(20942) מבוא ללמידה חישובית | סיכום הרצאה 2

מנחה: שי מימון

סמסטר: 2022א'

נכתב על ידי: מתן כהן

1 פרספטרון - The Perceptron Learning Algorithm (PLA)

הומצא על ידי פרנק רוזנבלט ב-1957 ונחשב לאחד האלגוריתמים החשובים בשל היותו אחד האלגוריתמים שביססו את מושג רשתות הנוירונים. נתחיל בדוגמה פשוטה על מנת להבין את המודל.

דוגמה. פרספטרון בשני ממדים

- נגדיר וקטור $\underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- מודל הפרספטרון יניב: $h(\underline{x}) = sign(x_1w_1 + x_2w_2 + w_0)$ כאשר:

$$sign(x) = \begin{cases} 1 & x > 0 \\ -1 & x \leq 0 \end{cases}$$

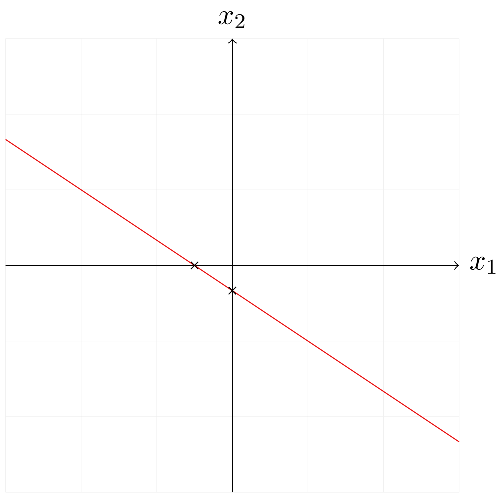
- למען הדוגמה, נניח ש-

$$w_0 = 1, \ w_1 = 2, \ w_2 = 3$$

ולכן:

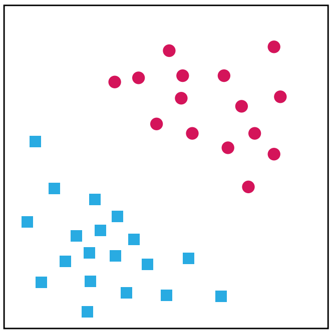
$$h(\underline{x}) = 1 + 2x_1 + 3x_2$$

- כעת נוכל לבצע ויזואליזציה של הפרספטרון, לשם כך נציב ערכי 0 ב- x_1 ו-x_2 על מנת למצוא 2 נקודות על הצירים (שהם בעצמם (x_1, x_2)) ולהעביר ביניהן קו:
  - $\star$ $1 + 3x_2 = 0 \iff x_2 = -\frac{1}{3}$
  - $\star$ $1 + 2x_1 = 0 \iff x_1 = -\frac{1}{2}$
  - $\star$ נקבל את הגרף הבא בו כל ערך מימין לקו הלינארי הוא 1 וכל מה שמשמאלו הוא -1:

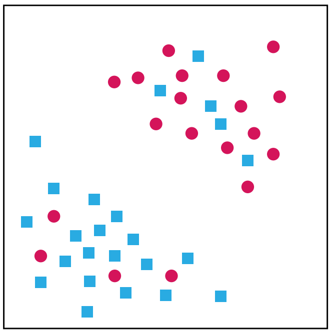


הערה. בממד גדול מ- 2 הקו הלינארי יקרא **hyperplane** - תפקידו לחלק את המרחב שלנו לשני חלקים

כאשר נקבל דוגמאות אימון נוכל להתקל בשני מצבים:



הפרדה אפשרית



הפרדה לא אפשרית

מסקנה. **אלגוריתם PLA** מניח שסט הדוגמאות **מקבוצת האימון** ניתן להפרדה בצורה לינארית

1.1 הפרספטרון - פישוט הייצוג

דיברנו על כך שהפרספטרון שלנו מיוצג בצורה הבאה:

$$h(\underline{x}) = sign\left(\sum_{i=1}^{d} w_i x_i + b\right)$$

כאשר:

- $\underline{w}$ - הוא וקטור המשקלים המתאימים לפיצ'רים שלנו
- $\underline{x}$ - הוא וקטור הפיצ'רים שלנו
- b - הוא ההיסט ($bias$) שלנו

נבחין כי ההפרדה בין ההיסט b לשאר הפרמטרים קצת "מפריעה" ולכן נרצה לרשום את הייצוג בצורה יותר נוחה.

לשם כך ניזכר כי כל וקטור ממד d מקיים: $\hat{\underline{x}} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$ ולכן נוכל להגדיר בצורה פשוטה וקטור ממד $d+1$ $\underline{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$

ולשנות את הנוסחה כך ש:

$$h(\underline{x}) = sign\left(\sum_{i=0}^{d} w_i x_i\right)$$

כאשר בעצם מתקיים: $w_0 = b$ וכמובן $x_0 = 1$

כעת נוכל לייצג את ה- perceptron בצורה וקטורית **מאוד** פשוטה.

עבור $\underline{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$ ו- $\underline{w} = \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$:

$$h(\underline{x}) = sign(\underline{w}^T \underline{x})$$

1.2 צוללים פנימה - אלגוריתם PLA

כעת, בהינתן סט דוגמאות מתויגות בצורה בינארית אשר ידוע מראש שניתן להפרידן בצורה לינארית - נרצה למצוא את סט הפרמטרים $\underline{w}$ אשר יפרידן על ידי אימון מודל.

כיוון שהאלגוריתם הוא אלגוריתם איטרטיבי, נרצה להתחיל עם "ניחוש" התחלתי ולהתחיל לתקנו.

- נניח שהניחוש ההתחלתי שלנו הוא $\hat{\underline{w}(0)} = \underline{0}$ (ניתן לבחור בצורה שונה - נדבר על כך בהמשך)
- נחפש נקודה כלשהיא (בצורה אקראית או על פי חיפוש של אחת כזו) אשר באיטרציה t התיוג שלה לא נכון
  - במילים אחרות, אם $y(t)$ הוא התיוג של דוגמה $\underline{x}$ באיטרציה t, ונסמן $sign\left(\hat{\underline{w}}^T(t)\cdot\underline{x}(t)\right) = \hat{y}(t)$ (התיוג הלא נכון שביצע האלגוריתם באיטרציה t) אז

$$\hat{y}(t) \neq y(t)$$

- האלגוריתם מציע לעדכן (update rule) את סט הפרמטרים $\underline{w}$ בצורה הבאה:

$$\boxed{\hat{\underline{w}}^T(t+1) = \hat{\underline{w}}^T(t) + y(t)\cdot\underline{x}(t)}$$

האינטואיציה מאחורי העידכון

- כפי שאמרנו, במידה והדוגמה לא מתויגת נכון על ידי האלגוריתם מתקיים: $\hat{y}(t) \neq y(t)$ וכן:

$$y(t) \in \{-1,1\} \quad (1)$$
$$sign\left(\hat{\underline{w}}^T(t)\cdot\underline{x}(t)\right) = \hat{y}(t) \neq y(t) \quad (2)$$

ולכן:

$$\hat{\underline{w}}^T(t)\cdot\underline{x}(t)\cdot y(t) < \mathbf{0}$$

- כעת, נרצה למצוא ביטוי גדול יותר מהביטוי הנ"ל
- לשם כך נרצה שבאיטרציה $t+1$ התיוג שלנו יהיה **גדול יותר** מהתיוג באיטרציה t (כיוון שצריך לשנות את התיוג לסימן חיובי בסופו של דבר) ובכך לגרום לתיוג להיות "**יותר נכון**"
- על כן נבחין שהעידכון שהאלגוריתם מציע מניב לנו את התוצאה הבאה:

$$y(t)\cdot\underline{x}(t)\cdot\hat{\underline{w}}^T(t+1) = y(t)\cdot\underline{x}(t)\cdot\left[\hat{\underline{w}}^T(t) + y(t)\cdot\underline{x}(t)\right] = y(t)\cdot\underline{x}(t)\cdot\hat{\underline{w}}^T(t) + \overbrace{\underbrace{y(t)^2}_{=1}\|\underline{x}\|^2}^{>0}$$

$$\Rightarrow \boxed{y(t)\cdot\underline{x}(t)\cdot\underline{w}^T(t+1) > \hat{\underline{w}}^T(t)\cdot y(t)\cdot\underline{x}(t)}$$

- וזה בדיוק מה שרצינו

בעיה. מה קורה אם אנחנו מנסים להתכנס בשביל לשפר את התיוג של דוגמה מסוימת? הרי ישנו סיכוי שנפגע בזיהוי של דוגמאות אחרות תוך כדי ניסיון לשפר את התיוג של דוגמה ספציפית זו.

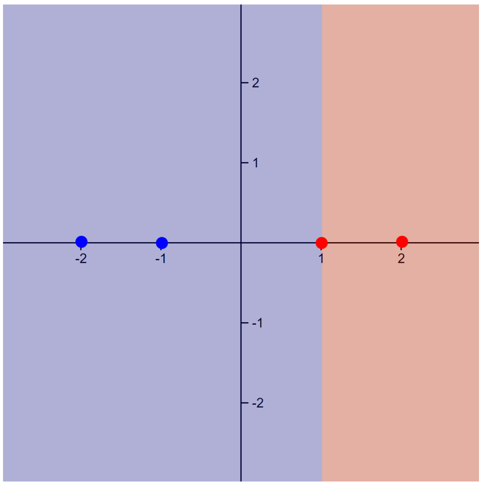
פתרון. ניתן להוכיח (שאלה בממ"ן 11) שאם הדוגמאות מקבוצת האימון ניתנות להפרדה בצורה לינארית (הנחת האלגוריתם), אזי האלגוריתם יתכנס לאחר מספר סופי של צעדים למפריד שיפריד בצורה **מושלמת** את הדוגמאות. במילים אחרות נקבל שעבור כל דוגמה $\underline{x}_i$:

$$y_i = h(\underline{x}_i)$$

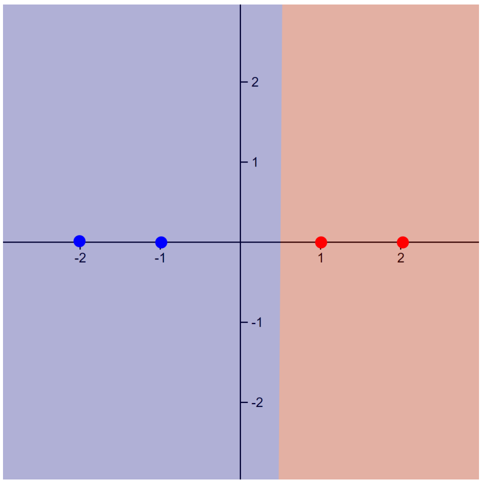
1.3 התכנסות האלגוריתם

יש צורך לציין שהתכנסות האלגוריתם יכולה להשתנות על פי התיקונים שמבצעים בזמן הריצה. במידה ונשתמש בלינק הבא ונכניס נתונים אל הגרף המצוי בתוכו, נבחין כי אופן בחירת הנקודה הראשונה על פיה האלגוריתם יעבוד - משנה את ה-hyperplane שנקבל!

דוגמה.



בחירת הנקודה $(1,0)$ ולאחר מכן $(-1,0)$



בחירת הנקודה $(-2,0)$ ולאחר מכן $(1,0)$

מסקנה. **ישנם אינסוף פתרונות** להפרדת דוגמאות בצורה לינארית מושלמת.

כמו-כן נבחין כי:

- הפתרון שבסופו של דבר נתכנס אליו (המפריד הלינארית עצמו) תלוי ב-2 קריטריונים:
  (1) האיתחול של וקטור המקדמים $\underline{\hat{w}}$
  (2) סדר הצגת הדוגמאות לאלגוריתם

הערה. יש לציין כי שני הקריטריונים מעלה אינם משנים את העובדה שהאלגוריתם יתכנס לאחר מספר **סופי** של צעדים **אם** הדוגמאות **בקבוצת האימון** ניתנות להפרדה בצורה לינארית!

1.3.1 מה קורה עם סט אימון שלא ניתן להפרדה בצורה לינארית?

כיוון שהפרספטרון **מניח** כי ניתן להפריד את הדוגמאות בסט האימון בצורה ליניארית נוכל להסיק:

מסקנה. אם סט דוגמאות האימון שמקבל אלגוריתם PLA לא ניתן להפרדה לינארית - לעולם לא נתכנס להפרדה שכזו.

במילים אחרות, האלגוריתם יגיע למצב שהוא מנסה בכל איטרציה לתקן את המפריד הלינארי על ידי עידכון שמתבסס על נקודה שתויגה לא נכון וימשיך כך בלי להתכנס.

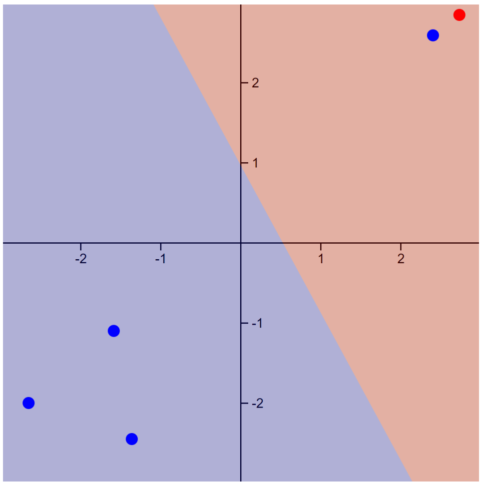
בעיה. מתי נדע האם "נתקענו" במצב בו לא ניתן להפריד?

הרי יכולים להתקיים שני מקרים שקשה לאלגוריתם להבדיל ביניהם:

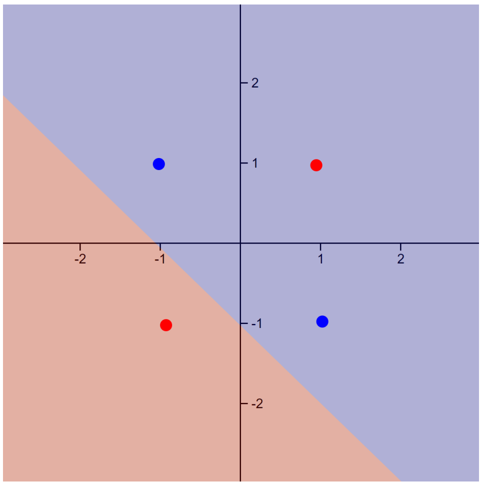
(1) האלגוריתם לא מצליח להתכנס כיוון שהדוגמאות בקבוצת האימון לא ניתנות להפרדה לינארית (כמו בדוגמה מעלה)

(2) ניתן להפריד את הדוגמאות שבקבוצת האימון בצורה לינארית - אך לאלגוריתם לוקח המון זמן להתכנס

דוגמה.



מצב בו לאלגוריתם לוקח זמן להתכנס



מצב בו האלגוריתם לא יכול להתכנס

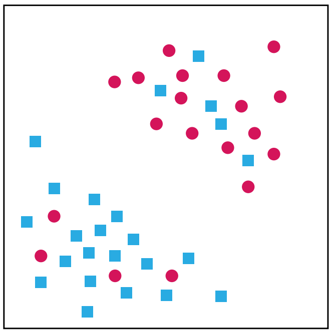
ואכן, כפי שנוכיח בממ"ן 11 - מספר הצעדים עד ההתכנסות תלוי ב-2 גורמים:

(1) הנורמה המקסימלית של הדוגמאות בסט האימון

(2) המרווח הקטן ביותר בין 2 נקודות המתויגות בצורה שונה (שלילית וחיובית)

1.4 דוגמאות שלא ניתנות להפרדה - Non-Separable Data

ניזכור כי קיימים מקרים בהם לא ניתן להפריד את סט דוגמאות האימון בצורה לינארית



ניזכור, כי המטרה שלנו היא **למצוא מפריד שיביא למינימום את מספר השגיאות בקבוצת האימון**

לפיכך, יכלנו להסתכל על מדד ה- in-sample error (שמסומן E_{in}) עליו דיברנו בשיעור 1 בצורה הבאה:

$$\min_{w\in\mathbb{R}^{d+1}} \underbrace{\frac{1}{N}\sum_{n=1}^{N}\left[sign\left(w^{T}x\right)\neq y_{n}\right]}_{E_{in}(w)}$$

במילים אחרות: נרצה למצוא את וקטור המקדמים $\underline{w}$ מממד $\mathbb{R}^{d+1}$ אשר יניב לנו את ממוצע שגיאות התיוג הנמוך ביותר.

הבעיות העומדות בדרכינו בחישוב זה הן בעיות קומבינטוריות קשות - אין אלגוריתם יעיל שיכול לפתור את הבעיה הנ"ל - ובעצם אין אלגוריתם שיכול למצוא לנו $\underline{w}$ אופטימלי שכזה.

בסעיף הבא נבחן דרך להתגבר על הבעיה.

1.4.1 אלגוריתם Pocket

אלגוריתם Pocket הוא אלגוריתם שמנסה לגשר על הבעיה הקומבינטורית הקשה שהוזכרה.
אלגוריתם זה מקבל מגבלת איטרציות T עבור אלגוריתם פרספטרון ומנסה למצוא $\underline{w}$ אופטימלי **במגבלה זו**

Algorithm 1: The Pocket Algorithm

Let T be the maximum number of iterations;
Set the pocket weight vector $\underline{\hat{w}}$ to $\underline{w}(0)$ of PLA;
for $t = 0$ **to** T **do**
    Run PLA for one update to obtain $\underline{w}(t+1)$;
    Evaluate $E_{in}(\underline{w}(t+1))$;
    if $\underline{w}(t+1)$ is better than $\underline{\hat{w}}$ in terms of E_{in} **then**
        $\underline{\hat{w}} \leftarrow \underline{w}(t+1)$;
return $\underline{\hat{w}}$

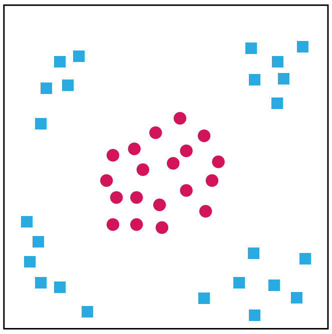
נקודה חשובה

ההבדל בין אלגוריתם Pocket לאלגוריתם PLA הוא העובדה שאלגוריתם PLA דוגם אקראית דוגמאות עד שמוצא דוגמה כלשהי שתויגה לא נכון (לכל היותר עובר על כל הדוגמאות) ולאחר מכן מעדכן את ההפרדה על פיה, לעומת זאת אלגוריתם Pocket מבצע בכל איטרציה אבלואציה של E_{in} מה שמכריח אותו לעבור על כל הדוגמאות - זה תהליך יקר

1.5 דוגמאות שניתנות להפרדה לא לינארית - Non-Linearly Separable Data

לעיתים נתקל במצב בו ניתן להפריד את סט האימון **אך לא בעזרת מפריד לינארי**

דוגמה. מצב בו מפריד אליפטי היה מצליח יותר טוב ממפריד לינארי



כעת, על מנת לפתור את הבעיה שנוצרה, בואו נחזור שניה אחורה ונתבונן בנוסחה:

$$\underline{w}^T\underline{x} = \sum_{i=0}^{d} w_i x_i$$

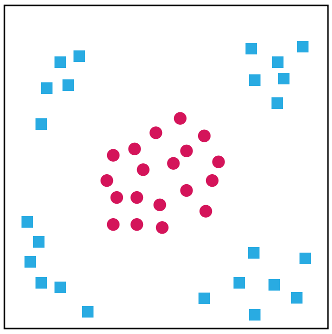
נוסחה זו לינארית **גם** ב-$\underline{x}$ **וגם** ב-$\underline{w}$. אך צריך להבחין בעובדה ש- $\underline{x}$ הוא וקטור **הפיצ'רים** ואכן **רוב** האלגוריתמים הלינארים שנדבר עליהם מתסמכים על התלות הלינארית ב-$\underline{w}$ **אך לא ב-**$\underline{x}$ שהוא וקטור של מספרים נתונים!

אבל איך זה קשור בכלל? **נסביר:**

- נתבונן בדוגמה של "אישור אשראי" ללקוח בבנק
- הלקוח נותן אינפורמציה לבנק על נתונים כמו: משכורת, מספר השנים שחי בבית בו שמתגורר וכו'
- כעת נתבונן בנתון כמו "מספר השנים שמתגורר באותו בית"
  - מצד אחד מספר השנים יכול להעיד על יציבות
  - מאידך לא משנה אם האדם גר 7 שנים או 8 שנים באותו בית - ההבחנה הגדולה היא האם מתגורר בבית זה יותר מ-5 שנים או פחות משנה (לדוגמה)
  - בעצם - **לא היינו מצפים** שההשפעה של מספר השנים שאדם מתגורר בבית מסוים תהיה לינארית!
  - במידה מסוימת קיימת אפשרות להשתמש באינדיקטור שיהפוך את הפיצ'ר הנ"ל לבינארי (מעל 5 שנים? פחות משנה?)

מסקנה. לפעמים נעדיף לא להשתמש בפיצ'רים מסוימים כפי שהם, אלא לעשות **טרנספורמציה כלשהיא** על הפיצ'רים שתעזור לנו במטרת הסיווג שלנו.

כעת, לאחר שדיסקסנו על האפשרות לבצע טרנספורמציה על מידע, נחזור שוב לדוגמה מעלה:



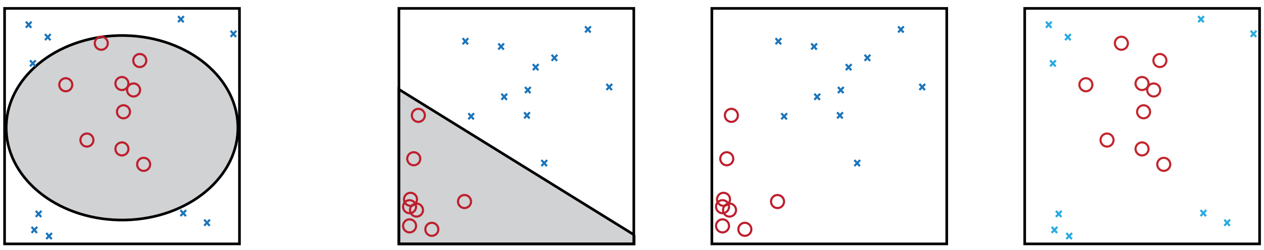
ניתן לראות בבירור שאפשרי לבצע טרנספורמציה מסוימת על המידע (הפיצ'רים) על מנת להפריד את הדוגמאות.
כפי שכבר הזכרנו, שימוש בהפרדה אליפטית היה עוזר לנו, וכפי שידוע נוסחה למעגל היחידה (סביב $(0,0)$) היא $x_1^2+x_2^2=r^2$.
נשתמש בנוסחה ונוכל להעביר אגפים ולקבל:

$$x_1^2+x_2^2-r^2=0 \underbrace{\Rightarrow}_{x_1^2=z_1 x_2^2=z_2} z_1+z_2-r^2=0$$

וכמובן שהנוסחה $z_1+z_2-r^2=0$ מוכרת לנו - **זה קו ישר** (hyperplane)!
בעצם - נמפה כל סקלר בוקטור $\underline{x}$ בעזרת פונקציה לא לינארית $z=\Phi(\underline{x})=(1,x_1^2,x_2^2)$ ונקבל:

$$h(\underline{x})=sign\left(w_0\cdot\underbrace{1}_{z_0}+w_1\cdot\underbrace{x_1^2}_{z_1}+w_2\cdot\underbrace{x_2^2}_{z_2}\right)$$

מהלך הפתרון:



פיצ'רים במרחב מקורי

פיצ'רים במרחב החדש אחרי טרנספורמציה $z_n=\Phi(x_n)$

פיצול הדוגמאות במרחב Z החדש בעזרת $\hat{g}(\underline{z})=sign(\underline{\hat{w}}^T\underline{z})$

חזרה למרחב המקורי והגדרת $g(\underline{x})=\hat{g}(\Phi(\underline{x}))=sign\left(\underline{\hat{w}}^T\Phi(\underline{x})\right)$

עם זאת, כפי שצוין, הטרנספורמציה Φ מיוחדת עבור המרכז $(0,0)$ ועל מנת לגרום לה להיות יותר גנרית נוכל לשנותה ולהשתמש בטרנספורמציה **קוואדרטית**:

$$\Phi_2(\underline{x})=(1,x_1,x_2,x_1^2,x_1x_2,x_2^2)$$

הערה. ניתן גם להמשיך ולפתח את הטרנספורמציה ולעלות בדרגת הפולינום המתקבל בוקטור החדש, בכך נקבל קו מפריד מדויק מאוד סביב הדוגמאות - יש לזכור שהדבר מגדיל את מספר הפיצ'רים ויכול לגרום למצב של **overfitting** (מצב שבו המודל מייצג יותר מידי את דוגמאות האימון ולא יפעל טוב בעולם האמיתי)!