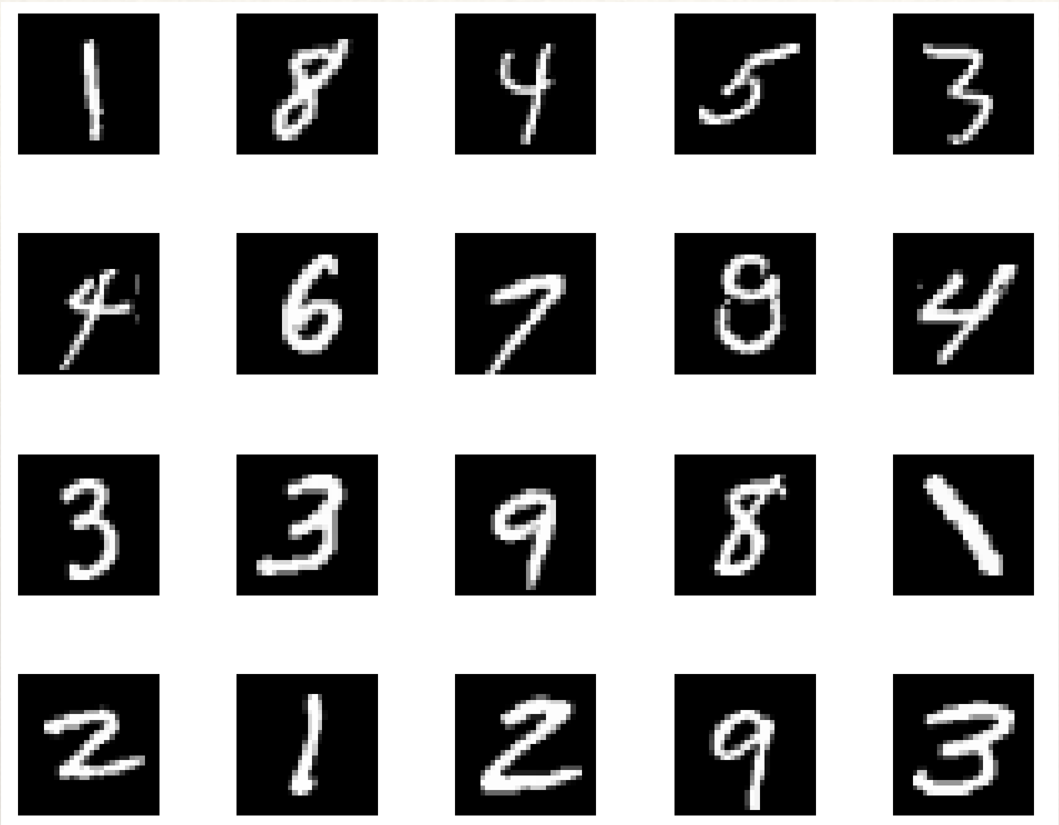
Machine learning

Artificial Neural Networks

Lecture VIII

פיתוח:
ד"ר יהונתן שלר
משה פרידמן

מוטיבציה כללית – עזרה במשלוחים



- הימים ימי קורונה – כולם התרגלו להזמנות קניות באינטרנט ולמשלוחים
- הדואר קורס – ולא רוצים שאנשים ימיינו את כל החבילות
- פיענוח כתובות זה קשה – אבל רוצים משהו יותר ממוקד
- רוצים לבנות מערכת אוטומטית לניתוח וזיהוי המיקוד של החבילה
- תוצאות הניתוח האוטומטי ינתב את החבילה לעיר או נקודת ההפצה היעודית
- בונים מערכת לפיענוח אוטומטי לכתב יד (או מכונה) של ספרות

מסווגים שראינו עד כה

- מסווג לפי "שכנים" – kNN
- עץ החלטה
- מסווג הסתברותי – ביסיאני – NB
- היום נראה משפחה חדשה של מסווגים - "מסווגים לינאריים", אבל לפני כן קצת רקע ומוטיבציה..

מקור השראה – המוח האנושי

יחידת עיבוד אלמנטרית: נוירון

100,000,000,000 נוירונים

כל נוירון מחובר לאלפי נוירונים אחרים

10,000 קישורים בממוצע מכל נוירון

הפלט של נוירון נקבע לפי הקלט של הנוירונים שמחוברים אליו

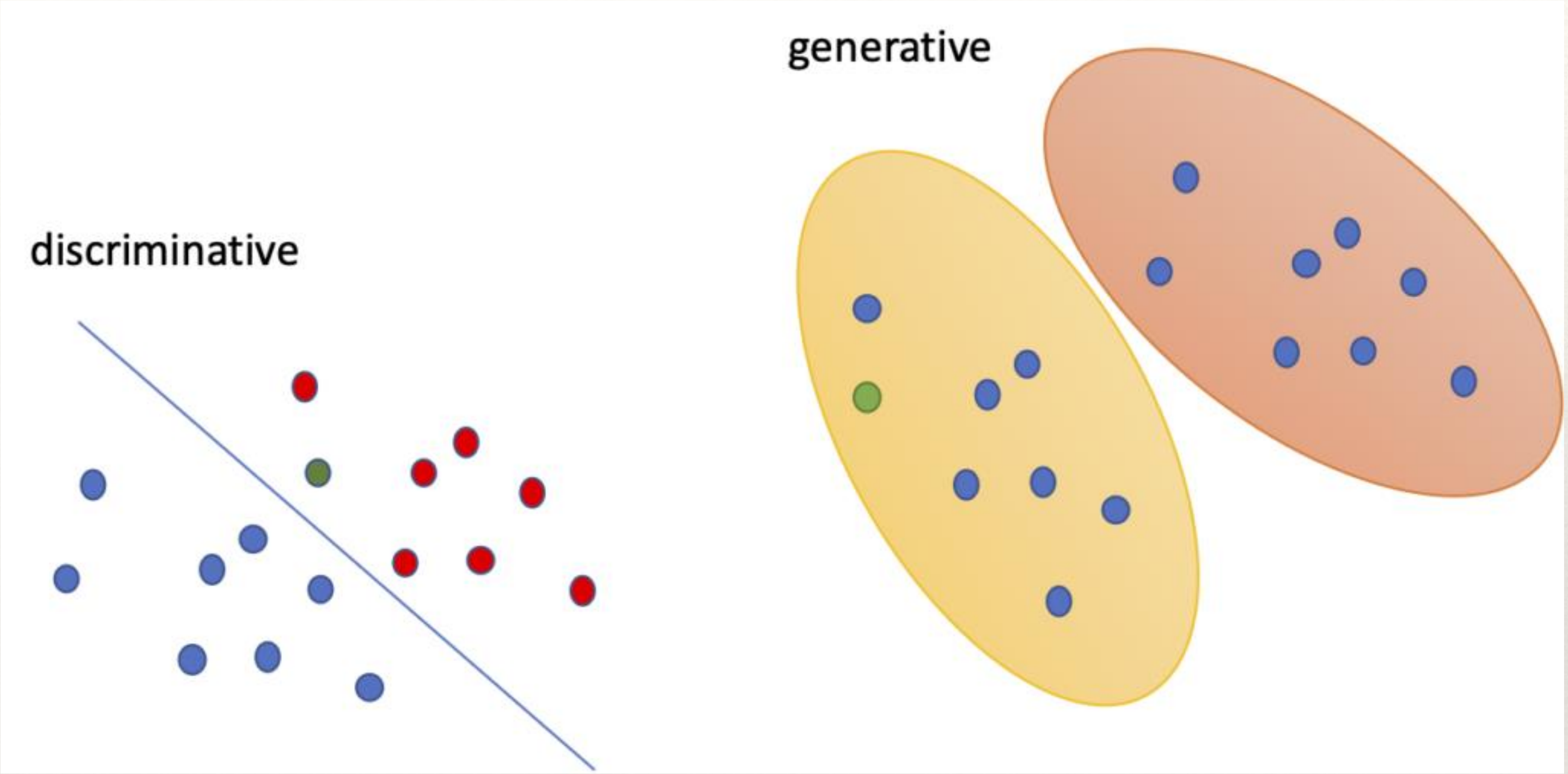
מדוע לחקות פעולת המח

- המטרה המרכזית של המחקר בבינה מלאכותית היא לבנות מכונה ששקולה ביכולותיה למוח האנושי (או עולה עליו)
- אולי כדאי לחקות את דרך הפעולה (הפיסית/כימית) של המוח
- המוח הוא מכונת חשיבה יעילה במיוחד שמגיעה להחלטות מורכבות תוך זמן קצר ע"י פעולה במקביל של מספר עצום של יחידות עיבוד פרימיטיביות (יחסית)

מודל דיסקרמינטיבי

- מפריד ומפריד לינארי

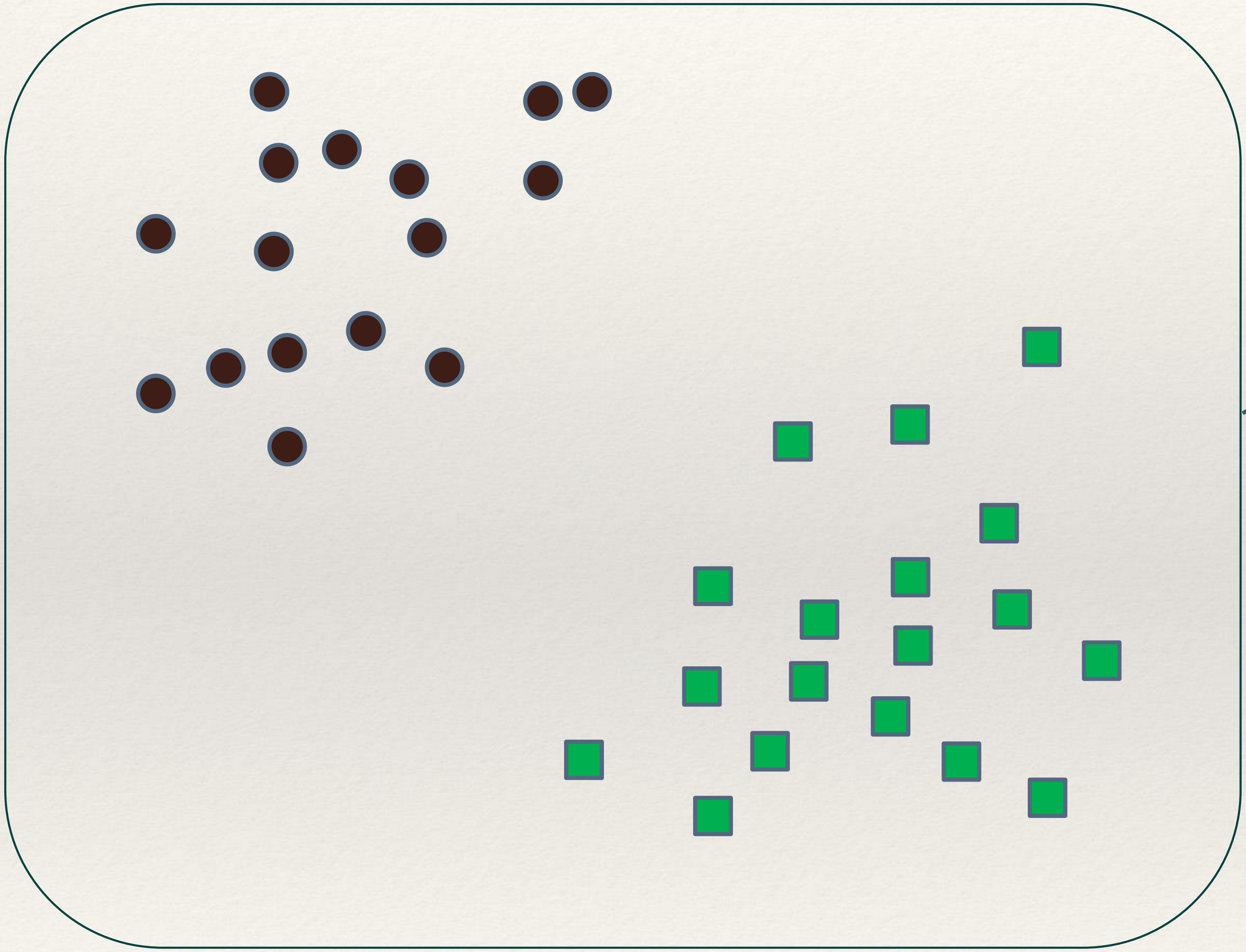
מודלים גנרטיבים ודיסקרמינטיבים - תזכורת



<u>מודל גנרטיבי</u> - מודל בו מנסים ללמוד את ההתפלגות משותפת בין המחלקה והמאפיינים. יש ביכולתם ליצר דוגמאות חדשות (generate).

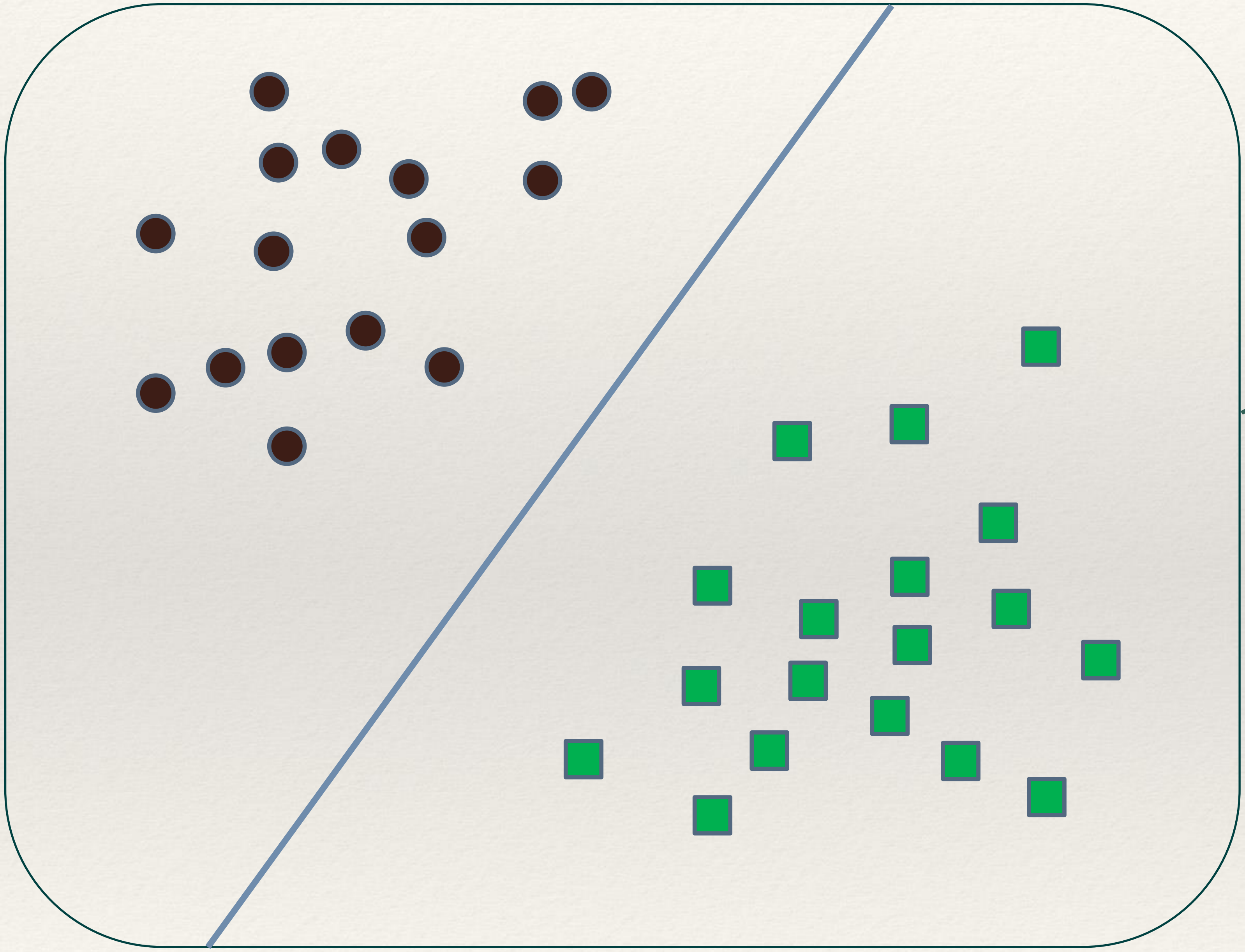
<u>מודל דיסקרמינטיבי</u> - מודל בו מחפשים הפרדה בה יש להחליט איפה מתחיל ונגמר מחלקה אחת ואיפה מתחיל ונגמר מחלקה שניה.

רקע: מודל דיסקרמינטיבי – מפריד לינארי



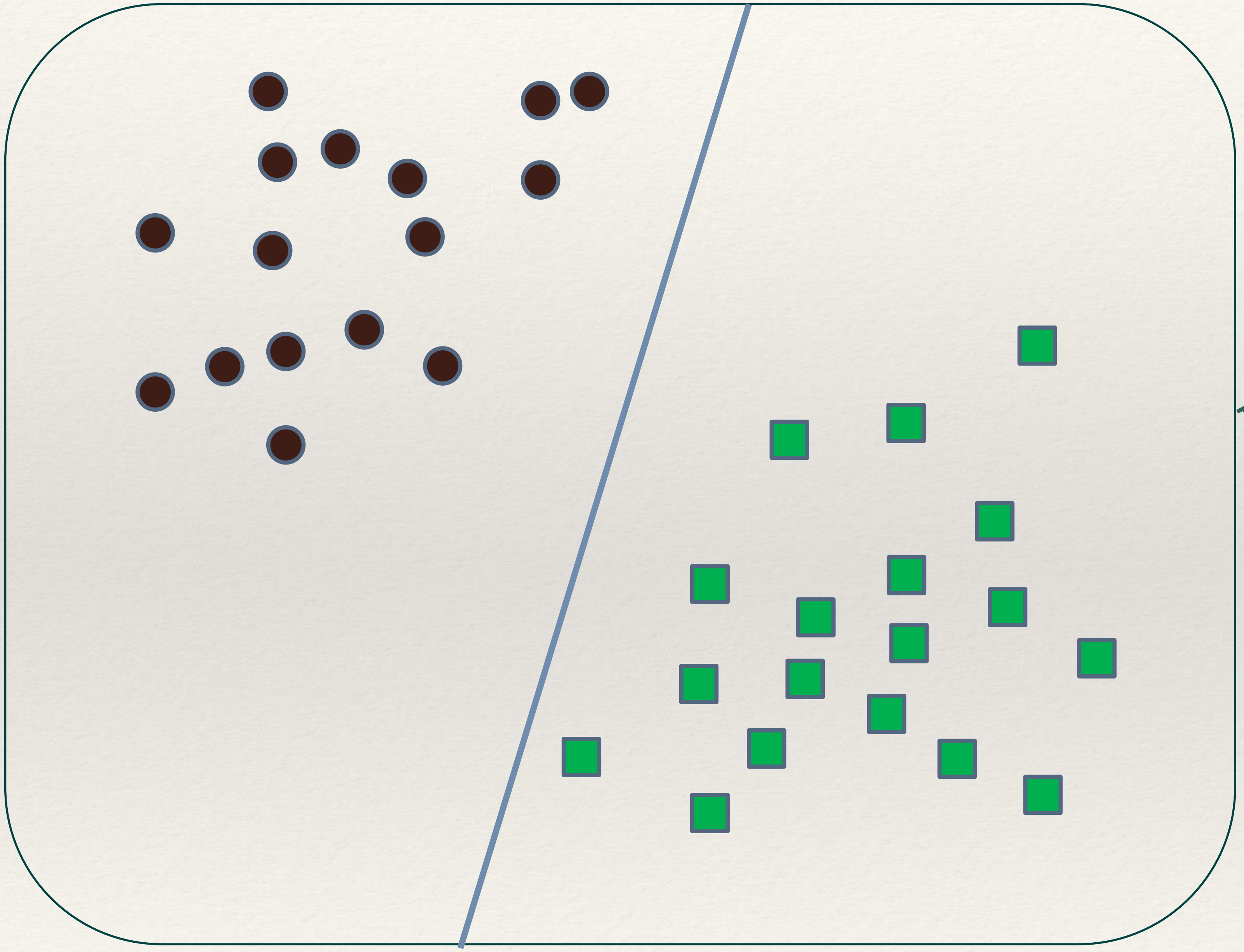
איך היינו מסווגים את
הנתונים הללו

רקע: מודל דיסקרמינטיבי – מפריד לינארי



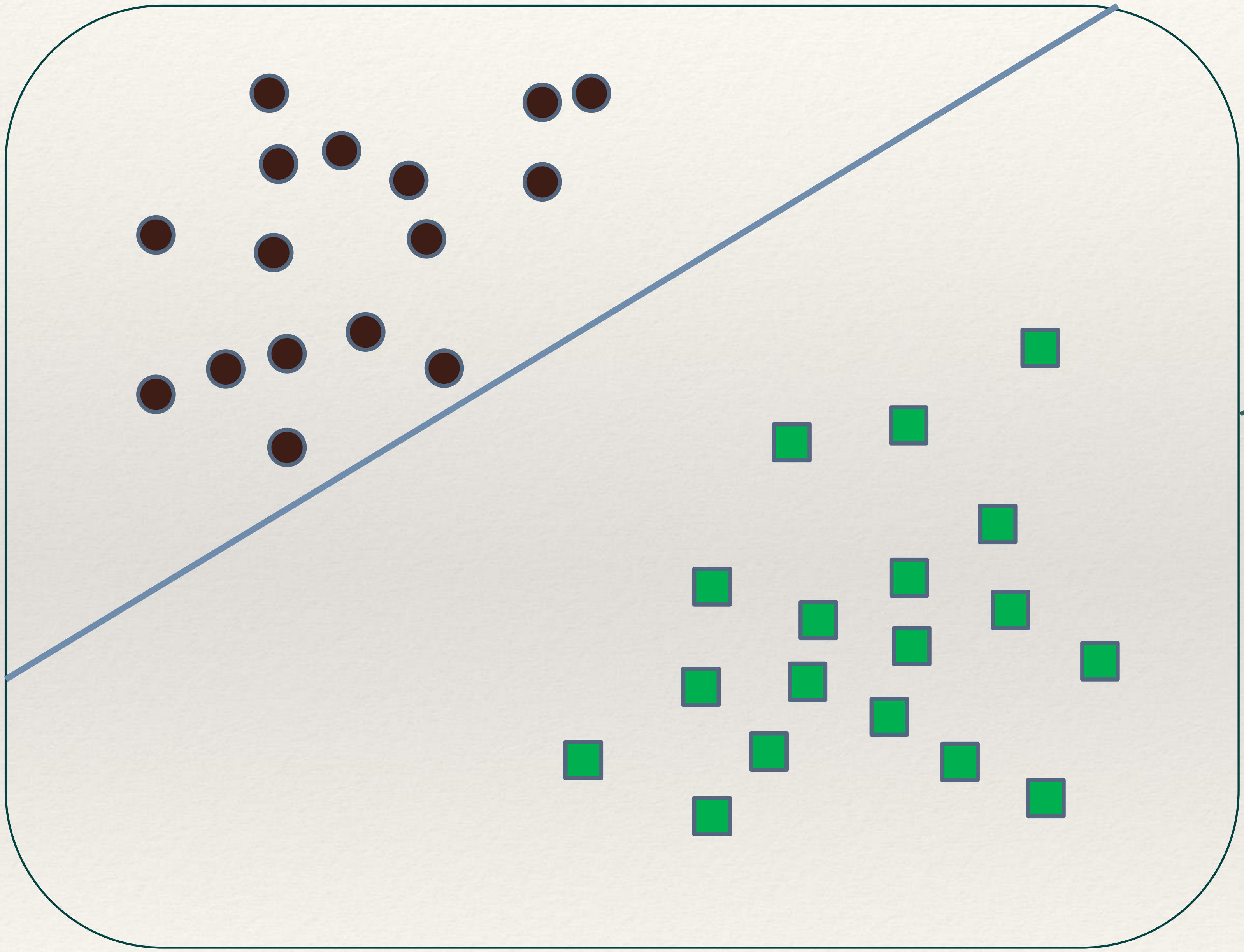
איך היינו מסווגים את
הנתונים הללו

רקע: מודל דיסקרמינטיבי – מפריד לינארי



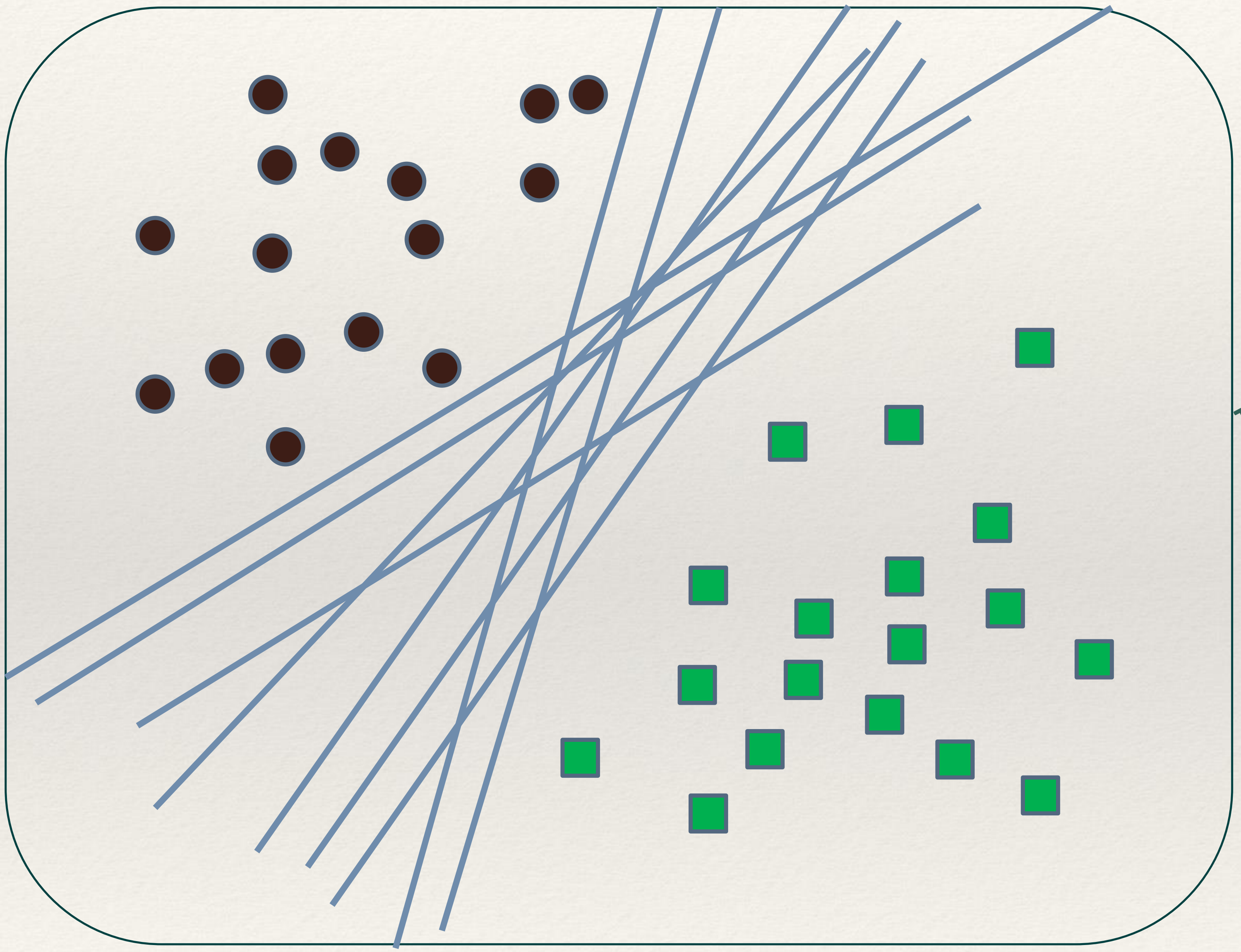
איך היינו מסווגים את
הנתונים הללו

רקע: מודל דיסקרמינטיבי – מפריד לינארי



איך היינו מסווגים את
הנתונים הללו

רקע: מודל דיסקרמינטיבי – מפריד לינארי

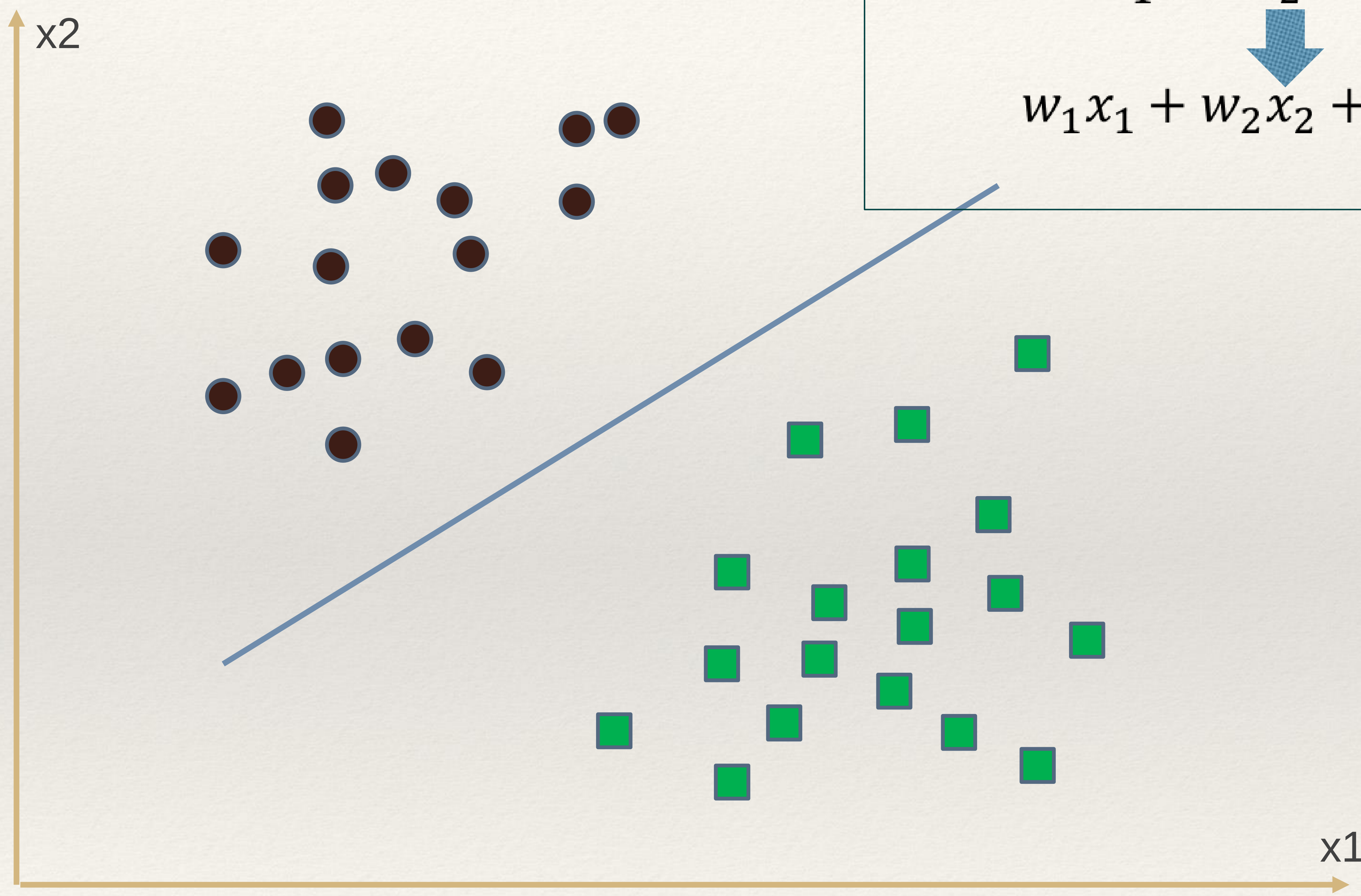


איך היינו מסווגים את הנתונים הללו

מסווג ליניארי – ממשואת קו ישר לקומבינציה ליניארית של ערכים ומשקולות

$$ax_1 + bx_2 + c = 0$$

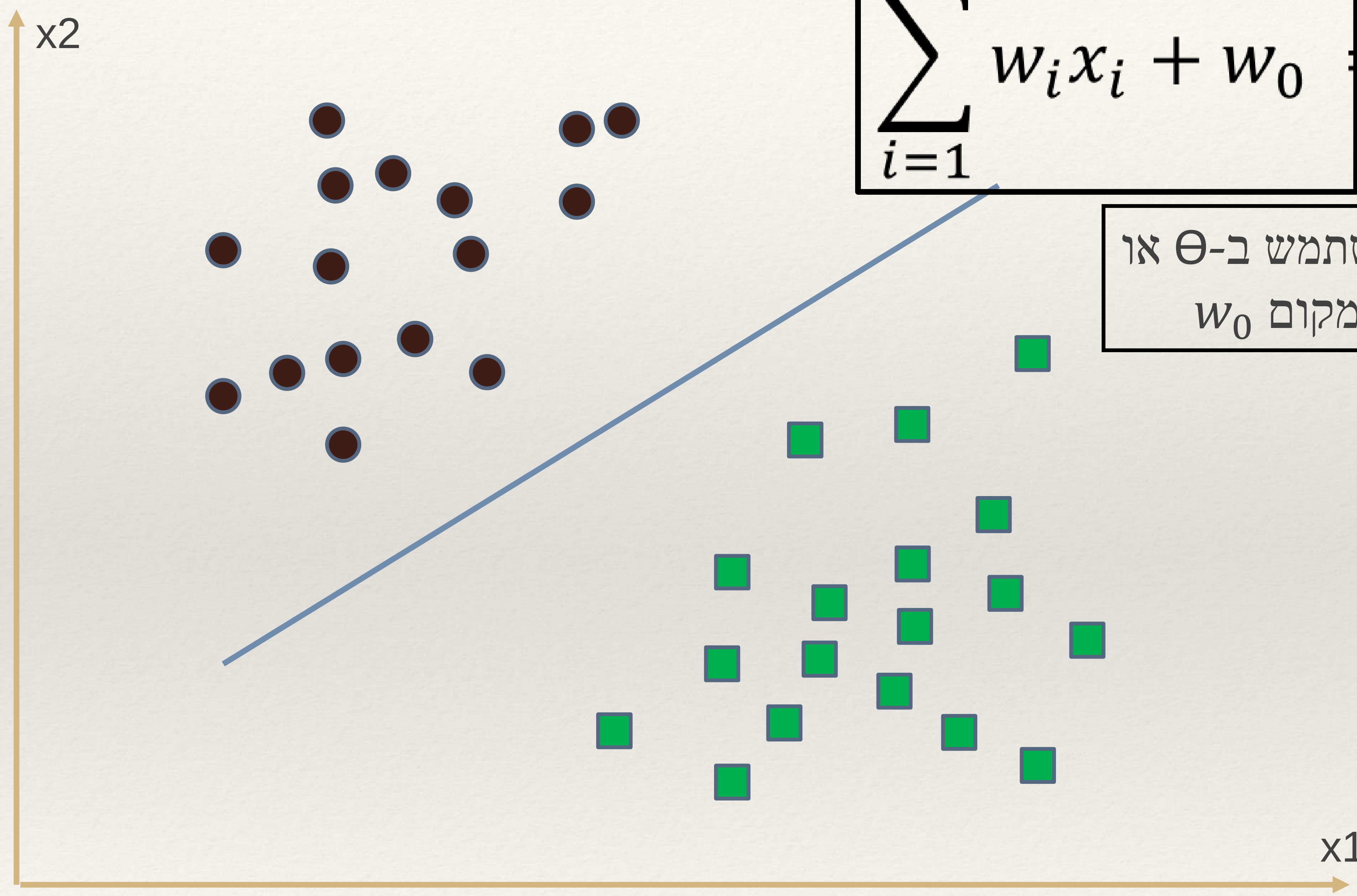
$$w_1x_1 + w_2x_2 + w_0 = 0$$



מסווג לינארי – משוואה רב מימדית

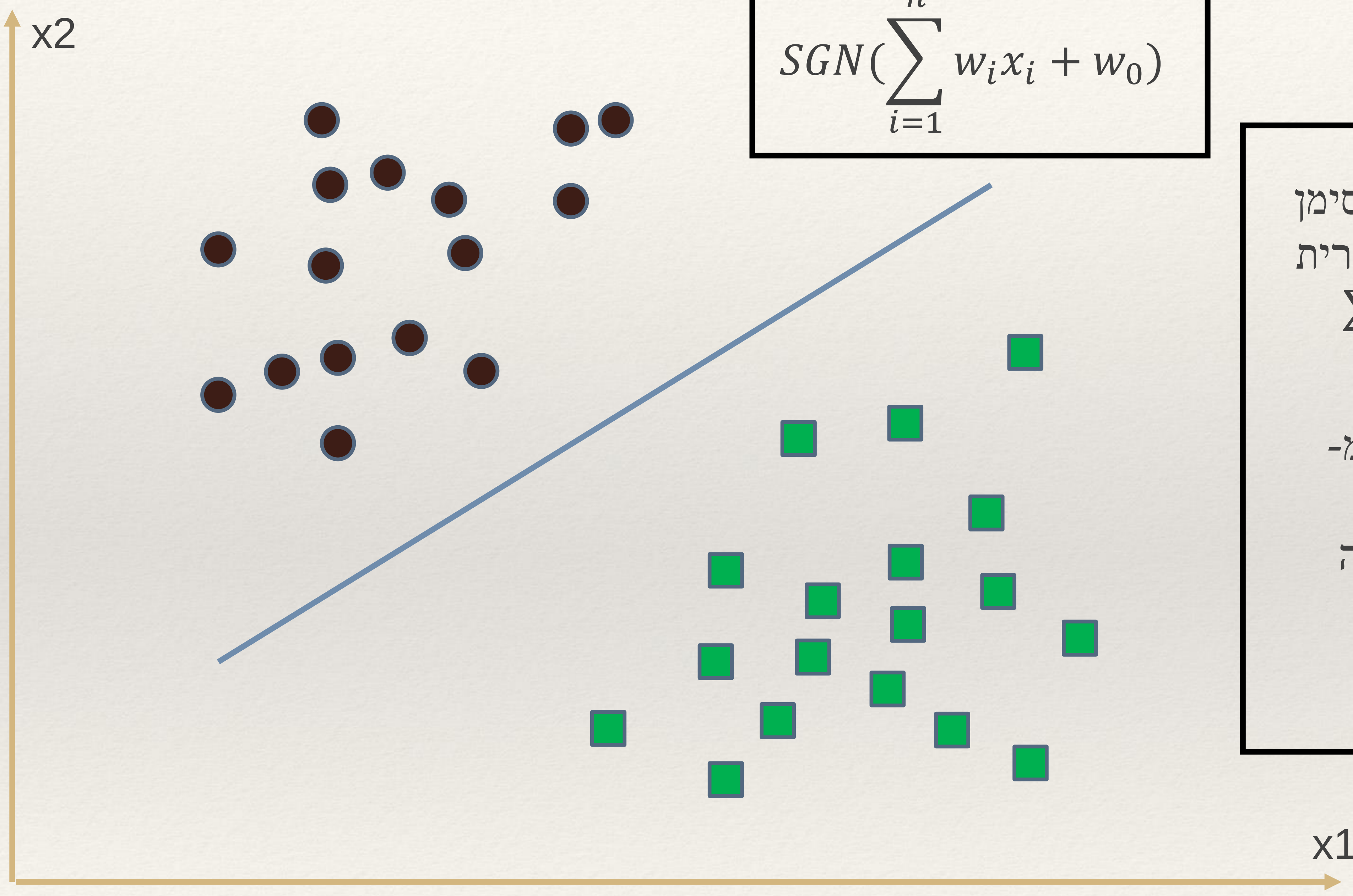
$$\sum_{i=1}^{n} w_i x_i + w_0$$

לעיתים נשתמש ב-Θ או
ב-b במקום w_0



מסווג לינארי – משוואת הסיווג

$$SGN(\sum_{i=1}^{n} w_i x_i + w_0)$$



מסווגים את x, לפי הסימן של הקומבינציה הלנארית $\sum_{i=1}^{n} w_i x_i + w_0$

אם התוצאה גדולה מ-0, נסווג קטגוריה חיובית, אם התוצאה קטנה מ-0, נסווג קטגוריה שלילית

מסווג לינארי – שאלות ביניים

שאלות:

1. איזו משוואה דיסקרמינטיבית (מפרידה) מההמשוואות הבאות הינה משוואה לינארית?

א. $3x_1+x_2+5=0$ ב. $x_1^2+2x_2+1=0$

תשובות אפשריות:

1. א – לינארית, ב – לינארית
2. א – לא לינארית ב- לינארית
3. א – לינארית ב- לא לינארית
4. א – לא לינארית ב- לא לינארית

תשובה נכונה:

ג. א. לינארית. ב. לא לינארית (פולינומיאלית)

מסווג לינארי – שאלות ביניים

שאלות:

2. עבור ישר $3x_1+x_2+5$, מה יהיה הסיווג של זוגות הערכים הבאים (האפשרויות: חיובית/ שלילית)?

א. $x_1=-2, x_2 = 0$ ב. $x_1=1, x_2 = -2$

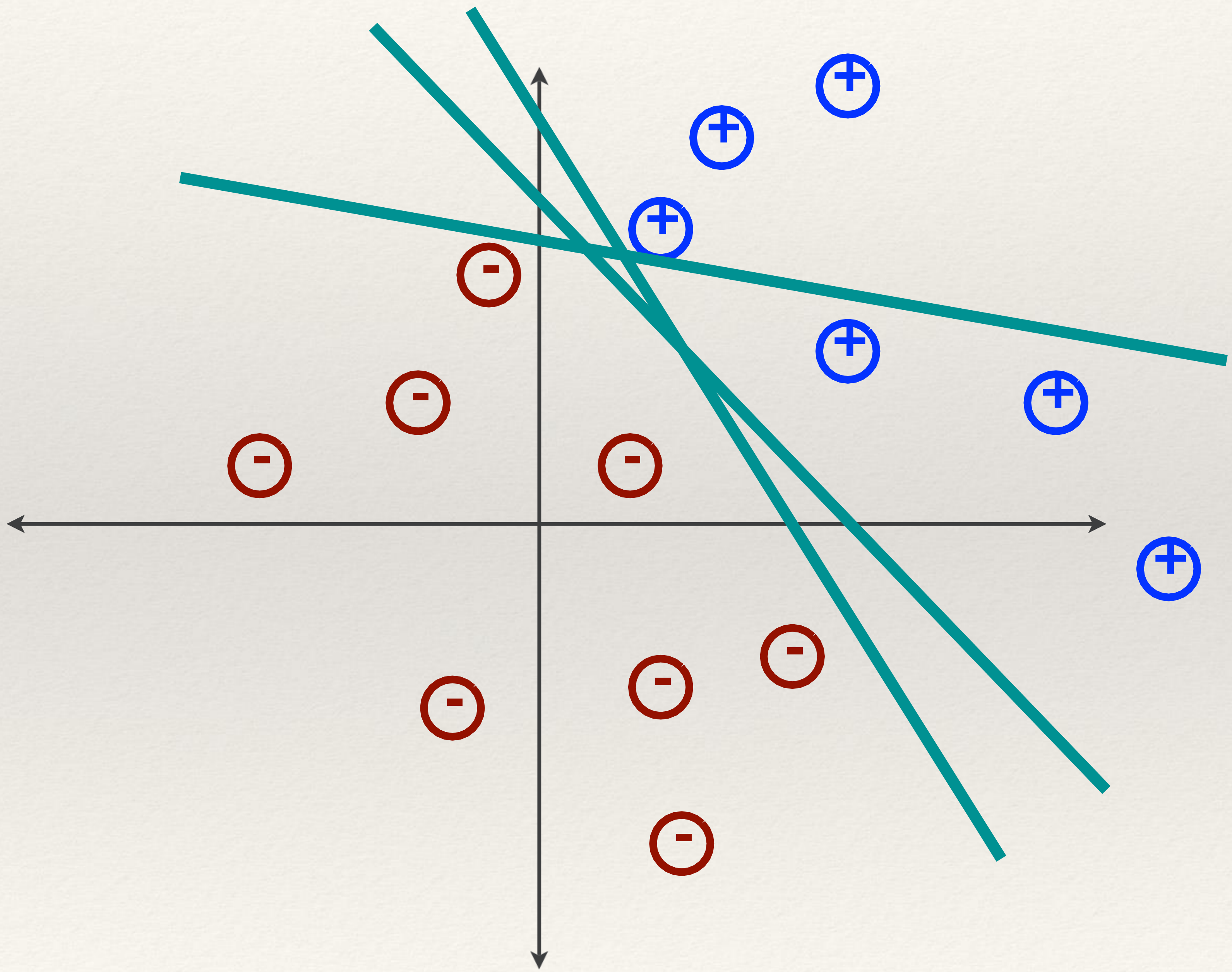
תשובות אפשריות:

1. א – חיובי ב – חיובי
2. א – חיובי ב – שלילי
3. א – שלילי ב – חיובי
4. א – שלילי ב - שלילי

תשובה נכונה:

3. א. שלילי ב. חיובי

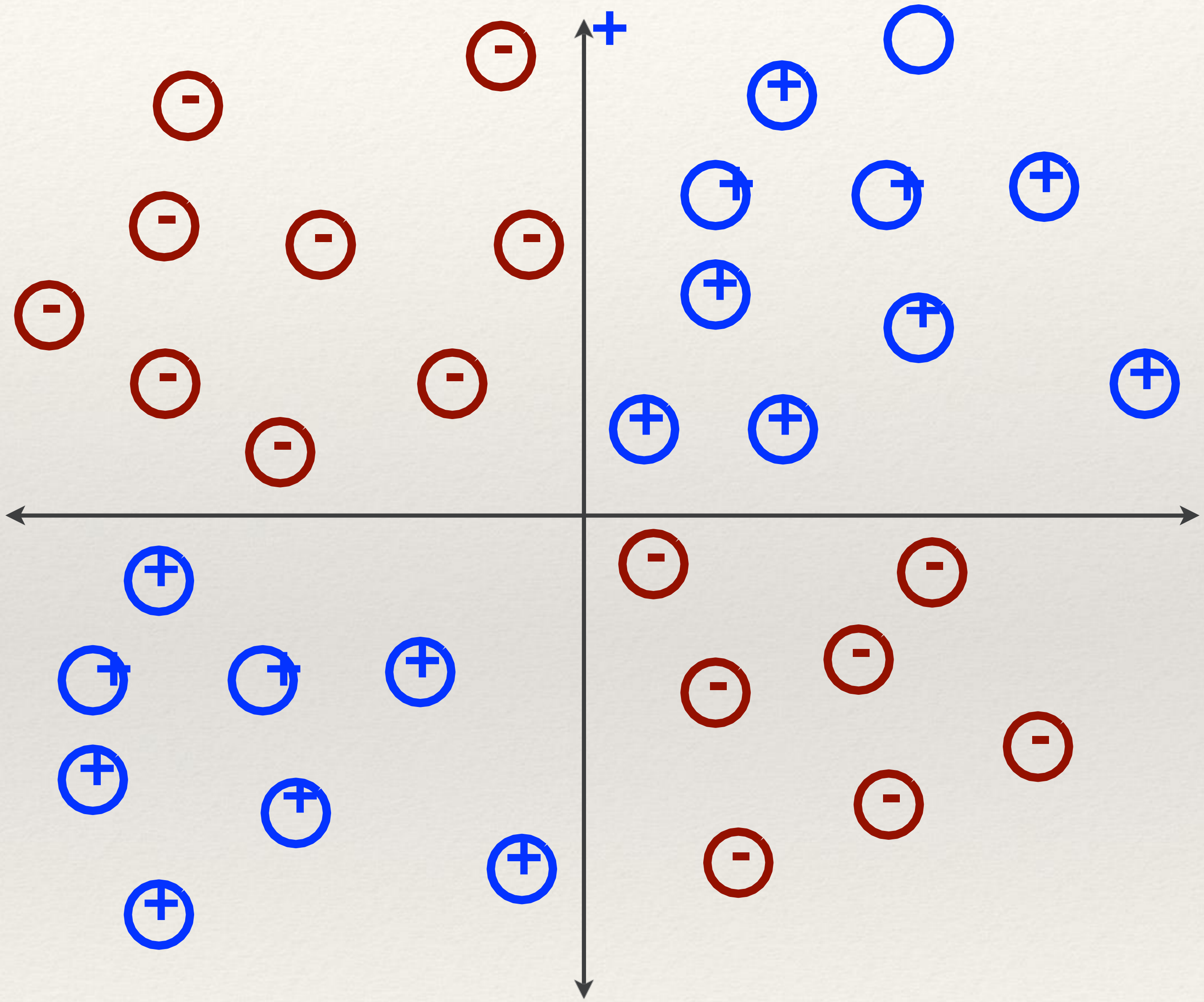
האם כל בעית למידה ניתנת למידול באמצעות מפריד לינארי?



קבוצת דוגמאות שלא ניתנת להפרדה לינארית

<u>שאלה</u>: האם ישנו מודל דיסקרמינטיבי (מפריד), שיכול לטפל במקרים ללא הפרדה לינארית?

<u>תשובה</u>: ישנם 2 אפשרויות:
1. משוואה לא לינארית.
2. תרכובת של משוואות לינאריות (נראה אפשרות זו בהמשך)



מנוירונים אנושיים לנוירונים מלאכותיים

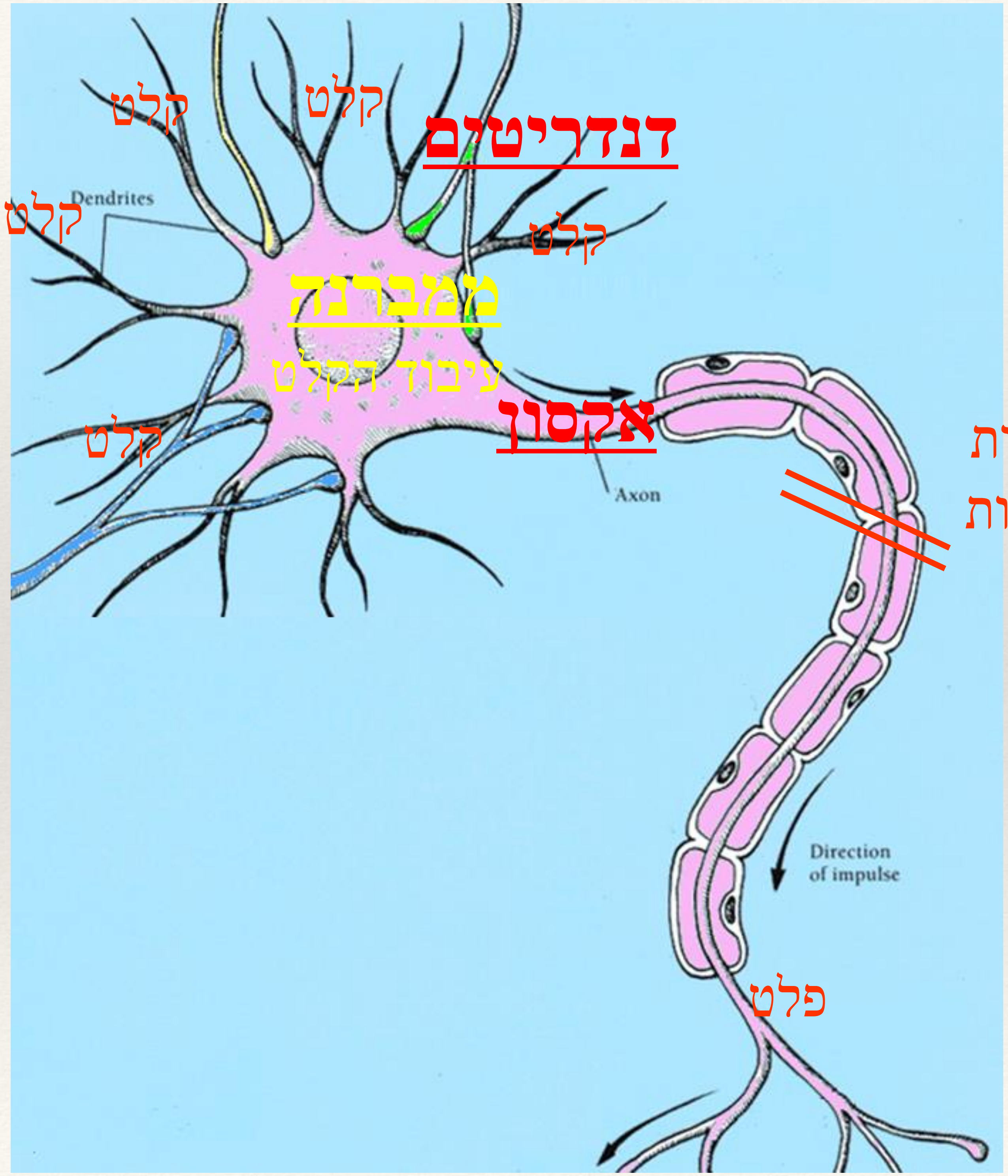
נוירון אנושי – מודל פשטני

- פוטנציאל פעולה

הנוירון המלאכותי:

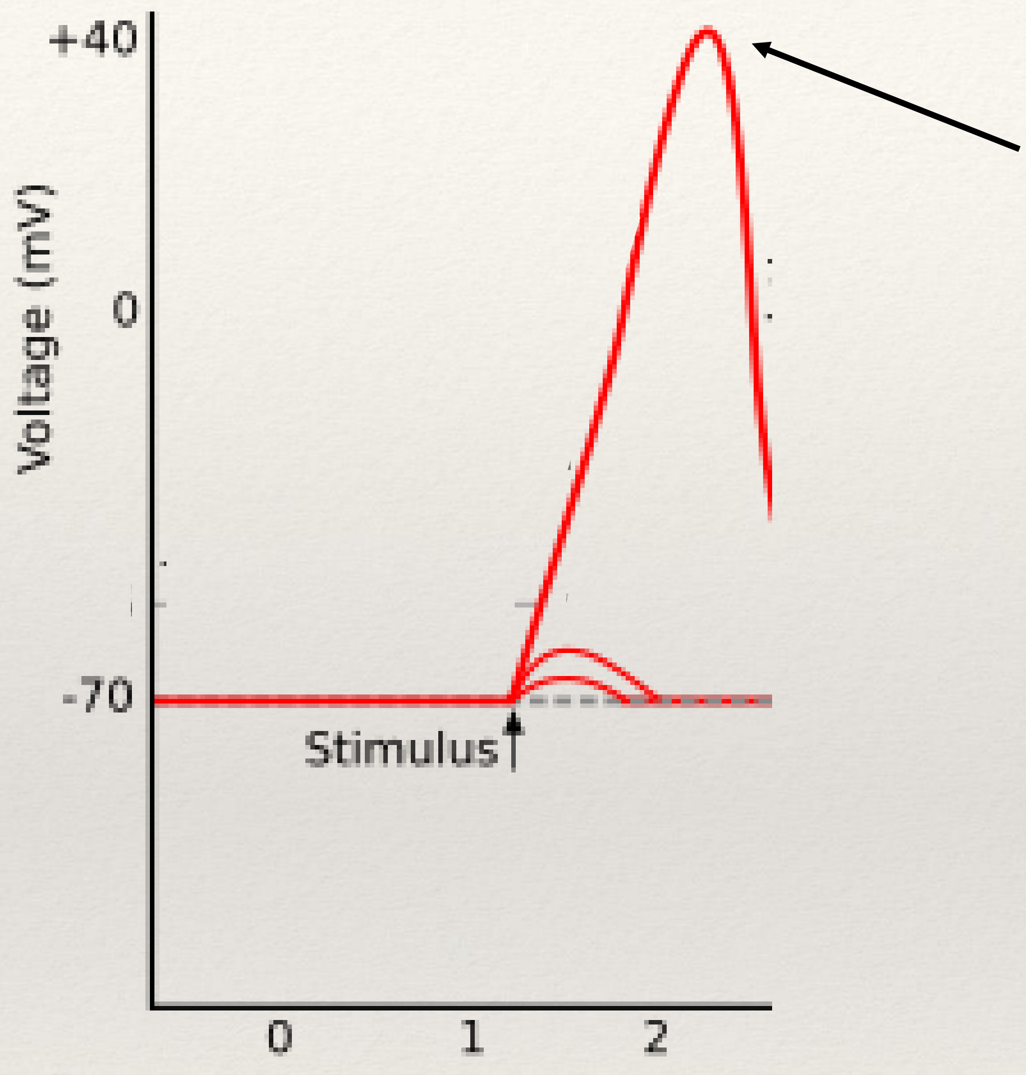
- perceptron
- פונקציות הפעלה

נוירון אנושי – מודל פשטני



תעלות
תלויות
מתח

נוירון אנושי – מודל פשטני – פוטנציאל פעולה



זמן (מילי-שניות)

פוטנציאל פעולה (אקטיבציה)

- במידה ונוצר מספיק מתח מהקלט, נוצר פוטנציאל פעולה.
- הפלט הוא יחיד (אקסון אחד)
- הפלט מהווה קלט לנוירונים אחרים

*Perceptron – הנוירון המלאכותי – מודל א'

❖ היחידה הבסיסית ברשת נוירונים

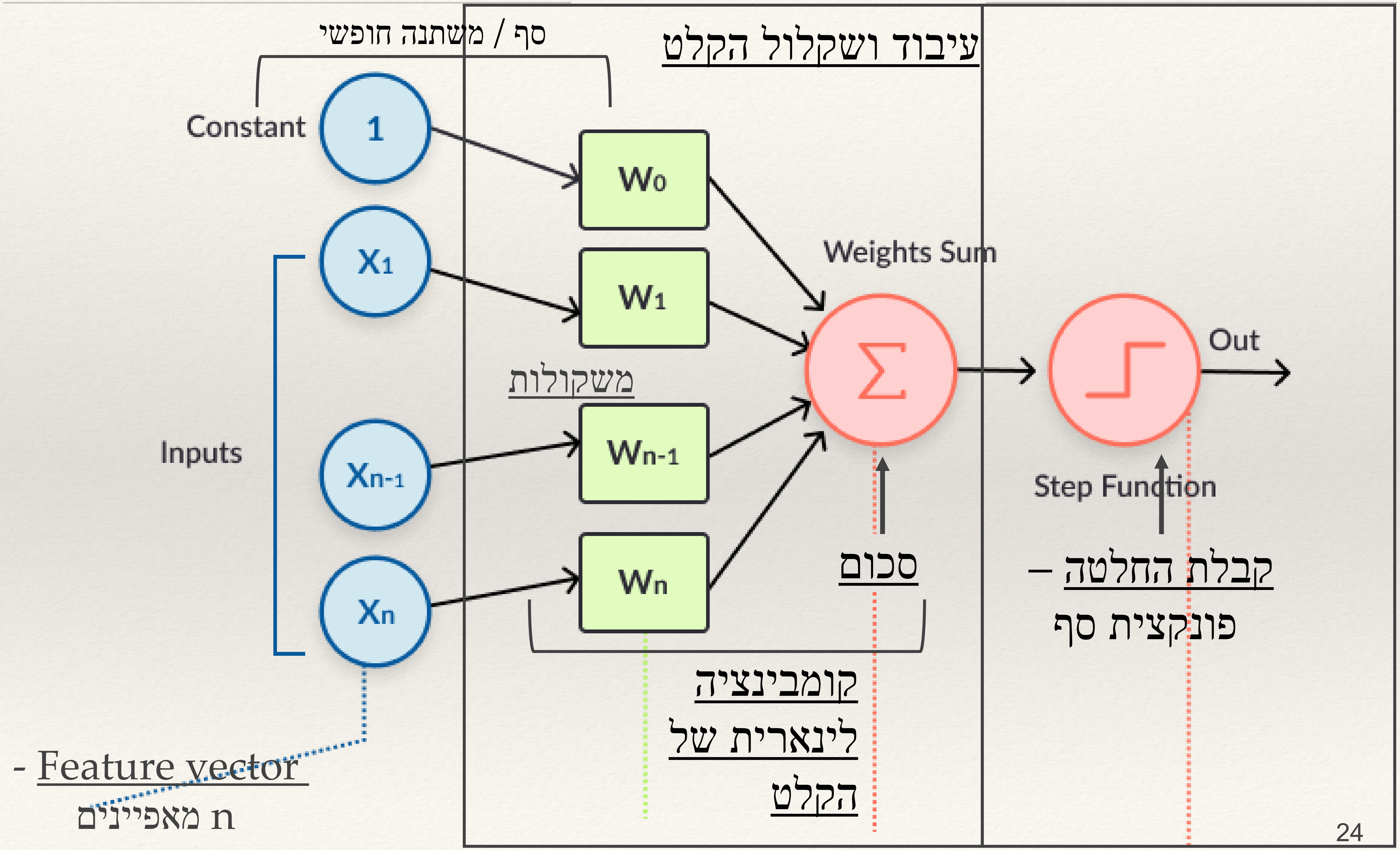
❖ יכולה לשמש כמערכת לומדת גם באופן עצמאי

❖ לפרספטרון יש n קלטים ופלט אחד

❖ הפרספטרון מחשב קומבינציה לינארית של הקלטים, ופולט 1 אם היא גדולה מסף נתון, 0 (או לעיתים 1-) אחרת

הערה:
* בעברית נקרא ל-perceptron פשוט פרספטרון או פרצפטרון או נוירון מלאכותי

Perceptron – הנוירון המלאכותי – מודל א'



Perceptron - הסבר

נתונים n קלטים: $x_1, \ldots, x_n$

(בבעית סיווג זהו ה-feature vector שלנו ואלו ערכי המאפיינים)

$$o(\langle x_1, \ldots, x_n \rangle) = \begin{cases} 1 & if\ w_0 + w_1 \cdot x_1 + \cdots + w_n \cdot x_n > 0 \\ -1 & otherwise \end{cases}$$

מכיוון שניתן לכתוב את התנאי כך:

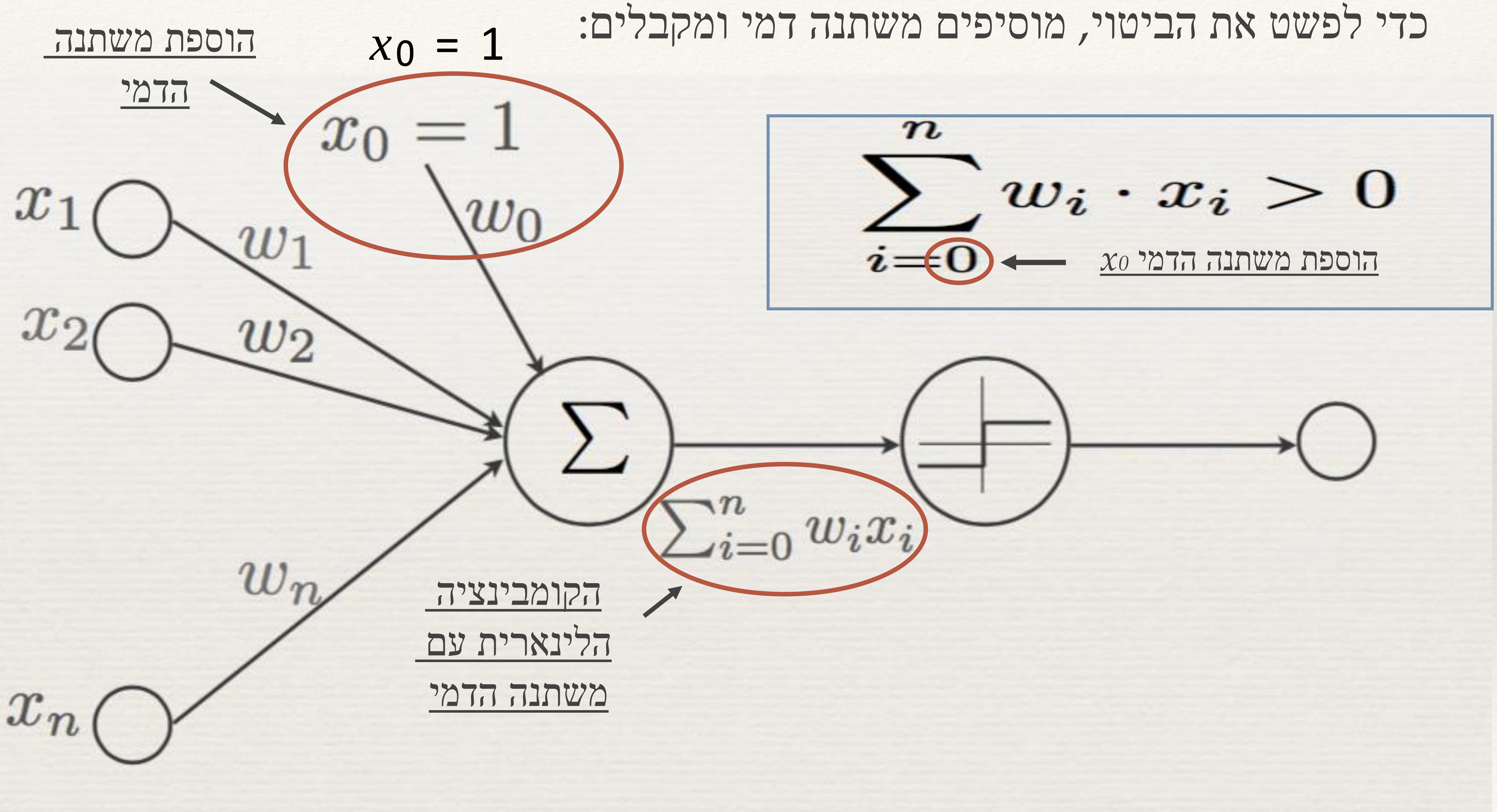
$$w_1 \cdot x_1 + \cdots + w_n \cdot x_n > -w_0$$

$-w_0$ מהווה בעצם סף שקובע את תוצאת הסיווג

Perceptron – הסבר – הוספת משתנה דמי

כדי לפשט את הביטוי, מוסיפים משתנה דמי ומקבלים:

$x_0 = 1$



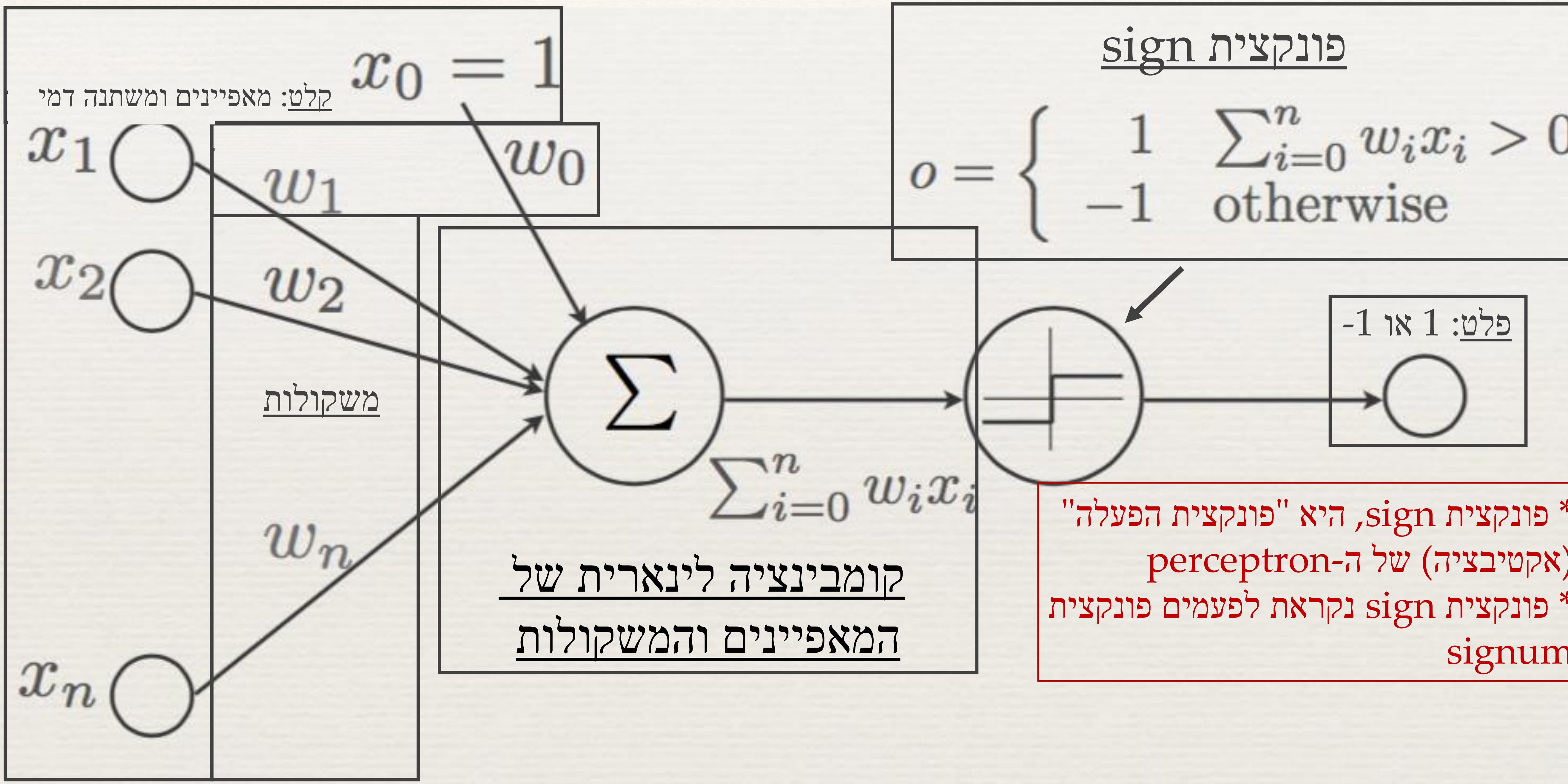
Perceptron – הסבר – הוספת משתנה דמי – צורה מקובלת לכתיב מתמטי

- בהינתן ווקטור לסיווג (x) ובהינתן סט משקולות (w)

$$\begin{pmatrix} w_0 & w_1 & w_2 & \cdots & \cdots & w_d \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_d \end{pmatrix} = \sum_{i=0}^{d} w_i x_i = W^T x$$

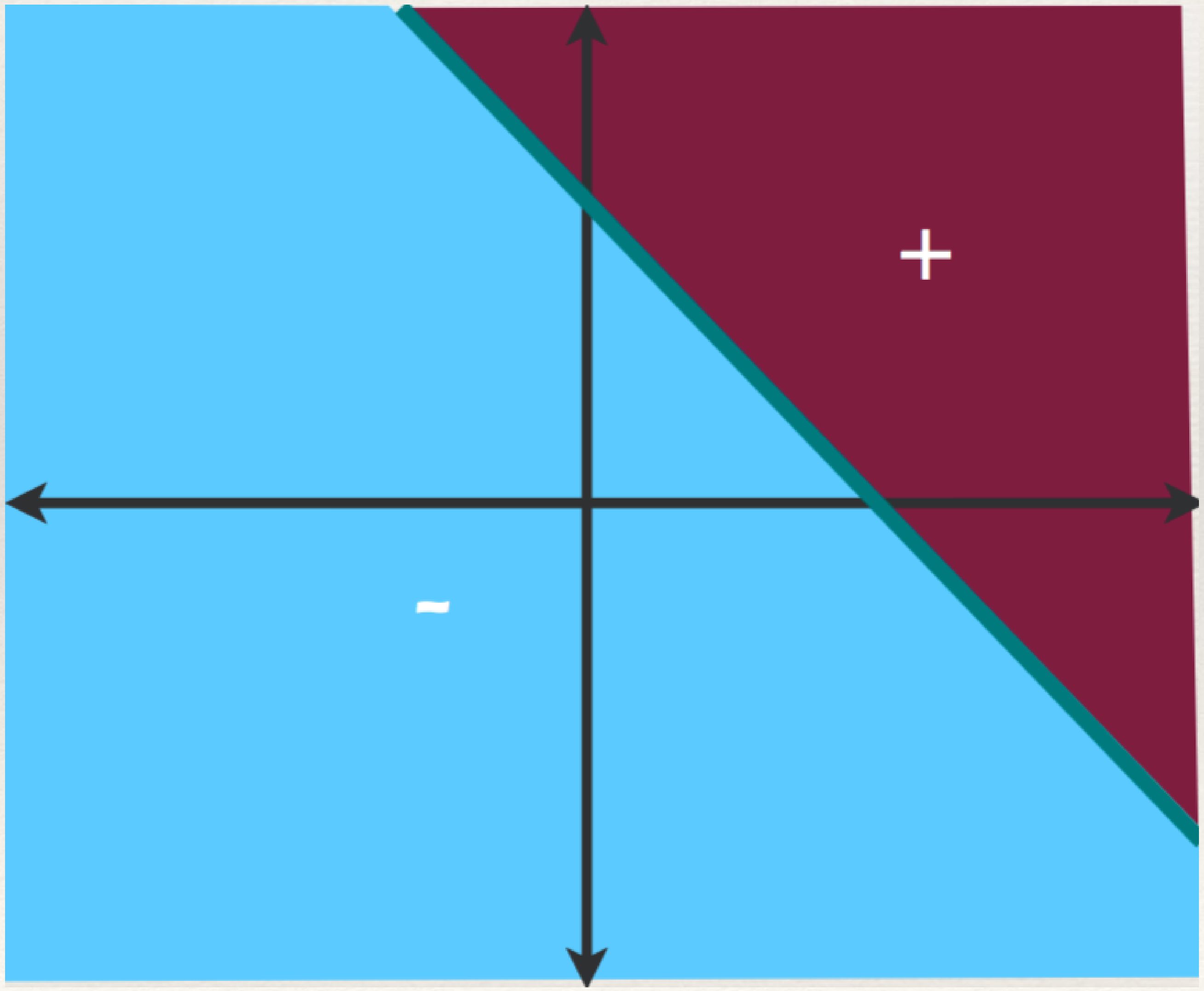
- נסווג את הווקטור עפ"י תוצאת המכפלה הפנימית בין X ו-W והשוואה לסף.

* sign פונקצית – הסבר – Perceptron



Perceptron – הסבר – פלט

- הפרצפטרון הוא למעשה מפריד ליניארי – מישור שמפריד את המרחב לשני חלקים – חלק עבור כל סיווג



דוגמה לקבלת החלטה - Perceptron

נתונה משוואת המישור (Hyperplane equation): $-1x_1$+$1x_2$-2=0

אליו מתאימה פונקצית Sign: SGN($-1x_1$+$1x_2$-2)

א. מהי יהיה הפלט של sign, עבור הוקטור x_1=3,x_2 = 29 ?

ב. מה המשמעות מבחינת סיווג הקטגוריה (חיובית או שלילית)?

ג. מהם ערכי המשקולות של וקטור המשקולות $\vec{w}$?

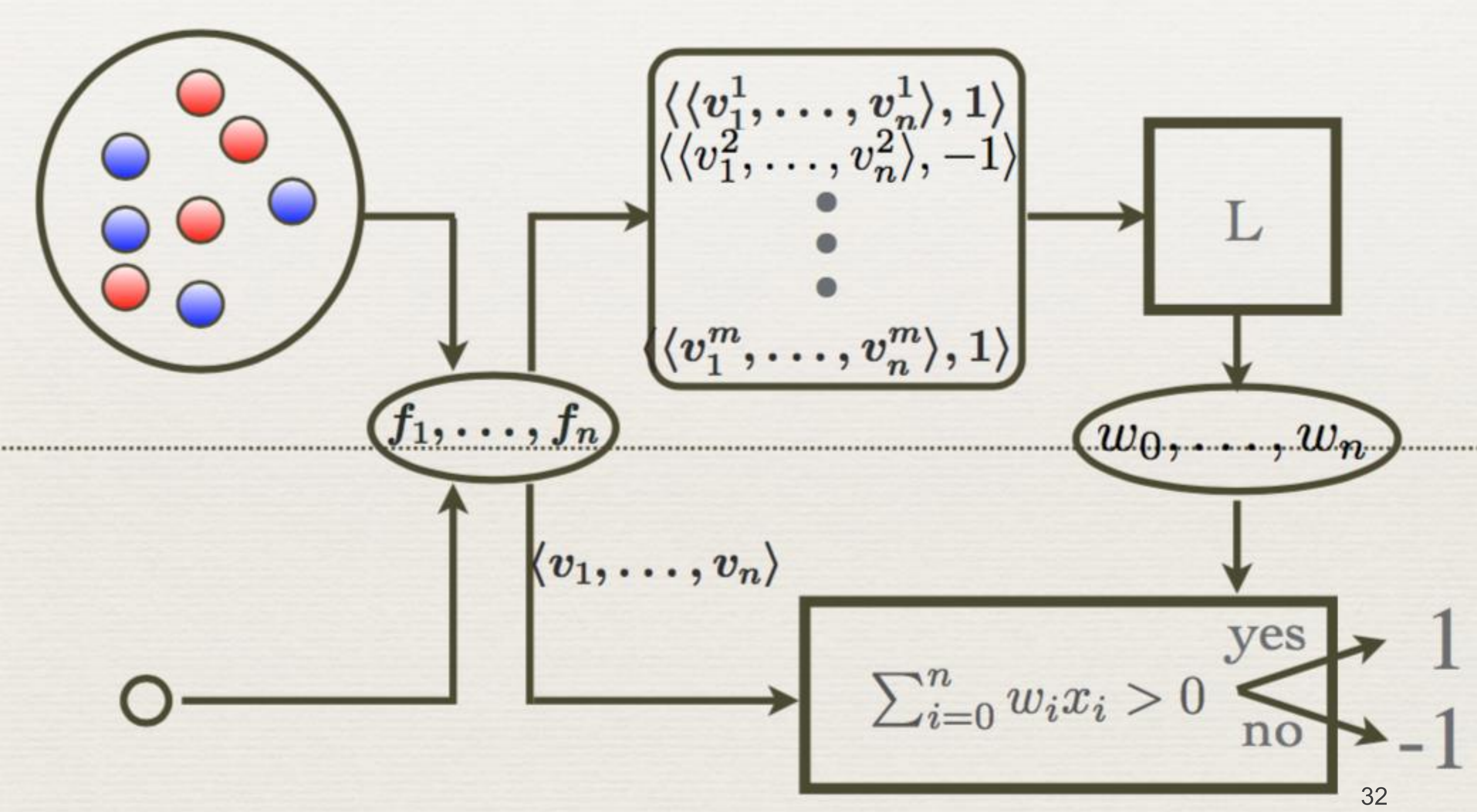
<u>תשובה</u> נציב ב-sign ונקבל: (-1x3 + 1x29 - 2)>0

לכן sign=1, ונסווג קטגוריה חיובית (positive)

$\vec{w}$=(-2,-1,1)

פרצפטרון - שלב האימון

אימון פרצפטרון - תהליך הלמידה



אימון פרצפטרון – על אלגוריתם הלמידה

- מבצע חיפוש חמדן (Greedy) במרחב ההיפותזות
- עוצר כשמוצא היפותזה עקבית
- לא עוצר כאשר לא קיימת הפרדה (עדיין?)

אימון פרצפטרון –אלגוריתם הלמידה

- מעבד דוגמא אחרי דוגמא
- דוגמא שסימונה שונה מזה של ההיפותזה הנוכחית, גורמת לעדכון המקדמים

אם הסכום קטן מידי – מגדילים את המקדמים של התכונות עם ערך **חיובי**, מקטינים כאלה עם ערך **שלילי**

אם הסכום גדול מידי – מגדילים את המקדמים של התכונות עם ערך **שלילי**, מקטינים כאלה עם ערך **חיובי**

אימון פרצפטרון – עדכון המקדמים

- כלל העדכון של המשקולות:

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(t - o)x_i$$

Δw_i - עדכון ל-w_i (הוספה או הורדה ביחס לאיטרציה הקודמת)

η קבוע (קטן מ-1) הקובע את קצב הלמידה (למשל 0.1)

t סימון הדוגמא הנוכחית - ערך הקטגוריה האמיתי של הדוגמה

o הערך שנותן ה perceptron עבור הדוגמא הנוכחית

אימון פרצפטרון – עדכון המקדמים - הסבר

$$\boxed{\Delta w_i = \eta(t - o)x_i}$$

η קבוע הלמידה

t ערך הקטגוריה האמיתי של הדוגמה

o הערך שנותן ה perceptron

$$t = o \Rightarrow \Delta w_i = 0$$

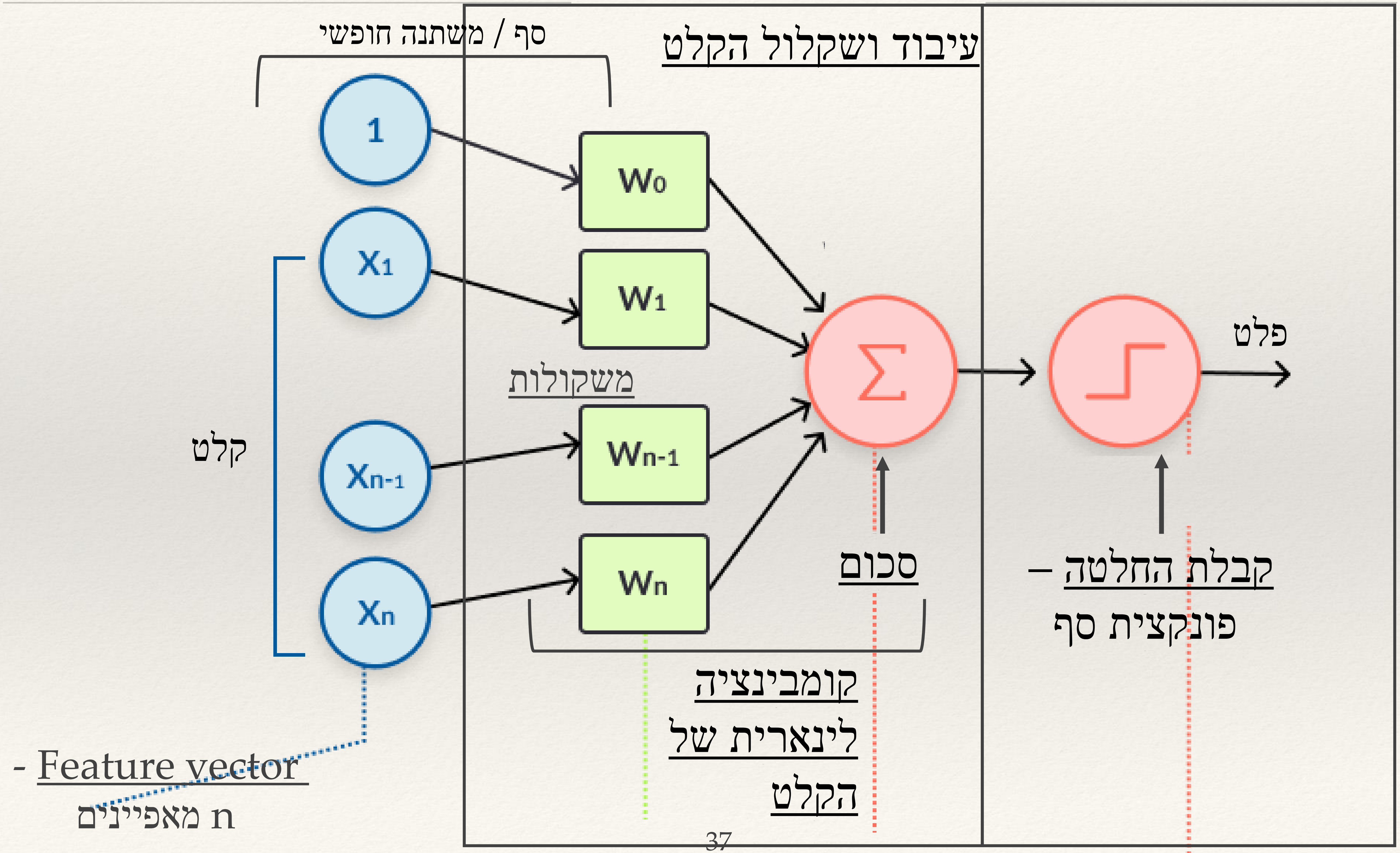
$$t = 1 \wedge o = -1 \wedge x_i > 0 \Rightarrow \Delta w_i > 0$$

$$t = 1 \wedge o = -1 \wedge x_i < 0 \Rightarrow \Delta w_i < 0$$

$$t = -1 \wedge o = 1 \wedge x_i < 0 \Rightarrow \Delta w_i > 0$$

$$t = -1 \wedge o = 1 \wedge x_i > 0 \Rightarrow \Delta w_i < 0$$

Perceptron – מודל א'
הנוירון המלאכותי עם פונקצית הפעלה sign



אימון פרצפטרון –אלגוריתם הלמידה

- מעבד דוגמא אחרי דוגמא
- דוגמא שסימונה שונה מזה של ההיפותזה הנוכחית, גורמת לעדכון המקדמים

אם הסכום קטן מידי – מגדילים את המקדמים של התכונות עם ערך **חיובי**, מקטינים כאלה עם ערך **שלילי**

אם הסכום גדול מידי – מגדילים את המקדמים של התכונות עם ערך **שלילי**, מקטינים כאלה עם ערך **חיובי**

אימון פרצפטרון – עדכון המקדמים

- כלל העדכון של המשקולות:

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(t - o)x_i$$

Δw_i - עדכון ל-w_i (הוספה או הורדה ביחס לאיטרציה הקודמת)

η קבוע (קטן מ-1) הקובע את קצב הלמידה (למשל 0.1)

t סימון הדוגמא הנוכחית - ערך הקטגוריה האמיתי של הדוגמה

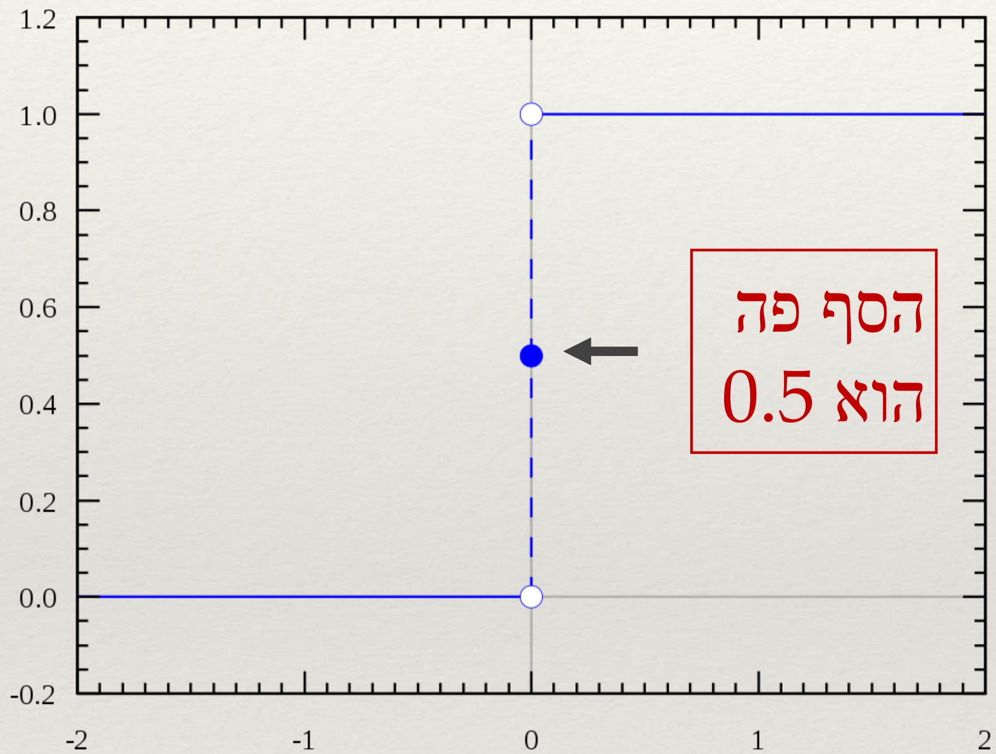
o הערך שנותן ה perceptron עבור הדוגמא הנוכחית

הנוירון המלאכותי – מודל ב' – החלפת היחידה הבסיסית - sigmoid

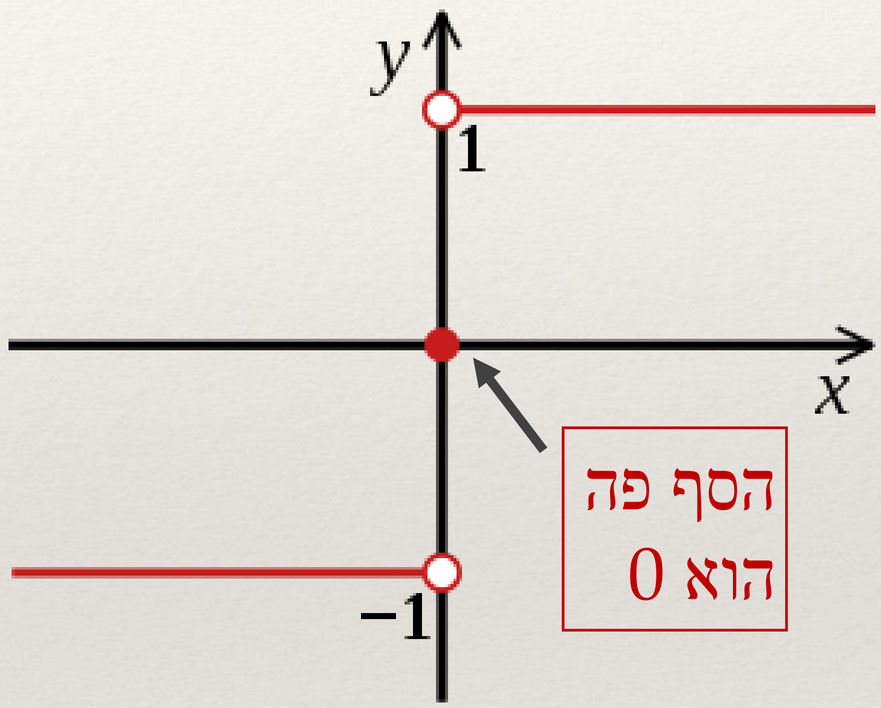
- החלפת פונקצית sign
- פונקצית הפסד ופונקצית מטרה
- טענת התכנסות
- Gradient Decent

Perceptron – מודל א' – פונקצית ההפעלה

<u>פונקצית הסף</u> – פונקצית ההפעלה היא פונקצית מדרגה ובעצם ראינו 2 כאלה:

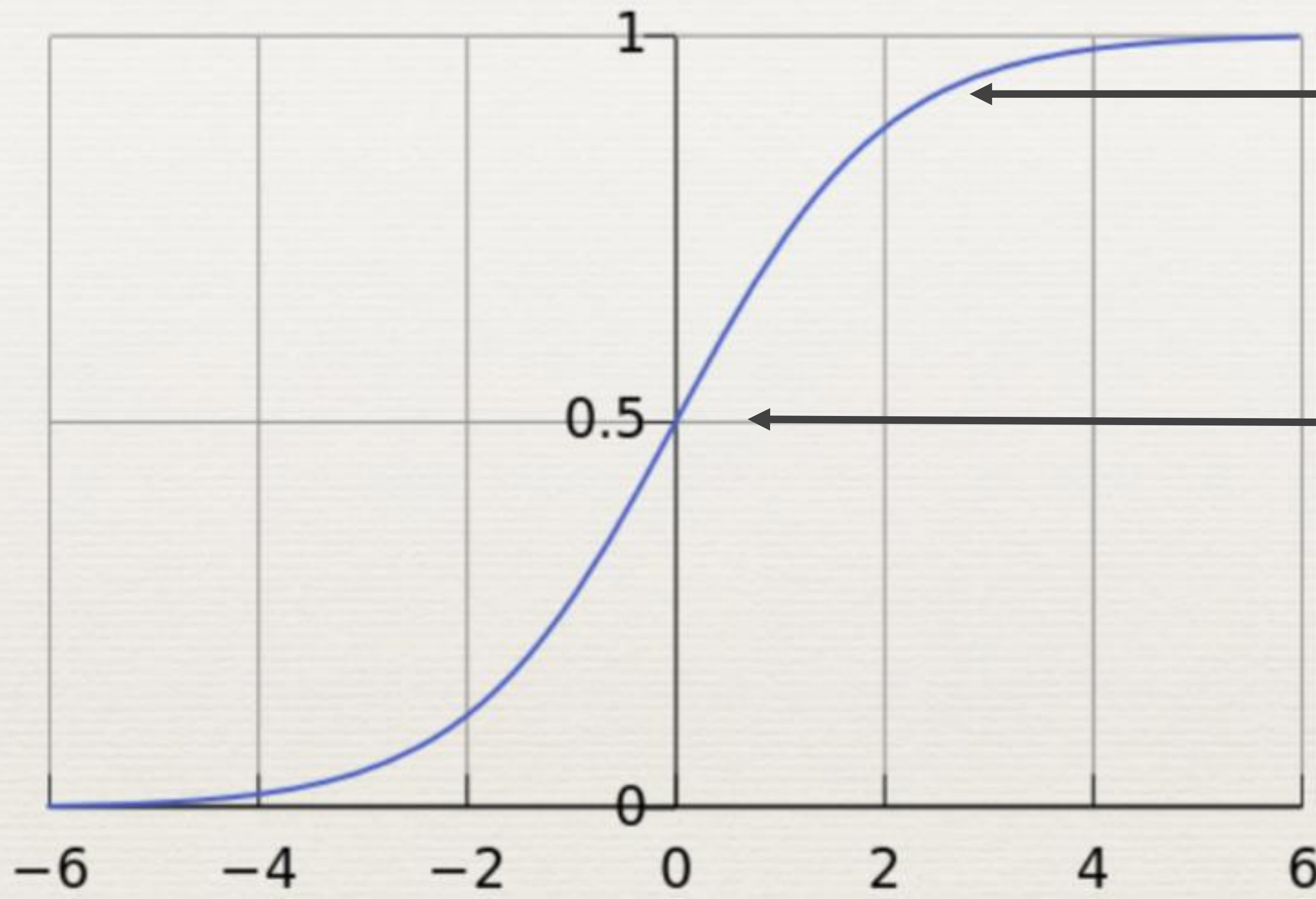


Heaviside step function
(פונקצית מדרגה)



Signum function
(sign – פונקצית הסימן)

הנוירון המלאכותי – מודל ב' – החלפת "פונקצית ההפעלה" - sigmoid *



פונקצית ה-sigmoid רציפה

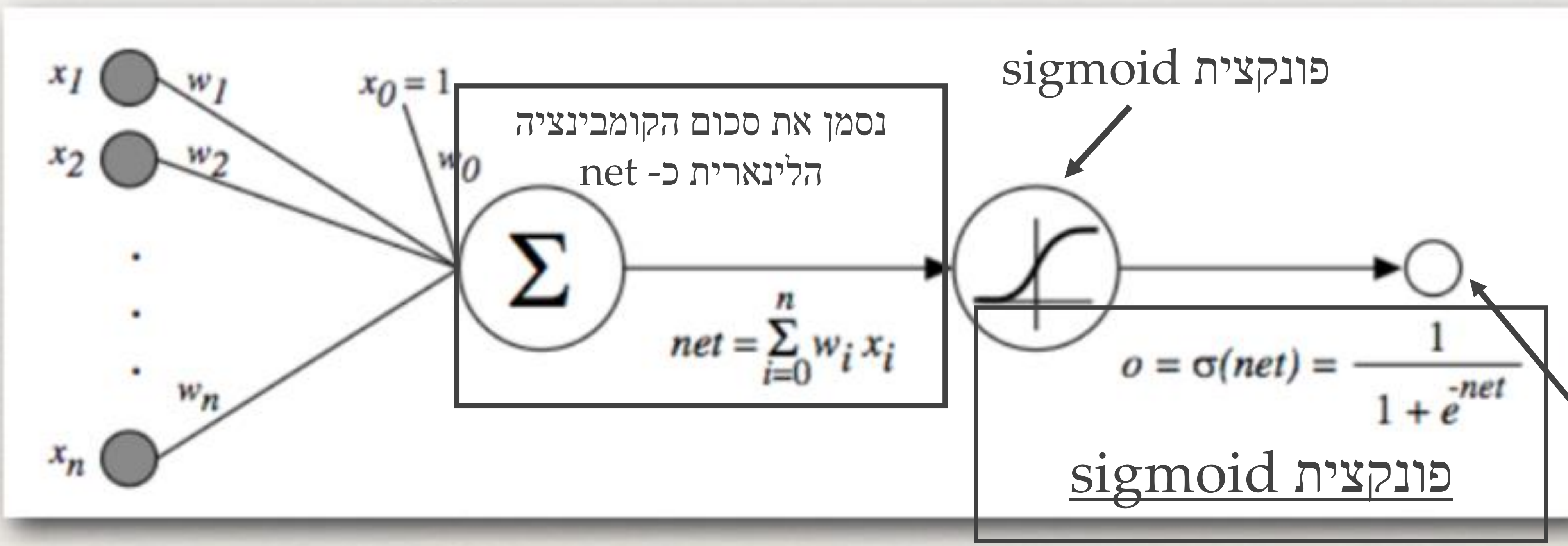
* מוסיפים כ- post processing שלב בדיקת סף, לקבלת החלטה על ערך הקטגוריה

* הסף פה הוא 0.5 (ולא 0)

- אם תוצאת הפונקציה גדולה מ-0.5,
- נסווג כקטגוריה החיובית, אחרת, כקטגוריה שלילית

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}.$$

הנוירון המלאכותי – מודל ב' – יחידת sigmoid



* בסוף רשת הנוירונים (ובסוף ה-sigmoid) מוסיפים שלב בדיקת סף, לקבלת החלטה על ערך הקטגוריה
* ערך הסף: 0.5

$\sigma(x)$ is the sigmoid function $\frac{1}{1+e^{-x}}$

פונקציית ה-sigmoid
שימו לב שפה ה-x משמש כסכום הקומבינציה הלינארית

$$\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$$

ה-sigmoid רציפה וגזירה, מה שחשוב ללמידה (נראה בהמשך)

דוגמה לקריאת נוירון בודד עם sigmoid

$$S(x) = \frac{1}{1+e^{-x}}$$

נתונה יחידת הנוירון הבאה, כאשר מישור ההפרדה מיוצג ע"י:

$$0.2 \cdot x_1 - 0.1 \cdot x_2 + 0.4 \cdot x_3 = 0$$

בצעו סימולציה לקבלת הפלט של הנוירון, תוך שימוש בגרף של נוירון ה-sigmoid, עבור הוקטור: (10,30,20)

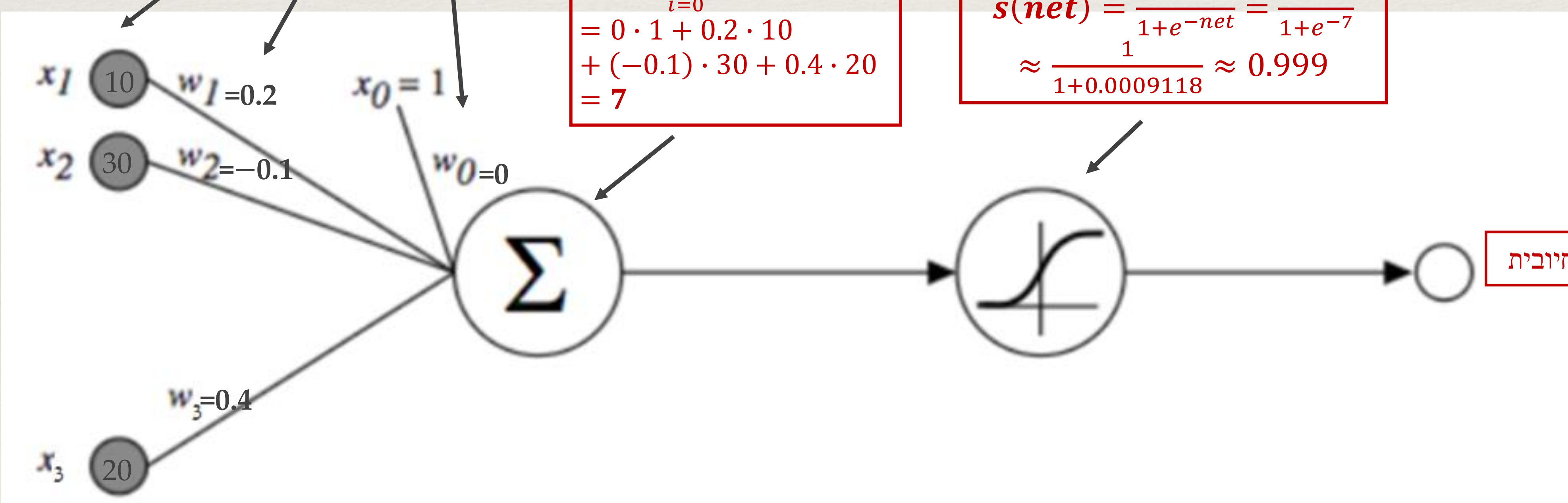
<u>שלב א'</u> – נציב את ערכי המשקולות וערכי ה-feature vector

<u>שלב ב'</u> - נכפיל ונסכום:

$$\boldsymbol{net} = \sum_{i=0}^{3} w_i x_i$$
$$= 0 \cdot 1 + 0.2 \cdot 10 + (-0.1) \cdot 30 + 0.4 \cdot 20$$
$$= \mathbf{7}$$

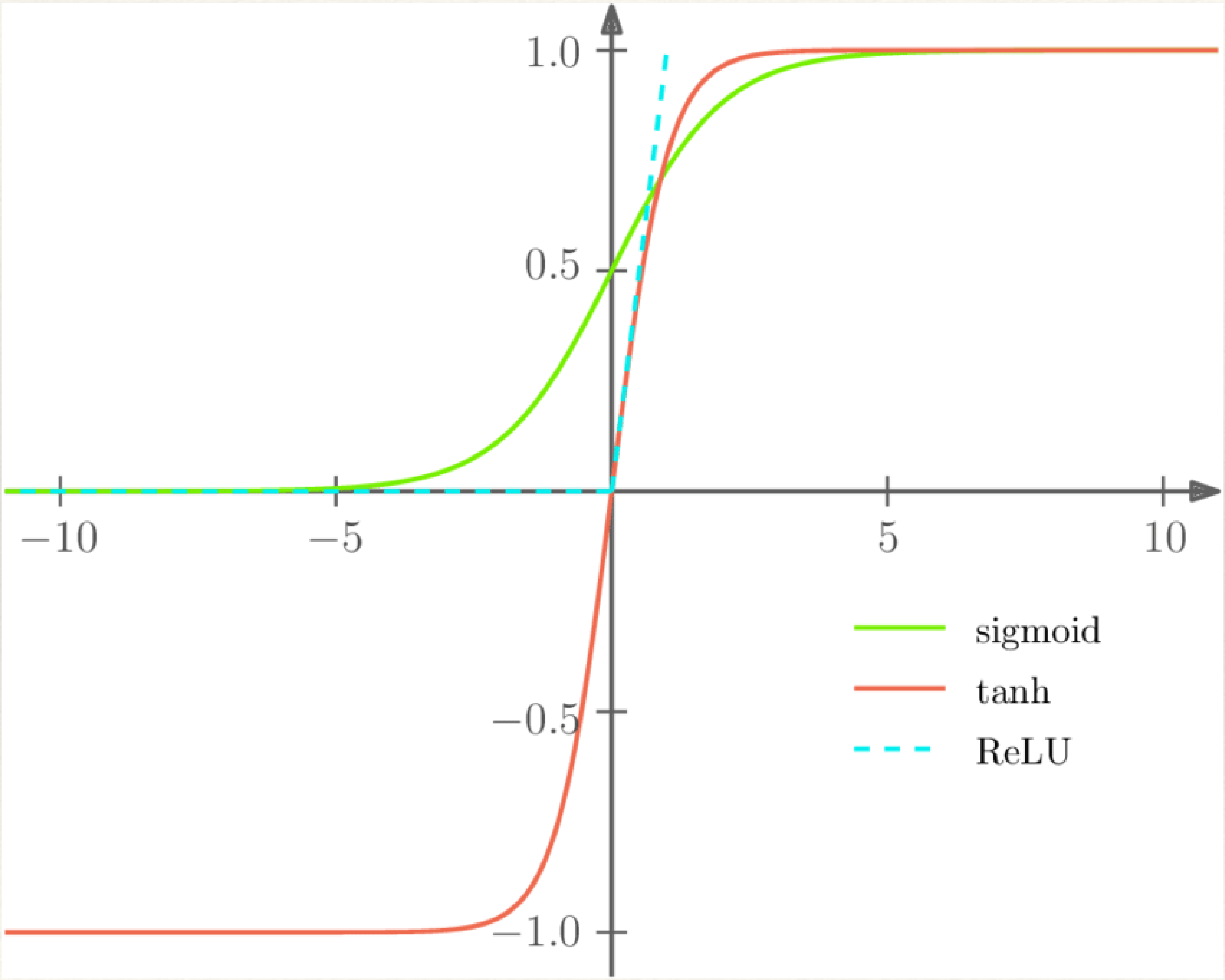
<u>שלב ג'</u> – נחשב את ערך הסיגמויד:

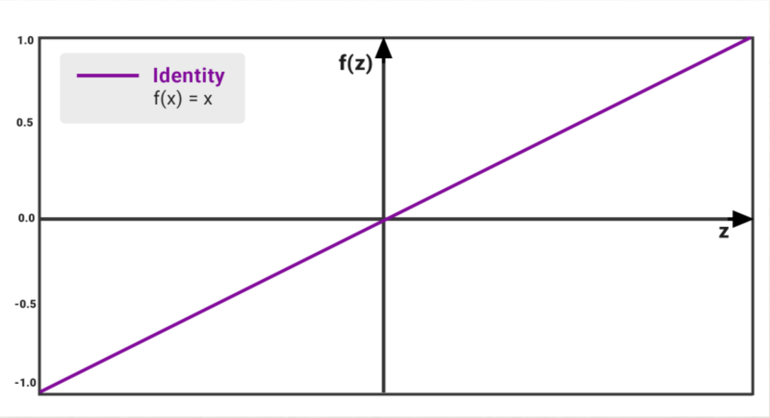
$$\boldsymbol{s(net)} = \frac{1}{1+e^{-net}} = \frac{1}{1+e^{-7}} \approx \frac{1}{1+0.0009118} \approx 0.999$$



<u>הסיווג</u> =המחלקה החיובית

פונקציות אקטיבציה נוספות





identity
ident(z) $= z$

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

tanh
tanh(z) $= \frac{e^{2z}-1}{e^{2z}+1}$

ReLU (Rectified Linear Unit)
R(z) = max(0,z)

בעיית סיווג

- קלט: n ווקטורים x, כל ווקטור באורך D – $x_i \in \Re^D$
- תווית y_i לכל ווקטור $y_i \in \{1,2,\ldots\ldots,k\}$

  נתמקד ב- $y_i \in \{-1, 1\}$
- לא ידועה לנו התפלגות הנתונים.

- אנו רוצים למצוא פונקציה $y=f(x)$ כך שבהינתן ווקטור חדש x המערכת תסווג אותו נכון בהסתברות גבוהה

פונקצית הפסד (Loss function) או
פונקצית עלות (Cost function)

<u>פונקצית הפסד או פונקצית מחיר</u> - פונקציה הממפה מאורע או ערכים של משתנה אחד או יותר למספר ממשי המייצג "עלות" של מאורע.

- בלמידה נסתכל למשל עלות של טעות בסיווג למשל

<u>נסמן</u>: J או loss – כפונקצית ההפסד

J(misclassification of i) – מחיר ההפסד של סיווג לא נכון

- סך ההפסד בלמידה, משתמש בשיטות כפי שראינו (ונראה עוד) בשערוך המודל.

<u>פונקצית מטרה (objective function)</u> –

בהקשר שלנו - פונקצית המטרה תהווה, בדרך כלל, פונקצית ההפסד או פונקצית הטעות, אותה נגדיר.

הנוירון המלאכותי – טענת ההתכנסות

- עבור דוגמאות הניתנות להפרדה לינארית, ועבור קבוע למידה (η) מספיק קטן, מובטח שהאלגוריתם יתכנס להיפותזה עקבית

- היפותזה עקבית = מודל עם משקולות שיכול לסווג דוגמאות באופן עקבי

בעיית סיווג

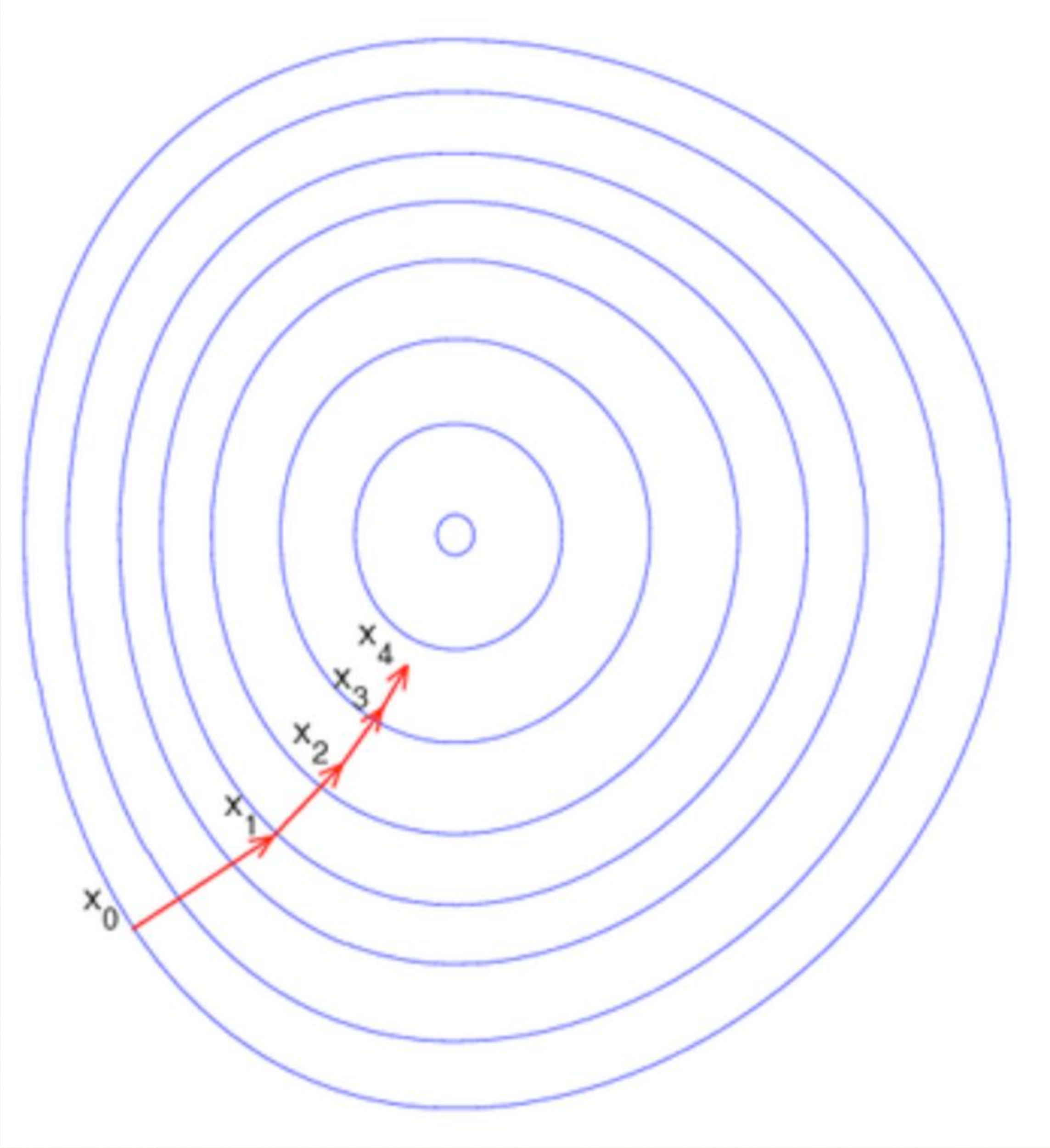
כלומר, אנו רוצים למצוא היפותזה במרחב ההיפותזות H, כך ש –

$$H \in \Re^D \longrightarrow \{-1, 1\}$$

כך שבהינתן "מחירון ענישה" לטעות $J(y,H(x)) = loss(y,H(x))$

- אנו נמצא את h שממזערת לנו את הטעות על ה-training set

תזכורת- Gradient Descent



- שיטה כללית לאופטימיזציה של פונקציה רבת משתנים
- השיטה למעשה מבצעת חיפוש hill climbing כשהצעד נקלח בכיוון הנגזרת (או השלילה שלה במקרה חיפוש מינימום)

תזכורת – Gradient Descent

המטרה:

- ניתן לראות תהליך למידה כתהליך חיפוש במרחב היפותזות (במקרה שלנו כל מודלי הסיווג האפשריים לבעיה שלנו)
- אנחנו מחפשים היפותזה שתתאים בצורה הטובה ביותר לקבוצת דוגמאות האימון

הרעיון:

- התאמה של המודל (ההיפותזה) בדרך כלל ממזערת שגיאה אמפירית
- loss function - לפונקציה שאותה רוצים למזער קוראים loss function

השיטה:

- אנחנו בעצם מבצעים חיפוש הפוך של המשקולות (המקדמים).
- במקום למצוא את המקדמים ישירות, מחפשים מקדמים בעלי השגיאה הקטנה ביותר על דוגמאות האימון (מודל שיסווג נכון ככל הניתן את דוגמאות האימון)
  - מזעור בד"כ באמצעות בשיטה הדומה לנגזרת (נסביר על כך עוד בשיעורים הבאים).
- תנאי סיום – ראו בסוף המצגת

תזכורת - תכונות – Gradient Descent

<u>פונקציית הפסד קמורה</u>

במקרה של perceptron (עם sigmoid) פונקציית ההפסד היא פונקציה קמורה

בגלל שפונקציית ההפסד קמורה, מובטח לנו שהאלגוריתם מתכנס להיפותזה בעלת שגיאה מינימלית, עבור קבוע למידה קטן מספיק

- הדבר נכון גם כאשר יש רעש בדוגמאות, וגם עבור קבוצת דוגמאות שאינה ניתנת להפרדה ליניארית

Gradient Descent בפרצפטרון (with <u>sign</u> function) – פסאודו קוד

<u>סימונים</u>:

η קבוע (קטן מ-1) הקובע את קצב הלמידה (למשל 0.1)

t סימון הדוגמא הנוכחית - ערך הקטגוריה האמיתי של הדוגמה

o הערך שנותן ה נוירון עבור הדוגמא הנוכחית

$\langle \vec{x}, t \rangle$ - נסמן כל דוגמה כזוג feature vector וקטגוריה

Δw_i - עדכון ל-w_i (הוספה ביחס לאיטרציה הקודמת)

<u>אלגוריתם</u>:

:Gradient Descent (training examples, η)

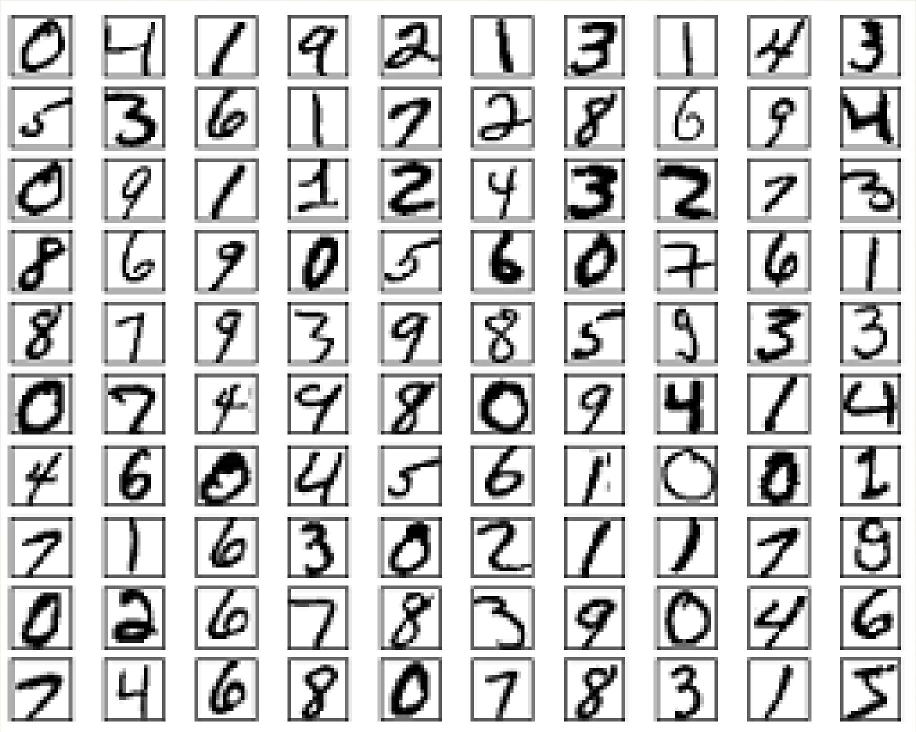
- לכל משקולת w_i, אתחל ערכים התחלתיים אקראיים קטנים
- מעדכנים את הפרמטרים עד להתכנסות:
  - אתחל כל Δw_i ל-0
  - עבור כל $\langle \vec{x},t \rangle$ בדוגמאות האימון
    - חשב את o ע"י הכנסת $\vec{x}$ כקלט ליחידת הנוירון
    - לכל משקולת w_i:
      - $\Delta w_i = \Delta w_i + \eta(t-o) x_i$
  - $w_i^{t+1} := w_i^t + \Delta w_i$

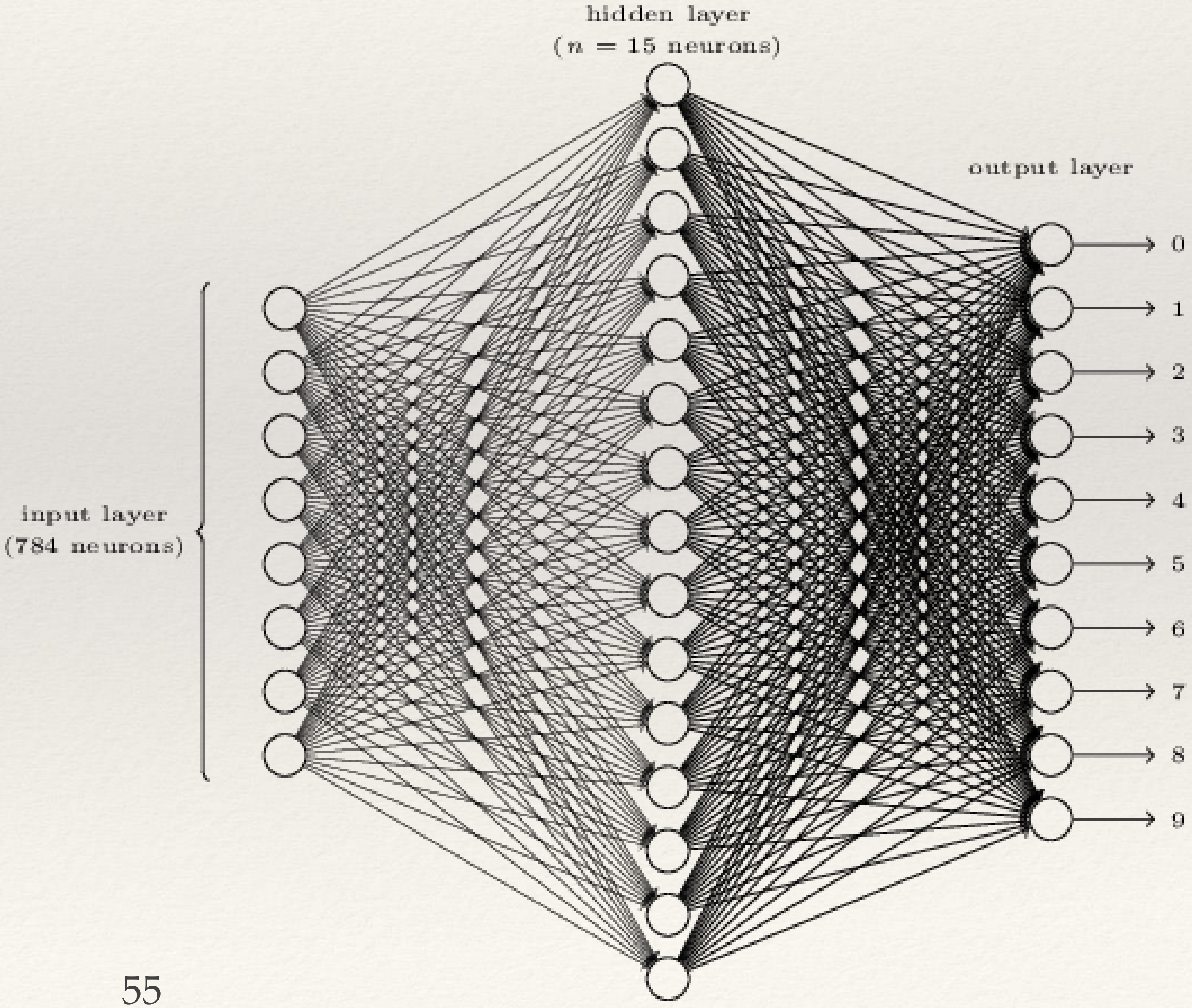
גרסה זו לא מהווה gradient descent (רק דומה לה), מכיוון שפונקצית ה-sign, אינה גזירה ואינה דפרנציבלית

ANN – Artificial Neural Network
רשתות רב שכבתיות מלאכותיות

רשתות רב שכבתיות - דוגמא

- אבחנה בין 10 סימנים שונים כדי להבין את הספרה שכל סימן מייצג





רשת רב שכבתית – מציאת מודל הסיווג - השיטה

אתחול פרמטרים אקראי

ננסה לשנות את הפרמטרים - במטרה לשפר את האיכות של המודל

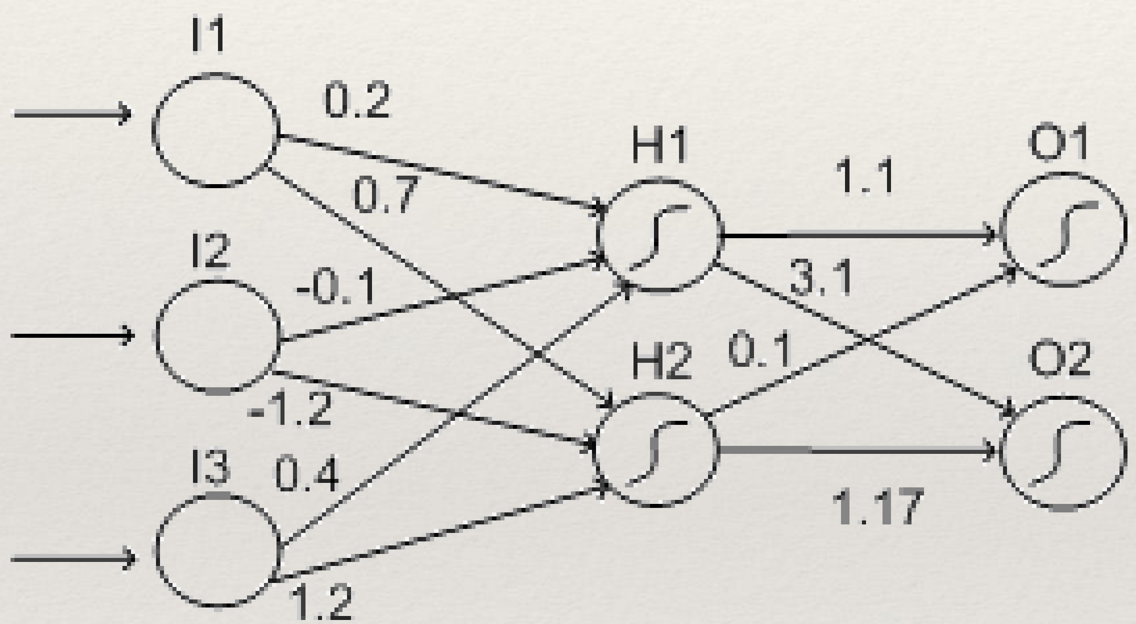
- השיטה: חיפוש ערכים טובים יותר של הפרמטרים, כדי שתוצאות המודל ישתפרו

איכות המודל = loss function =J= טעויות סיווג של דוגמאות האימון. שיפור המודל = מזעור פונקציות הloss

חיפוש ערכים טובים יותר = gradient descent = חיפוש חמדני על מגוון הערכים של הפרמטרים (מרחב ההיפותזה) – כדי לקבל loss (כלומר J) נמוך יותר

דוגמה לקריאת רשת רב שכבתית

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}.$$



בצעו סימולציה לקריאת הרשת הבאה,
הדוגמה אותה נרצה לסווג: (10,30,20)

אם היה מדובר בדוגמה חדשה:
פלט: O2 < O1 --> נסווג את הדוגמה כ-O2

שלב קריאת הרשת וחלחול הערך משכבת הקלט
עד לשכבת הפלט נקרא Feed Forward

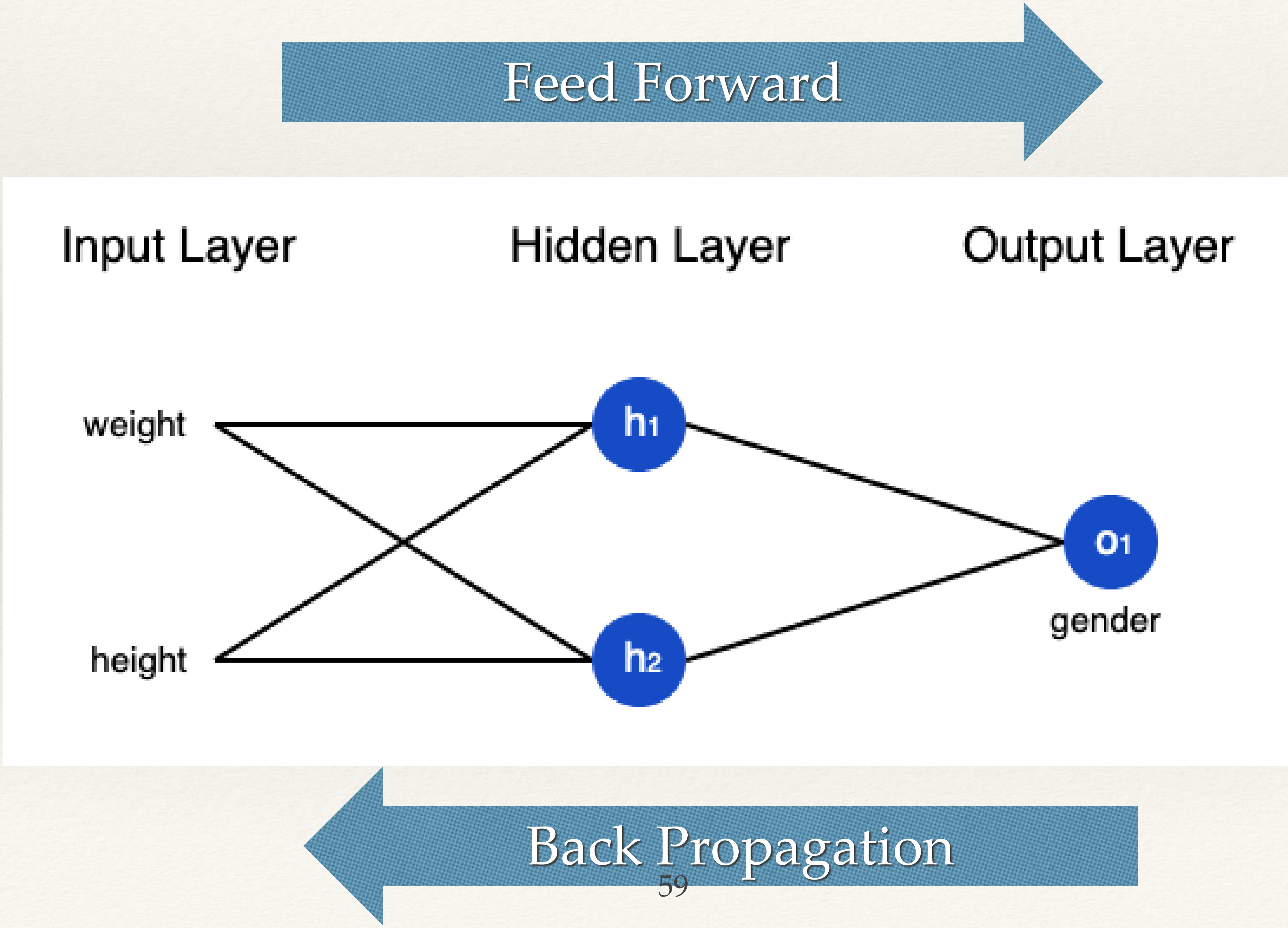
Input units		Hidden units			Output units		
Unit	Output	Unit	Weighted Sum Input	Output	Unit	Weighted Sum Input	Output
I1	10	H1	7	0.999	O1	1.0996	0.750
I2	30	H2	-5	0.0067	O2	3.1047	0.957
I3	20						

אם היה מדובר בדוגמה חדשה

רשת רב שכבתית – מציאת מודל הסיווג

- פונקצית ה loss (הטעות על שגיאות הסיווג של דוגמאות האימון) אותה רוצים למזער היא סכום השגיאות המרובעות על פני כל יחידות הפלט
- השיטה: האלגוריתם מבצע חיפוש gradient descent במרחב ההיפותזות
- היפותזה היא בעצם ווקטור של משקולות לכל יחידות הרשת
- בניגוד לחיפוש עם יחידה אחת, כאן הפונקציה אינה קמורה ויתכנו נקודות מינימום מקומיי (כלומר יתכן וישנה רשת טובה יותר, בעלת מינימום שגיאה גלובלית) – נסביר על כך בהרצאות בהמשך
- אלגוריתם backpropagation - האלגוריתם שנלמד למציאת מודל הסיווג ברשת נוירונים רב שכבתית.

backpropagation אלגוריתם



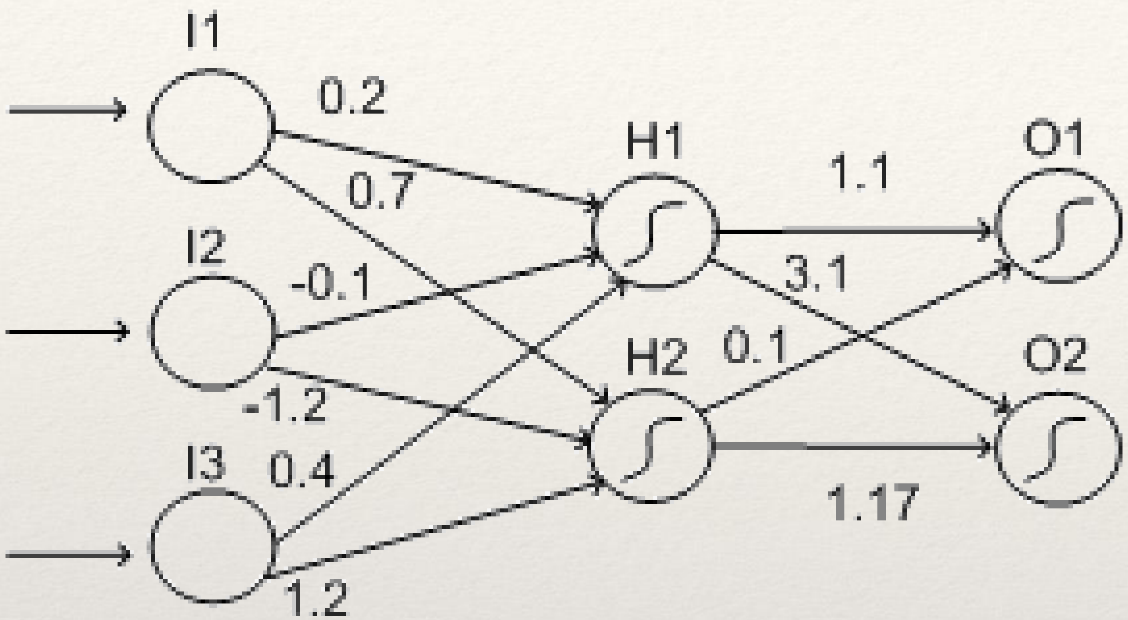
אלגוריתם backpropagation

- הלולאה הראשית של האלגוריתם מריצה את כל קבוצת האימון שוב ושוב עד שאנחנו מרוצים (עד להתכנסות):
  - בלולאה הפנימית, מריצים דוגמה אחר דוגמה:
    - feed forward – משתמשים בערכי המשקולות בשלב הנוכחי, מחשבים את ערכי הפלט של השכבות משכבות הביניים, שכבה אחר שכבה עד לשכבת ה-output
    - Back propagation - חישוב השגיאה וחלחול לאחור (בכל פעם משנים את המשקלות בכיוון הנגזרת (כלומר מורידים את הטעות))
      - מחשבים את הטעות על פני כל יחידות הפלט, ומתקנים את המשקולות (בדומה ליחידה בודדת של נוירון מלאכותי עם signum או sigmoid)
      - מחשבים את הטעות על פני כל היחידיות הנסתרות, ומתקנים את המשקולות

אלגוריתם backpropagation –
חישוב נגזרת השגיאה (לצורך תיקון המקדמים)

- <u>חישוב עבור יחידות הפלט</u>, השגיאה מחושבת כמו קודם, אבל ההכפלה היא בנגזרת פונקציית הסיגמויד
- <u>חישוב עבור היחידות הנסתרות</u>, הדוגמא לא מספקת ערך שגיאה
  - לכן, עבור יחידה נסתרת, השגיאה היא סכום השגיאות המשוקלל של יחידות הפלט שקשורות אליה

דוגמה לתחילת שלב back propagation



$\varepsilon(O_1) = 0.1$

$\varepsilon(O_2) = 0.05$

נניח שקיבלנו את הטעויות הבאות עבור שכבת הפלט חשבו מה יהיה השוני שיחלחל לטעות של H1?

תשובה –

$1.1 \cdot \varepsilon(O_1) + 3.1 \cdot \varepsilon(O_2) =$

$= 1.1 \cdot 0.1 + 3.1 \cdot 0.05 = 0.265$

רשתות רב שכבתיות – אלגוריתם backpropagation

backpropagation(train_set, η, topology):

Initialization: Initialize all weights ($w_{i,j}$) to small random numbers.

Outer loop: for all train_set Until satisfied, Do

- For each training example $\langle \vec{x}, t_k \rangle$, Do

Feed forward:

1. Input the training example to the network and compute the network outputs (o_k)

Back propogation:

2. For each output unit k

$$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k)$$

3. For each hidden unit h

$$\delta_h \leftarrow o_h(1-o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

4. Update each network weight $w_{i,j}$

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$

where

$$\Delta w_{i,j} = \eta \delta_j x_{i,j}$$

נתון כקלט לאלגוריתם:

Train set– כל זוגות האימון $\langle \vec{x}, t_k \rangle$.

$\vec{x}$ - מייצג את ה-feature vectors של האימון.

x_i - ערך מאפיין i (x_iים מהווים את שכבת הקלט)

t_k - יכול להיות כמה ערכים (אחד עבור כל יחידה חיצונית)

η – קבוע הלמידה (מספר הקטן מ-1, גדול מ-0)

topology – המבנה של רשת הנוירונים

יחידה חיצונית:

o_k - פלט עבור יחידה חיצונית k

t_k - פלט מצופה עבור יחידה חיצונית k

δ_k - הטעות (בעזרת גרדיאנט) בשכבה חיצונית k

יחידה נסתרת:

o_h - פלט עבור יחידה נסתרת h

הערה: אין t_h - פלט מצופה עבור יחידה נסתרת h

δ_h - הטעות עבור יחידה נסתרת h (לפי התרומה היחסית ליחדות הפלט החיצוניות)

$w_{h,k}$ - המשקולת בין יחידה נסתרת h ליחידה חיצונית k

כללי (i - קלט / יחידה נסתרת, j –יחידה נסתרת / חיצונית)

$w_{i,j}$ - משקולת הקשת בין i ל-j,

$\Delta w_{i,j}$ - התיקון למשקולת הקשת בין i ל-j

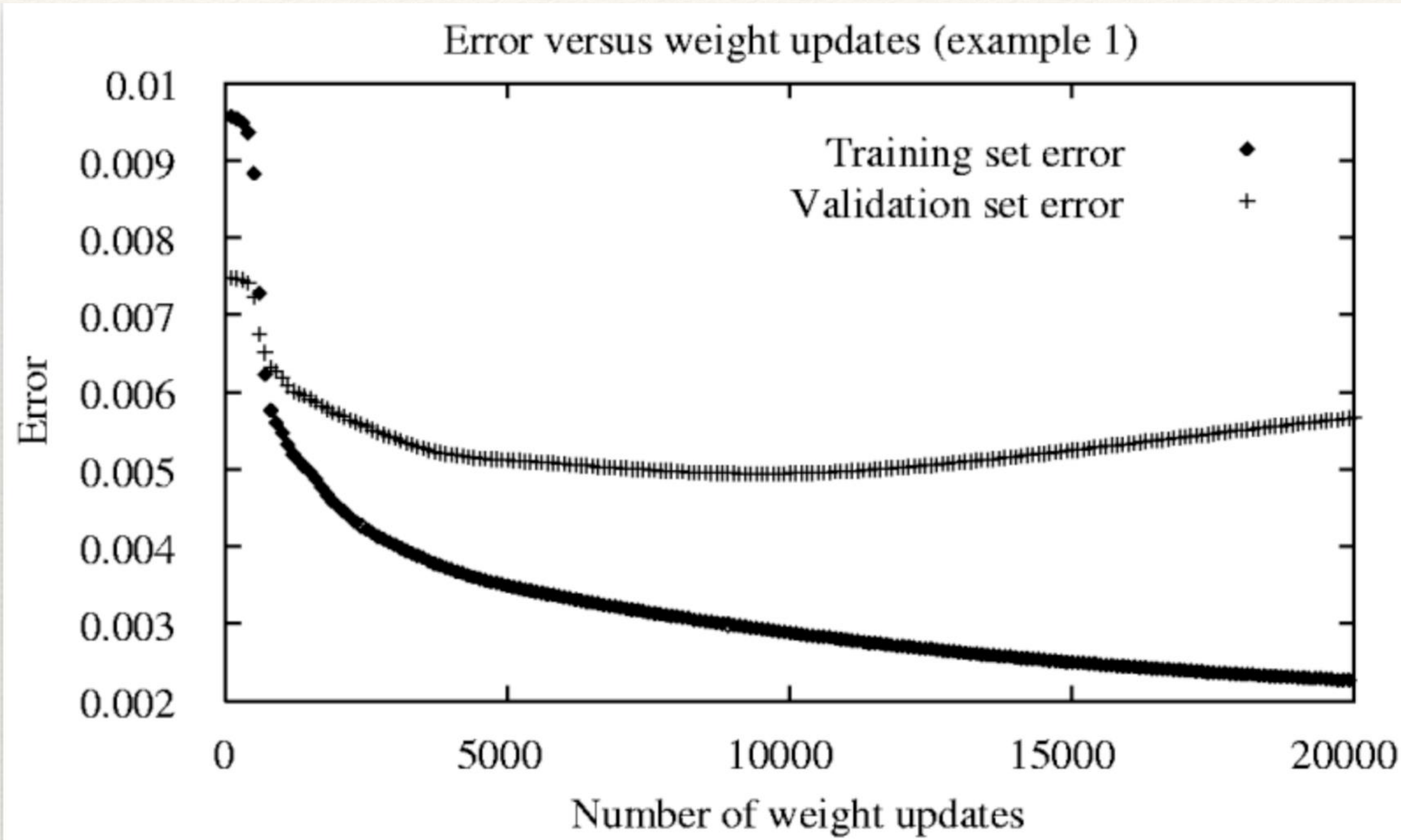
$x_{i,j}$ - הקלט המחובר לקשת $w_{i,j}$

δ_j - הטעות בשכבה j

אלגוריתם backpropagation (ו-gradient descent) – תנאי סיום הלולאה הראשית (על כל דו' האימון)

- סיום מוקדם מדי יגרום להיפותזה עם שגיאה גדולה מדי
- סיום מאוחר מדי יגרום ל overfitting (ולא ייצג את העולם האמיתי)
- תנאי אפשריים לסיום:
  - מספר קבוע של איטרציות
  - ערך סף לשגיאת האימון
  - הפרש בין השגיאות ב2 איטרציות יורד מתחת לסף נתון
  - אין שינוי במשקולות / בפלט של הנוירונים
  - ביצועים על validation set

תזכורת ל overfitting



רשתות עצביות - סיכום

- יתרונות:
- אלגוריתם מחפש חיפוש חמדני על אוסף משקלות הרשת
- בפועל אלגוריתם אפקטיבי (למרות שיש חשש להתכנסות למינימום לוקלי)
- מסווג מהיר בזמן ריצה
- עשוי להגיע לתוצאות מאד טובות
- יכול ללמוד פלט: בוליאני, קטגורי, רציף או ווקטור

- חסרונות:
- אלגוריתם אימון איטי
- לא קריא לבני אדם
- עלול להתכנס למינימום לוקלי