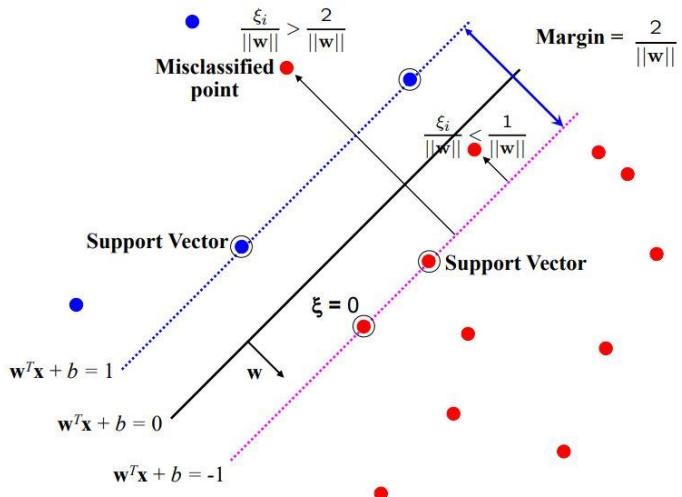


$$\underset{\beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n, M}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^p \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^n \epsilon_i \leq C,$$



## Support Vector Classifier

המשתנה  $\epsilon$ :

זהו משתנה המציין את החריגת של הדגימה  $i$ -th (slack variable).

אם  $\epsilon_i = 0$ : הדגימה נמצאת בצד הנכון של המרווה

אם  $0 < \epsilon_i$ : הדגימה נמצאת בצד הלא נכון של המרווה (אך עדין בצד הנכון של העל-מישור)

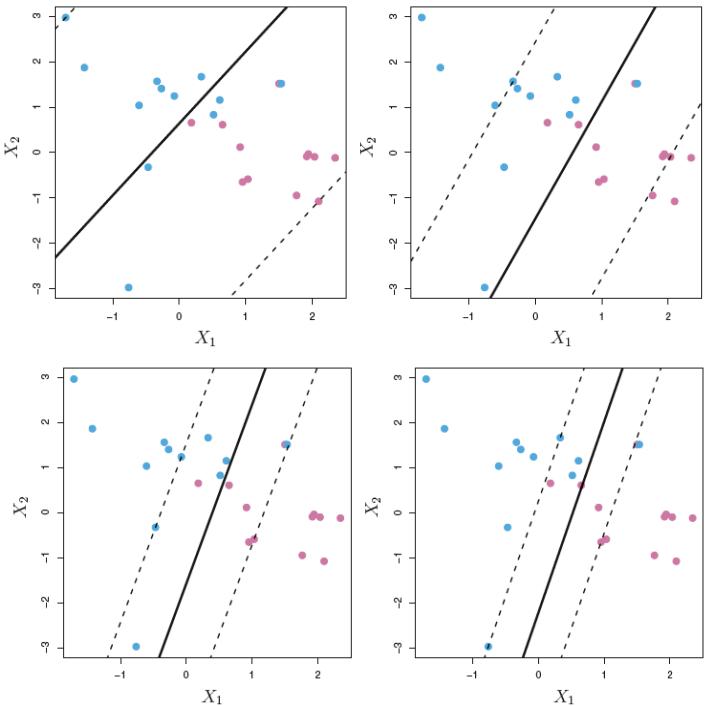
אם  $1 > \epsilon_i$ : הדגימה נמצאת בצד הלא נכון של העל-מישור

# Support Vector Machines



$$\begin{aligned} & \text{maximize}_{\beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n, M} M \\ & \text{subject to} \quad \sum_{j=1}^p \beta_j^2 = 1, \\ & y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \\ & \epsilon_i \geq 0, \quad \sum_{i=1}^n \epsilon_i \leq C, \end{aligned}$$

ניתן לראות כיצד  
הערך של  $C$  משנה  
את רוחב המרווה



## Support Vector Classifier

### המשתנה $C$ :

$C$  הוא משתנה החוסם את סכום החריגות של כל הדוגמאות, ובעצם מגדר את רמת החריגות המותרת. זהו פרמטר הנitin לשיליטה ונקבע לפי סוג הנתונים והבדיקה בפועל.

אם  $C=0$ : אז אנו מקבלים בעית MMC רגילה (אנו דורשים שלא יהיו חריגות) לעומת זאת, ככל ש- $C$  עולה אנו מקבלים קו "גמיש" ומכליל יותר



מה זה מזכיר לך?

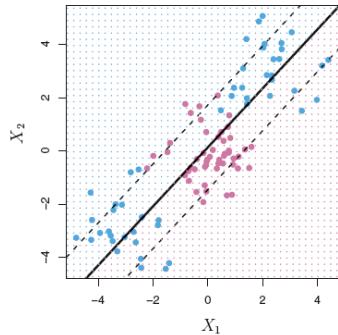
# Support Vector Machines



## Support Vector Classifier

- תכונה מעניינת של SVM היא שהוא תלוי רק בדוגמאות החוצות את המרווה ואיננו מושפע מהמיוקם של הדוגמאות המשווגות לצד המרווה הנכוון.
- נוכל להבין, כי השליטה על פרמטר C קובעת את מספר הדוגמאות המוגדרות כ-Support Vectors. ככל ש-C גדול יותר נקבל יותר דוגמאות ש"חוצות" את המרווה ומוגדרות כ-SVs ולהפך.
- העיקרון דומה ל-Bias-Variance Tradeoff שאותו הזכרנו במאגרים הקודמים. ככל ש-C גדול יותר כך על-המשיר נקבע בידי יותר דוגמאות ולכן בעל Variance נמוך יותר אך נשלם על כל ב- Bias גבוה ולהפך.
- העובדה שהמסווג נקבע ע"י מספר בודד של דוגמאות וכל שאר הדוגמאות אינן משפיעות יוצר מסווג "עמיד" יותר מאשר מסווגים המתחשבים בכל הדוגמאות

# Support Vector Machines



מה נעשה כאשר הפרדה היא איננה לינארית לחלווטין?

אם נסתכל על המקרה הבא:

נוכל להבין שבמקרה זהה היעילות של SVC איננה גבוהה...

כבר נתקלנו במרקם כאלו בעבר (מפגש הסדנא הראשון) כאשר עסקנו ברגRESSED לינארית.

הצנו פתרון למרקם כאלו בשימוש ברגRESSED פולינומיאלית – שימוש במשתנים הקיימים אך בתצורה ריבועית (בשלישית וכו'). כך שאומנם הרגRESSED נותרת לינארית אך היא מבוטאת בצורה שאיננה לינארית וכך נותנת גמישות שיכולה לשיע בתחום דודות עם הפרדה (או ערך) שאינו לינארי.

גם כן נוכל להשתמש בעיקרן זהה על מנת להתמודד עם חוסר הלינאריות.

נניח ובידינו נתונים בעלי  $\mathbb{P}$  פיצרים:  $X_1, X_1^2, X_2, X_2^2, \dots, X_p, X_p^2$  נוכל לייצג אותם בצורה הבאה:

נקבל שהעל-מישור המפריד איננו קו אלא פולינום ריבועי שמקיים:  $0 = q(x)$  لكن קיבלנו גבול שאיננו לינארי. ישן כמובן אפשרות ליצירת משתנים כאלה

הבעיה בפתרון זה שהוא יטה ליצור המון משתנים זה יהיה בלתי ישים חישובית לחשב את הפרדה. לכן נראה מספר טרנספורמציות שימושות את הפיצרים ומערכות אותן מן הלינאריות למרחבים שונים וגם שומרות על "

# Support Vector Machines



## SVM

SVM הוא למעשה הכללה של SVC המאפשרת להרחב את ייצוג הפיצרים באמצעות פונקציות הנקראות Kernels. העיקרון העומד מאחורי רעיון זה הוא האפשרות לייצג את הקו המפריד בייצוגים שאינם לינאריים ויאפשרו הפרדה טובה יותר של הנתונים.

ניתן לייצג את על המישור המפריד באמצעות:

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^p x_{ij} x_{i'j} \quad \text{כasher, } f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle$$

בහינתן וקטור דגימה חדשה  $X$  אנו נחשב את הסיווג לפי הפונקציה זו.

i - משתנה הקובע את חשיבות הדגימה ה $i$ 'ה במסווג, מוחושב לכל דגימה.

המשתנים  $\alpha$  ו-  $\beta_0$  נקבעים לפי דוגמאות האימון (מחשבים  $\binom{n}{2}$  מכפלות סקלריות של כל הוקטורים בין עצמם)

מכיוון שהוא יודיעים שבפועל רק הדוגמאות המוגדרות כsupport vectors משפיעות על המסווג יוכל לרשום:

-כasher S מצין את קבוצת הדוגמאות שהן SP (מוריד משמעותית את רמת הסיבוכיות)

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle$$

# Support Vector Machines

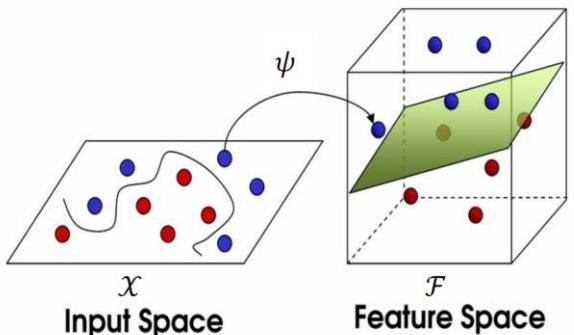


## Kernel Trick-SVM

از מה עוזר לנו הייצוג שקיבלנו?

cut במקום לייצג את על-המישור באמצעות פונקציה לינארית  
(מכפלה סקלרית) נוכל להשתמש בפונקציות שונות.

נדירים:



- פונקציה כללית לתיאור הקשר בין וקטורי הדגימות ונקבל:  $f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)$

נוכל להשתמש ב Kernel לינארי:  $K(x_i, x_{i'}) = \sum_{j=1}^p x_{ij}x_{i'j}$  - זהה המכפלה הסקלרית

או ב Kernel פולינומי:  $K(x_i, x_{i'}) = (1 + \sum_{j=1}^p x_{ij}x_{i'j})^d$  - בדומה לרגרסיה פולינומיאלית אפשר לשחק ב"דרגה"

או ב Kernel רדיאלי:  $K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})^2)$

# Support Vector Machines



## SVM

cut באמצעות kernels נוכל לאמן את ה-SVM.

נחזיר לנוטונים בהם SVC נכשל ביצירת מסגר מוצלח:

נוכל לראות בצד שמאל SVM עם Kernel מסוג פולינומיAli מדרגה 3 ואילו מצד ימין נוכל לראות Kernel רדיאלי.

נשים לב כי שניהם מצליחים במקרה זה להציג הפרדה מאוד טובה של הנוטונים

בין-air Kernel הרדיאלי עובד:

מגיעה דוגמה חדשה  $x^*$ , אם המרחק בין דוגמה מסוימת האימון  $x^* = (x_1^* \dots x_p^*)^T$  גודל איזי ההשפעה שלו  $\exp(-\gamma \sum_{j=1}^p (x_j^* - x_{ij})^2)$  תהיה קטנה



זה אומר Kernel רדיאלי מתנהג בצורה מאוד מקומית (מציר KNN) נזכיר שסיווג הדוגמה קבוע לפי היסויון של  $f(x^*)$  מה המשמעות?

# Multiclass SVM



עד כה עסקנו במקורה הבינארי בו אנו נדרשים לשווג ל- $\{1 - 1\}$

ומה אם יש לנו מספר מחלקות?

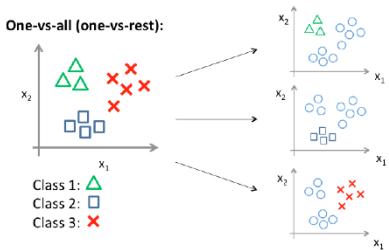
ההרחבות למקרים كالו אינם טריוויאליים, וsono נציג 2 פשوطות יחסית:

## One-Versus-One Classification •

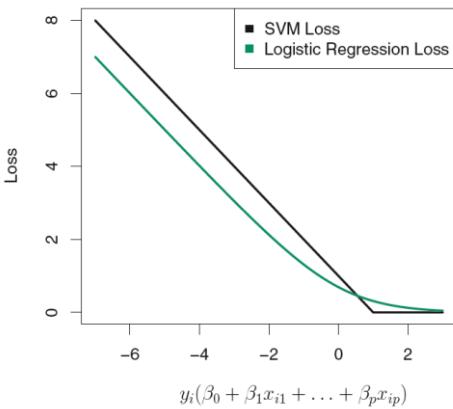
אם יש לנו  $K$  מחלקות, נבנה  $\binom{K}{2}$  מסוגי SVM, אחד עבור כל זוג מחלקות. נסואג כל דגימת מבון בכל המסוגים שבינו ונספור את המחלוקת שקיבלה הכי הרבה "ניצחונות"

## One-Versus-All Classification •

אם יש לנו  $K$  מחלקות נבנה  $MV$  כאשר בכל פעם נבחר מחלקה אחת ונציג מולה (כתיגר השני) את כל שאר המחלקות. נקבל עבור כל מחלקה וקטור- $\beta$   $\beta_{pk}, \beta_{1k}, \dots, \beta_{0k}$ , אשר נקבל דגימת מבחן, נמצא את וקטור המחלקה הממקסם את התוצאה (כל שהתוצאה גבוהה יותר  $\leftarrow$  אנו יותר בטוחים בתוצאה)



# Logistic Regression and SVM



ניתן לראות את הבדל בין הגרפים של פונקציית  $\text{-Loss}$  –  $\text{SVM}$  לבין הפונקציה "קשייה" בהתאם לעובדה שננקודות שאינן SV אינן משפיעות. ואילו ב- $\text{LR}$  ניתן לראות את החלקה מסוימת

- כשהופיע בתחילת ה- $\text{SVM}$  (בשנות ה-90), החוקרים חשבו כי הוא שיטה מיוחדת ו שונה מאשר השיטות שהיו ידועות בזמןו ( $\text{LDA}$ ,  $\text{Logistic Regression}$  ועוד). מזמין, נמצא קשרים בין השיטות השונות אשר עוזר להבין את האופן המוחיד בו כל שיטה עובדת
- מסתבר, שאות הקритריון לקביעת  $\text{SVM}$  (בעיית האופטימיזציה) ניתן לרשום בצורה הבאה:
$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{minimize}} \left\{ \sum_{i=1}^n \max [0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^p \beta_j^2 \right\}$$

Loss function (Hinge-Loss)      Penalty function

- ה- $\lambda$  הוא גורם דומה ל- $C$  בכיר שהוא "שלוט" ברמת החירוגות המותרות
- ניתן לרשום זאת בצורה כללית:  $\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{minimize}} \{ L(\mathbf{X}, \mathbf{y}, \beta) + \lambda P(\beta) \}$
- ההבדל בין  $\text{LR}$  לבין  $\text{SVM}$  נועד בפונקציית  $\text{-Loss}$  –  $\text{Hinge-Loss}$  מול  $\text{Log-Loss}$  שלhn (Shallow) ביחס לפונקציה  $f(x^*)$  המתארת את התיאוג
- בפועל, ברוב המקרים התוצאות של  $\text{RL}$  ו-  $\text{SVM}$  תהינה דומות עם יתרון קל ל- $\text{SVM}$  אך במקרים בו הנתונים מעורבים דוחק ל-  $\text{LR}$  יהיה יתרון (עקב החלוקת)

# Tree-Based-Models



- ראיינו שיטות שונות לרגסיה וקלסיפיקציה כאשר לכל שיטה יתרונות וחסרונות בתחוםה שלה. מובן שישנן עוד הרחבות ושיפורים לכל המודלים שראינו, ויישום בפועל יידרש להטמת המודל לנתחים עליהם אנו עובדים
- שיטה נוספת השייכת הן לקלסיפיקציה והן לרגסיה היא שיטה מבוססת עץ. היתרון המרכזי בשיטה זו הוא יכול ההציגה (representation) של התוצאות ומרכיבי המודל אם כי היא נוחותה ביכולת הביצועים לעומת שיטות מתקדמות יותר (שאתן חלקן ראיינו)
- בתחילת נושא מודל פשוט הנקרא: עץ החלטה, נעבור על דוגמא ונבין את תהליכי הבנייה של עץ כזה
- בהמשך, נזכיר מגוון שיפורים למודל פשוט (Boosting, Bagging ועוד). שיתרונם הוא בשיפור איכות המודל וחסרונו בפגיעה ביכולת ההציגה שלו.

# Decision-Tree



## דוגמה

נניח ובידינו הנתונים הבאים על שחקני הבייסבול בליגת ה-MLB:

1. מספר עונות בליגות הבכירות (Years)
2. מספר חבטות מוצלחות בעונה החולפת (Hits)

אנו מעוניינים בהתאם לנ נתונים אלו למצוא מודל החוצה את המשכורת של השחקן (Salary)-אנו נניח שהמשכורת מבוטאות ב-log-scale)

כיצד היינו נגשים לבעה זו על בסיס המודלים שלמדנו?



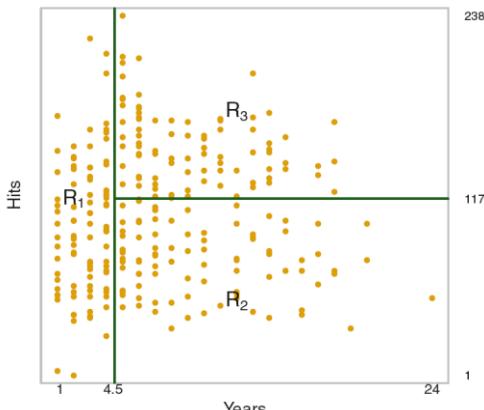
# Decision-Tree



אם נريץ מודל פשוט של עז החלטה נקבל את הדיאגרמה הבא:

כניתן לראות כי בתחילת, אנו מבדילים בין שחקנים עם ותק של מעל 4.5 שנים לבין אלו עם פחות מאשר האחרונים מרוביים, על פי ממוצע של השחקנים באזרע זה,  $e^{5.11} \times 1000$  (לא רע בכלל!)

לאחר מכן, אנו מקבלים התפלות לפי מספר החבוטות בעונה החולפת.



ניתן לחושב על כר-על חלוקה ל-3 אזורים בצורה הבא:

$$R_1 = \{X \mid \text{Years} < 4.5\}$$

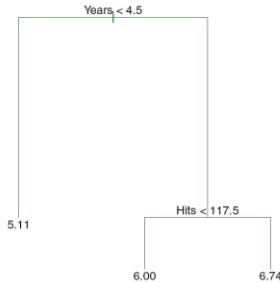
$R_2 = \{X \mid \text{Years} >= 4.5, \text{Hits} < 117.5\}$

$R_3 = \{X \mid \text{Years} >= 4.5, \text{Hits} >= 117.5\}$

# Decision-Tree



## מנחים



**צמתים פנימיים**- המחלקים של העץ, אצלנו ישים **שנים שניים**  $\text{Hits} < 117.5$  ו-  $\text{Years} < 4.5$

**עלים**- אזוריו החלוקה, אצלנו אלו:  $R_3$ ,  $R_2$  ו-  $R_1$

ניתן להסביר את העץ בצורה הבאה:

ישנו הבדל מהותי ברמת השכר בין שחקנים בעלי ותק של 4.5 ויתר לבין השאר. בנוסף, לשחקנים שאינם בעלי ותק, מספר החבותות אינו גורם משפיע במיוחד על רמת השכר, לעומת זאת השפעה שיישנו לגורם זה על השחקנים הוותיקים.

כמובן שמודל זה הוא פשוטי ביחס למודלים מורכבים אך ניתן לשימוש לבלייטרונות שבמצג הפשוטה והנוחה שלו על פניו.

# Decision-Tree



از איך בונים עץ כזה?

שני צעדים מרכזיים:

1. מחלקים את מרחב הפיצרים ל-J אזורים שונים
2. מבצעים פרדיקציה עבור דוגמה שנמצאת באזור מסוים באמצעות המוצע של הדגימות באזור שלא

איך נחלק את המרחב ל-J אזורים שונים?

הינו מעוניינים ליצור מעין קופסאות רבי מידות (שאין חופפות) כך שנמצער את ההפרש בין התוצאות באותו אזור לבין

$$\text{הערך הנוכחי: } \sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

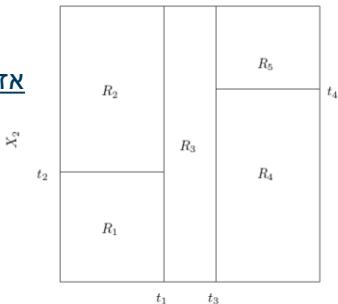
האם זה אפשרי?

לא כל כך ...

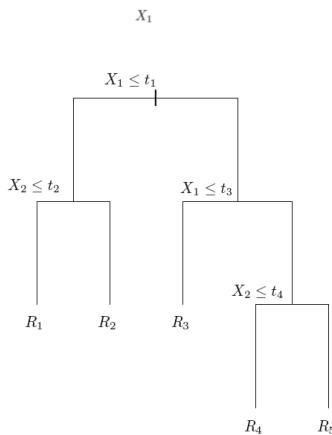
# Decision-Tree



## אזור החלוקת



## עץ החלוקת



از נעשה בכל זאת?

עובד בצורה חמדנית, שנקראת **recursive binary splitting**.

נתחיל בניסיון לחלק את המרחב ל 2 חלקים באמצעות אחד הפיצרים (כאשר בפועל אנו בוחנים לכל פיצר את כל החלוקת האפשרות).

## מבחן מתמטית:

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

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

לאחר מכן, נעבור שוב על כל הפיצרים וכל החלוקת האפשרות ונחלק את אחד מהתאי האזוריים שקיבלנו ( $R_2, R_1$ ) ל-2 אזוריים (סך הכל קיבל 3 אזוריים), וכך בצורה איטרטיבית עד לחוק התכנסות (למשל: 5 דוגמאות לפחות באזור)

# Decision-Tree



## גיזום

כאשר בפועל נבנה עץ נתקל ככל הנראה בעיית ה-Overfitting (מדוע?), על מנת לפתור בעיה זו משתמשים בגישה **הגיזום** (ישנן גישות נוספות

האינטואיטיבית מאחוריו עיקרונות זה היא שימוש בתת-עץ שיתן לנו תוצאות מסתיק טובות יחסית אך יוריד את רמת סיבוכיות העץ והפיצולים כך שנישאר עם מודל יציב אך בעל יכולות חיזוי מספקות.

## כיצד נבחר את תת-העץ?

הגישה הנאייבית תהיה לעבור על כל תת-עץ אפשרי ולבחר את האחד שיספק לנו את שגיאת האימון הנמוכה ביותר.

אך ישנו המון עצים אפשריים ← בעיה חישובית!



# Decision-Tree



## גיזום

על מנת להתמודד עם בעיה זו, נשתמש בגישה אחרת. נגידר "קנס" עבור מספר העלים בעץvr שיווצר איזון בין סיבוכיות העץ לבין השגיאה שלו בצורה הבאה:



$$R_\alpha = \sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

$|T|$  - מספר העלים בעץ

$R_m$  - זהה הולה ה- $m$ -ו ועבورو  $\hat{y}_{R_m}$  הוא התחזית (ממוצע הדגימות באותו אזור)

$\alpha$  - פרמטר חיובי השולט ב tradeoff בין מספר העלים לשגיאה.

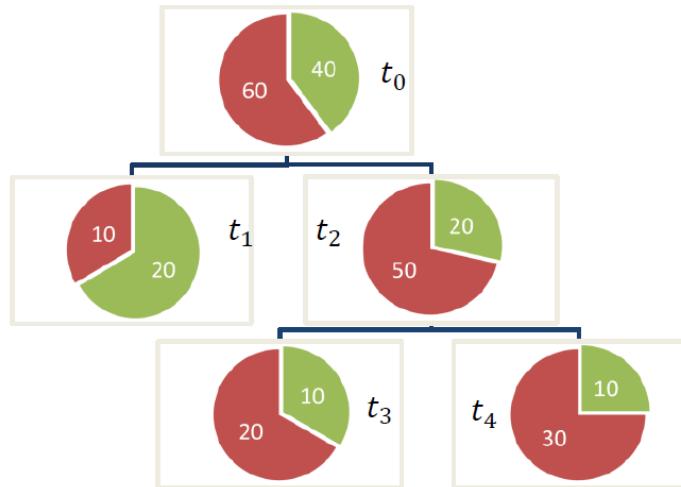
כעת, עבור כל  $\alpha$  יש לנו כל החלטה האם ל哲ום את העץ או שלא. מה שנתקבל מכך הוא בעצם סדרה של תת-יעדים עבור ערכי  $\alpha$  שונים. נוכל להשתמש בשיטות דגימה על מנת לקבוע את ה  $\alpha$  האופטימלי ולבנות את העץ התואם אליו

vr הרווחנו בעצם סדרה מצומצמת של תת-יעדים אותם נוכל לבדוק על נתוני מבחן ולקבוע את העץ הסופי

# Decision-Tree



- $T_0 = Tree(t_0)$  - the full tree



$$R_\alpha(T_0) = \left( \underbrace{\frac{30}{100} \cdot \frac{10}{30}}_{t_1} + \underbrace{\frac{30}{100} \cdot \frac{10}{30}}_{t_3} + \underbrace{\frac{40}{100} \cdot \frac{10}{40}}_{t_4} \right) + \underbrace{3}_{leaves} \cdot \alpha = \frac{3}{10} + 3\alpha$$

# Decision-Tree

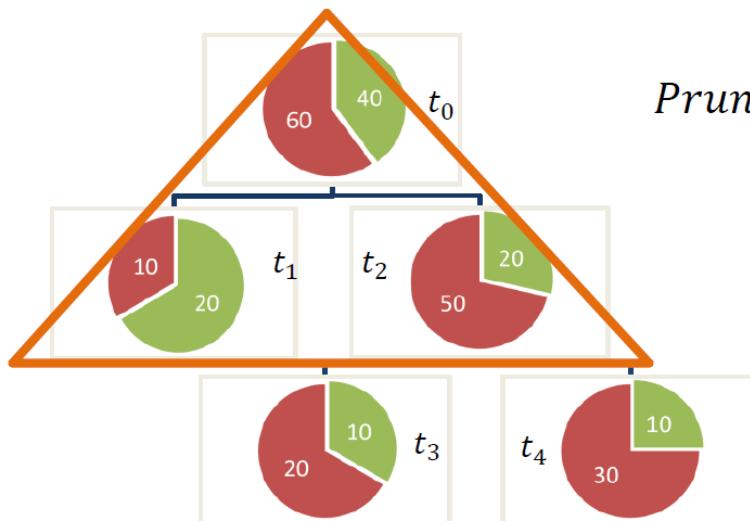


דוגמא

נסמן ב  $Prune(T_0, t_2)$  את העץ המתkeletal כשגוזמים את תת העץ

שורשו  $t_2$  מתוך  $T_0$

$Prune(T_0, t_2)$  הוא עלה ב  $t_2$  -



לכן, נבחר לגזום את  
העץ ונישאר עם  
העליה  $t_2$ .

$$\leftarrow R_\alpha(Prune(T_0, t_2)) = \frac{10 + 20}{100} + 2\alpha < R_\alpha(T_0) = \frac{3}{10} + 3\alpha$$

# Decision-Tree

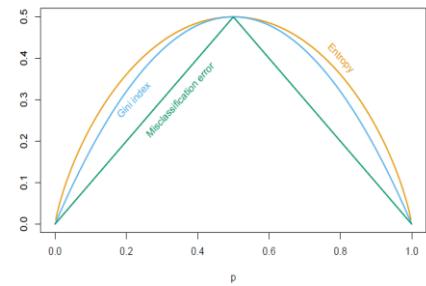


עצי סיג

אם בبنית עץ רגסיה חזינו את הערך בהתאם לממוצע האзор אליו הדגימה שייכת, בעז סיווג הבחירה הבסיסית תהיה המחלוקת אליה שייכות רוב הדגימות באazor.

כיצד נבנה את העץ הסיווג?

*classification* מעץ רגסיה בו השתמשנו ב- RSS , כאן הבחירה הטבעית היא *error rate*



אך מסתבר, כי בניית עץ על פי כלל זה מובילה לעיתים לעץ שאינו מאוזן ולכן מוצעים שני מדדים חשובים אחרים:

$$D = - \sum_{k=1}^K \hat{p}_{mk} \log \hat{p}_{mk} \quad \text{-Entropy} , \quad G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk}) \quad \text{-Gini ττη}$$

שני המדריכים הללו יותר "מוחלקים" מהמדד הבסיסי ונוטנים ביטוי לעיר-טהרה, של העז (יותר דגימות מסוימתן מחלוקת תחת הענף) כאשר "טהרת" הפיצול היא מדד לאיכותו ונוטנת לנו אפשרות לתרגם את הסיווג להסתברות

בבנייה העץ בד"כ נשתמש באחד מהמדדים הללו וכשנבחן גיזום נשתמש במידד הראשוני

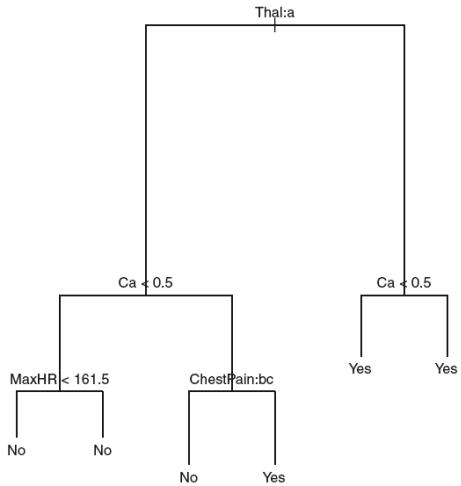
# Decision-Tree



עצי סינוג

- נשים לב ל蹶ה הבאה:

ונכל לראות כי ישנו פיצול (הימני התיכון) שני עליו מקבלים את אותו ערך (Yes), אם כן, מדוע צריך את הפיצול הזה בכלל?



התשובה היא שהפיצול חשוב בבדיקה בגל ממד "הטהרה" שהזכרנו.

ນחשוב על蹶ה בו העלה הימני הוא ( מבחינת המחלקות ) טהור לחלוטין (11/11) והעליה השמאלי איננו (7/11) אם קיבל דגימת מבחן שתסועג לעלה הימני נוכן לומר בבודאות גדולות שהוא אכן שייך למחלקה, בנגדוד למצב בו הדגימה תשאיר לעלה השמאלי.

נשים לב שה- Classification Error אינו משתנה בעקבות חלוקה זו אף ממד Entropy וה- Gini ו- Outliers, זהו חלק מחישיבות השימוש בהם.

# Decision-Tree



## יתרונות

- קל להסברת ולמיושן
- מודל שאינו לינארי
- בעל יכולת הצגה גרפית טובה וברורה
- שליטה טובה במשתנים קטגוריאליים (אין צורך ליצור משתנים מדומים)



+

## חסרונות

- בהמונ מקרים ביצועיו פחות טובים מאשר שיטות יותר עדכניות ומתוחכבות
- מאד רגיש לשינויים בנתונים (שינוי קטן יכול לשנות את כל העץ)

# Bagging



## Bootstrap

זו היא שיטה סטטיסטיות אשר באה לפתרו מקרה בו אנו מנסים לאמוד את השונות של המדגם (לצורך חישובים סטטיסטיים ולמידה ) אך ברשوتינו מדגם מצומצם בלבד.



הפתרון נועז בהרחבת המדגם באמצעות תת-مدגים (מתוך המדגם בעצמו) באותו גודל מקורי כאשר אנו דוגמים מתוך המדגם המקורי עם חוזרות.

שיטה זו יכולה לשמש גם לשיפור שיטות קיימות כמו עצי החלטה. כפי שראינו , עצי החלטה נתונים לשונות גובהה ולכן אנו נשאף להוריד אותה. נבצע זאת באמצעות הכללה של Bootstrap הנקראית .Bagging aggregated Bootstrap

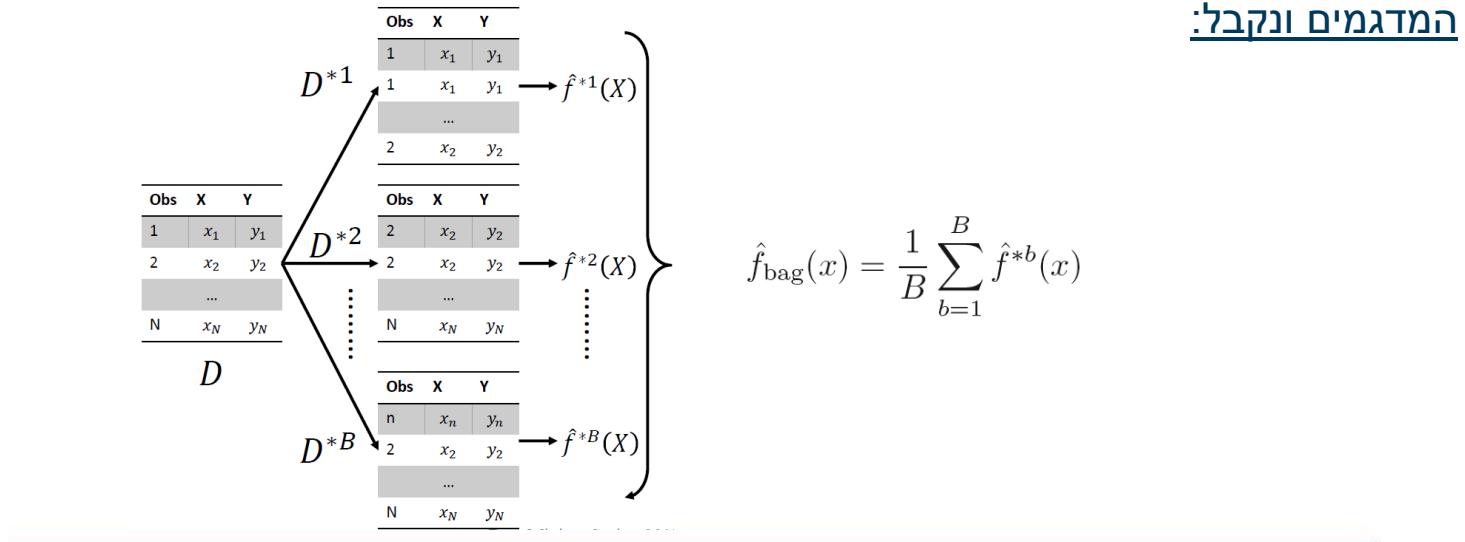
# Bagging



המשב

כפי שאמרנו, בהינתן שיש לנו מדגם  $D$  בגודל  $N$ , אנו נדגמים  $B$  מדגמים כאשר בכל מדגם יהיו  $N$  דוגימות (עם חוירות) מתוך המדגם המקורי.

כל מדגם מתוך  $B$  המדגמים יאומן ונתקבל פונקציה  $\hat{f}^{*b}(x)$  המתארת את אותו המדגם. לבסוף נמצע על פניו כלל המדגמים ונקבל:



# Bagging



בחירה ערך B  
הבחירה איננה קריטית כ"כ (בחירה של ערך גבוה לא טוביל לוג'יסטי) Overfitting  
אך רצוי לבדוק אמפירית עד שmagics עד להתכנסות מסוימת



## Algorithm Decision-Tree Bagging

**קלט:** קבוצת נתונים האימון  $D$ ,  $B$  כמות מדגמי Bootstrap לאgregציה

לכל  $b = 1, \dots, B$ :

הגרל מדגם  $D^{*b}$  בגודל  $N$  (גודל קבוצת האימון המקורי)

בנה עץ החלטה  $T_b$  על בסיס מדגם ה-Bootstrap

**פלט:** אוסף עצים ההחלטה  $T_1, T_B, \dots, T_B$

תחזית לתצפית חדשה  $x$ :

**גرسיה:** ( $\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(x)$ ) היא התחזית לתצפית  $x$  בעץ ההחלטה  $T_b$

**סיווג:** נסמן ב-  $\hat{c}^{*b}(x)$  את התחזית המחלקה בעץ  $T_b$  ואו  $(x)$  יקבע ע"פ הצבעת הרוב על פני כל העצים  $T_1, \dots, T_B$

# Bagging



## Out-of-Bag Error Estimation



זהי השיטה באמצעותה ניתן להעריך את שגיאת המודל. ניתן להסביר זאת בצורה הבאה:  
בכל תת-מבחן ס אנו דוגמים מ- $N$  הדוגימות, בפועל כ $2/3$  מן הדוגימות המקוריות נמצאות  
במבחן ס כך שאנו יכולים להניח כי עבור דוגימה נתונה בערך ב- $1/3$  מן המדוגמים היא איננה  
מושפיעת!

נשתמש בכךון זה ע"מ לעבור על כלל המדוגמים שבהם הדוגינה איננה מושפיעת, ונשתמש בה  
כדגם מבחן. כך נעבור על כלל הדוגימות ונקבל את שגיאת האימון.



אנו מבדים את יכולת ההציגה שהייתה מרכיב משמעותי במודל העז הפשוט עקב הריבוי של  
העצים.

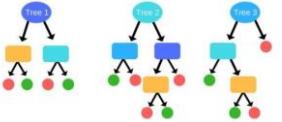
על מנת לחת בכל זאת הבנה מהם הפיצרים המרכזיים המשמעותיים במודל העז הפשוט עקב הריבוי של  
באמצעות בדיקת החלוקות המשמעותיות ביותר ב- $B$  העצים (ביטול שלהם גורם לעלייה ב-SSE  
או Gini) ותרגומם לפיצרים מרכזיים

# Random Forests



מה יכול להשתמש כשןפעיל את אלגוריתם Bagging?

נניח ויש לנו הרבה פיצרים מתוכם ישנו פיצר אחד משמעותי  
במיוחד ועוד מספר פיצרים משמעותיים.



Algorithm 15.1 Random Forest for Regression or Classification.

1. For  $b = 1$  to  $B$ :
  - (a) Draw a bootstrap sample  $\mathbf{Z}^*$  of size  $N$  from the training data.
  - (b) Grow a random-forest tree  $T_b$  to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size  $n_{min}$  is reached.
    - i. Select  $m$  variables at random from the  $p$  variables.
    - ii. Pick the best variable/split-point among the  $m$ .
    - iii. Split the node into two daughter nodes.
2. Output the ensemble of trees  $\{T_b\}_1^B$ .

To make a prediction at a new point  $x$ :

$$\text{Regression: } \hat{f}_{\text{rf}}^B(x) = \frac{1}{B} \sum_{b=1}^B T_b(x).$$

*Classification:* Let  $\hat{C}_b(x)$  be the class prediction of the  $b$ th random-forest tree. Then  $\hat{C}_{\text{rf}}^B(x) = \text{majority vote } \{\hat{C}_b(x)\}_1^B$ .

בשיטת Bagging אנו מקבל שכראה כל העצים יבחרו במשתנה המשמעותי ביותר לנקודת החלוקה וכך יוציאו לנו עצים דומים  
והתחזיות מהעצים השונים יטו להיות זהות ←פספסנו את המטרה

!Random Forest

ובצע תהליך זהה לתהליכי Bagging אלא שהפעם בכל אימון על תת-מדגם סט של  $m$  פרטמרים מຕור קacr שנקלבל עצים שונים.

בצורה כזו, קיבל עצים בעלי פרמטרים שונים וככל החלטה שונים.  
לאחר מכן עברו כל דגימת מבוחן נבדוק בצורה זהה על פני כלל העצים ונמצע את התוצאה

# Random Forests



**מהו הפקטור  $\alpha$  (מספר הפרמטרים) הנכון?**

**כלל האכבע אומר שኒקח:**

## דוגמא והשוואה

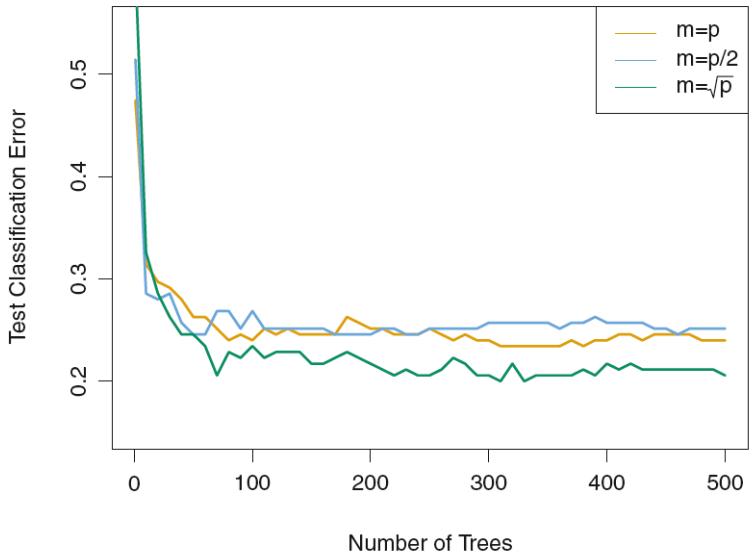
**נתוניים:** מاجر רפואי על מטופלים בו ישם 500 פיצרים שונים, וסיווג ל-15 סוגים שונים (או לא).

**בإيمان של עז החלטה בודד התקבלה שגיאה של 45.7%**

ניתן לראות בתמונה את ההשוואה בין Bagging לבין random forest עם  $n$  שוניים.

אפשר לשים לב לשיפור הקל של random forest על פני Bagging.

בנוסף, ניתן לראות שבשתי השיטות הוספה של עצים אינה גורמת ל*Overfitting* כפי שטענו מוקדם



# Boosting



זו שיטה כללית למידה המתבססת על למידה הדרגתית. ניתן להפעיל שיטה זו במגוון רחב של מודלים, אנו נתמקד בשימושה בעצি-חלהטה.

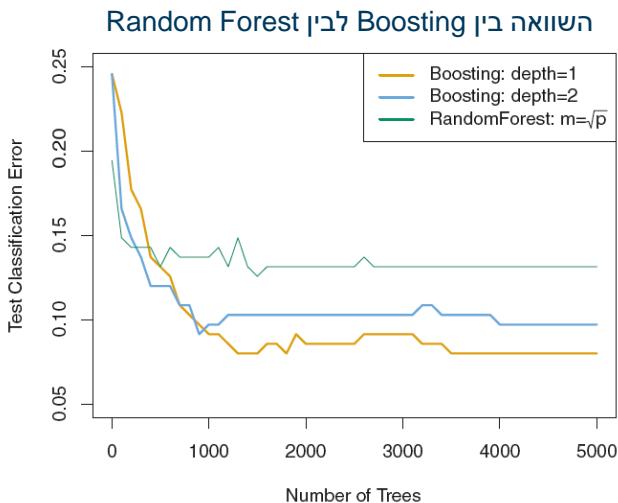
בניגוד ל-**Random Forest** ו-**Bagging** שבניו הרבה עצים במקביל,  
**Boosting** עושה דבר אחר:

אנו בונים בכל פעם עץ אחד ומשתמשים בו "שאריות" של העץ על מנת למדוד עץ חדש. בשיטה זו אנו בעצם תוקפים בכל פעם חלק אחר שבו העץ מתקשה לבצע תחזית טובה ומשמעותי יותר.

את העז החדש אנו מוסיפים לעז הקודם (עם פרט איזון מסוים) וכן אנו עושים בצורה סדרתית עד להתכנסות.

למידה זו היא **למידה איטית**, שכן בכל פעם אנו לומדים עץ אחד עם ערכיים שונים (השאריות מהעץ הקודם).

זהו שיטה מאוד עצמתית ומשתמשים בה במקרים רבים.



# Boosting



ישן מספר שאלות עליהן נרצה לענות:

## 1. כיצד נקבע את B?

בניגוד ל Random Forest ו-Bagging ניתן לרשום ש-Overfitting מתרחש כאשר דוגמאות חדשות לא ידועות נ训וטו במדויק על ידי המודל.

## 2. כיצד נקבע את ג?

זהו פרמטר השולט על קצב הלימוד. בדרך כלל נקבע את גודלו להיות 0.001 או 0.01 (תלוי בסוג הנתונים). נשים לב ש- גודלו קטנה תדרשו יותר מדגמי אימון כמפורט לעיל.

### 3. כיצד נקבע את p?

זהו פרמטר השולט על עומק העץ, לעתים קרובות עומק 1 יספיק לנו ויתן חלוקה טובה ל-2 אזורים. כאשר נצף את העצים (בצורה מדורגת, אחד על השני) נוכל לקבל additive model

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \quad (8.10)$$

(c) Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \quad (8.11)$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^B \lambda_b \hat{f}^b(x). \quad (8.12)$$



## מה ראיינו היום?

- Maximal Margin Classifier
  - Perceptron
  - SVM
- Kernels
- Multiclass SVM, LR vs SVM
- Decision-Tree
  - Bagging
  - Random Forest
- Boosting

**תודה על ההגשה**