אורט בראודה
המכללה האקדמית להנדסה
ORT Braude College
המחלקה למתמטיקה
Department of Mathematics

פרויקט מסכם לתואר במדעים (BS.c)
במתמטיקה שימושית

אלגוריתמי סיווג – Perceptron & SVM

מנדי פישמן, 209022649

המחלקה למתמטיקה שימושית, אורט בראודה

כרמיאל, ישראל

מנחה: פרופסור חגי כתריאל

תאריך להגשה: 10/10/2023

תוכן עניינים

1 **מבוא** 2
1.1 סקירה 3
1.2 רקע – ניתוח מודלי סיווג 4
2 **אלגוריתם Perceptron** 5
2.1 הפרדה ליניארית של נתונים 5
2.2 הצגת המודל 5
2.3 תרגום הבעיה למרחב R^{n+1} 6
2.4 Perceptron כלל החלטה 6
2.4.1 אלגוריתם ההחלטה 6
2.4.2 הגיאומטריה שעומדת מאחורי כלל ההחלטה 6
2.4.3 הקשר שבין המכפלה הסקלרית לבין הזוויות שבין הווקטורים 7
2.5 אלגוריתם הלמידה 8
2.5.1 אלגוריתם הלמידה – אינטואיציה 8
2.6 ניסוח מתמטי פורמלי לאלגוריתם הלמידה של ה- Perceptron 10
2.7 התכנסות האלגוריתם 12
2.7.1 משפט החסם על מספר הטעויות 13
2.7.2 התכנסות האלגוריתם – הגדרות ומושגים 13
2.8 התכנסות האלגוריתם – הוכחה פורמלית 15
2.9 בחינת ה- Perceptron על דאטה פרח האיריס 15
3 **אלגוריתם SVM, מכונת ווקטורים תומכים** 18
3.1 SVM Hard Margin – הפרדה קשיחה 18
3.1.1 הגדרת בעיית האופטימיזציה 20
3.1.2 קיום ויחידות הפתרון + קמירות הבעיה 20
3.1.3 הבעיה הדואלית 25
3.1.4 הבעייתיות בהפרדה קשיחה 27
3.2 הפרדה רכה (Soft Margin) 27
3.2.1 בעיית האופטימיזציה של הפרדה רכה 27
3.2.2 קמירות הבעיה 28
3.3 פתרון בעיית SVM באמצעות שיטת Batch Gradient Descent 30
3.4 הסתייגות מאלגוריתם SVM תחת הפרדה רכה 33
3.5 שיטת הגרעין (Kernel Method) 33
3.5.1 שימוש בשיטת הגרעין 34
3.5.2 דוגמאות לשיטת הגרעין 34
3.6 תעלול הגרעין בהפרדה קשיחה (Kernel Trick) 35
3.7 בחינת SVM על דאטה מוגרל 37
4 **יישום לסיווג טקסט באמצעות אלגוריתם SVM** 42
4.1 אופן התהליך 42
4.2 תהליך טיוב הנתונים 42
4.3 ווקטוריזציה – יצירת ווקטורים מתוך טקסט 43
4.3.1 דוגמא עבור תהליך טיוב ו-ווקטוריזציה: 45
4.4 יישום של SVM עבור סיווג טקסט 47
5 **סיכום** 49
6 **הכרת תודה** 49
7 **רשימת מקורות** 50
8 **מימוש הקוד** 51

1 מבוא

קלסיפיקציה - בעיות סיווג:

בעיות סיווג הן נושא מרכזי בתחום למידת מכונה כאשר המטרה העיקרית היא ללמד מכונות איך לקטלג דאטה למחלקות בהתאם למאפיינים הנתונים בדאטה. למעשה, קלסיפיקציה היא לרוב תהליך מתחום הלמידה המונחית, כלומר מספקים למחשב דאטה הכולל דוגמאות המכילות מידע של פיצ'רים מסוימים אודות הדוגמאות ואת שיוך הדוגמאות למחלקות מסוימות. המטרה היא ליצור "הבנה" אצל המכונה כיצד המידע על פיצ'רים של דוגמא מסוימת קובע את השתייכותה למחלקה כזו או אחרת. דוגמאות רבות יש לבעיות קלסיפיקציה, אחת הבעיות המוכרות היא תיוג מיילים לספאם. כלומר, המכונה לומדת לזהות דפוסים של מיילים בעלי תוכן ספאם לעומת מיילים שאינם כאלה. כיום, קלסיפיקציה מהווה כלי חשוב בעולם ועושים בה שימושים בעיקר בנתוני עתק (Big Data) מתחומים שונים אשר אמורים לקבל בהם החלטות וביניהם – כלכלה, רפואה, מדיניות ועוד. בעוד שכדי ללמד מכונה לפתור בעיות סיווג נדרש שימוש באלגוריתמים מסובכים, לנו בני האדם נראה שאנו עושים זאת באופן טבעי ופשוט, אך מתחת לפני השטח המוח שלנו גם כן נדרש להפעיל אלגוריתמים מורכבים כדי לבצע את הפעולות הללו. במהלך חייו האדם נדרש לבעיות סיווג מבלי ששם לב לאלה, מזיהוי סוג חיה (לדוגמא – כלב או חתול) ועד לזיהוי תוכן פוגעני מתוך טקסט. המטרה באלגוריתמי הסיווג היא לאפשר למחשב "לרכוש" את יכולותיו של האדם במידה מסוימת.

אלגוריתם Perceptron:

הפרספטרון הוא אלגוריתם מתחום למידת מכונה תחת הקטגוריה של למידה מונחית של סיווג בינארי. יש אפשרות להציג את האלגוריתם כרשת נוירונים, כך שאת אלגוריתם זה ניתן גם לייחס לתחום הלמידה העמוקה. מטרת הפרספטרון הוא להחליט האם קלט מסוים המוצג בצורת ווקטור של מספרים שייך למחלקה מסוימת. הפרספטרון הוא מסווג ליניארי. כלומר, אלגוריתם זה חוזה באמצעות פונקציה ליניארית אשר משתניה הם הקלט של דגימה מסוימת לאיזה משתי המחלקות הדגימה שייכת. הפרספטרון הומצא לראשונה ב-1943 על ידי Warren McCulloch ו-Walter Pitts. המימוש הראשון היה מכונה שנבנתה ב-1958 במעבדה אווירונאוטית על ידי Frank Rosenblatt. אמנם, היישום הראשון של האלגוריתם היה בתוכנה, אך לאחר מכן יושם בחומרה. מטרת היישום בחומרה היה לזהות תמונה. התמונה כללה מערך של 400 תאי פוטו, המחוברים באופן אקראי ל-"נוירונים". משקולות קודדו בפוטנציומטרים ועדכוני המשקולות במהלך הלמידה בוצעו על ידי מנועים חשמליים. תחילה, הפרספטרון נראה מבטיח, אך הוכח במהירות שאינו מצליח בזיהוי סוגים רבים של דפוסים. דבר זה גרם לקיפאון בתחום חקר הרשתות העצביות במשך שנים רבות עד שלבסוף נוצר פרספטרון רב שכבתי. נציין כי פרספטרון חד שכבתי הוא בעל יכולת למידה של דפוסים המופרדים באופן ליניארי. כיום, נעשה שימוש באלגוריתם פרספטרון עבור משימות סיווג שונות. למשל, זיהוי הונאה בכרטיסי אשראי כפי שניתן לראות במאמר [1].

אלגוריתם SVM:

אלגוריתם Support Vector Machine – SVM, הוא שיכלול של אלגוריתם פרפסטרון. הוא הומצא במקור על ידי Vladimir N.Vapnok ו- Alexya Ya.Chevonenkis בשנת 1964. גם אלגוריתם זה מייצג מודל מתחום הלמידה המונחית אשר נועד לפתור בעיות קלסיפיקציה בינאריות. הוא נבדל מאלגוריתם הפרספטרון בכך שמטרתו היא למצוא את המסווג הליניארי תוך מיקסום השוליים של המסווג למול הדאטה הנתון. לאלגוריתם זה יש גרסאות רבות – הפרדה קשיחה, הפרדה רכה ושיטת הגרעין. אחת הגרסאות החשובות היא שיטת הגרעין המאפשרת לאלגוריתם לספק תוצאות טובות גם במקרה שבו דגימות הדאטה לא ניתנות להפרדה ליניארית. כמו כן, שימושים רבים לגרסה – "הפרדה רכה" שהרי גרסה זו מאפשרת שימוש באלגוריתם תוך הימנעות ממצב של התאמת יתר.

1.1 סקירה

בפרויקט זה, נציג את התיאוריה העומדת מאחורי אלגוריתמי הסיווג – Perceptron, SVM ונציג מימושים של אלגוריתמים אלו בשפת התכנות Python תוך הרצה על דאטה מוגרל ודאטה אמיתי כדי לאפשר מבט פרקטי על חשיבותם בפתירת בעיות קלסיפיקציה. כחלק מפרק המבוא נספק לקורא רקע בבעיות סיווג ובבחינת מודלי סיווג [2].

פרק 2 - יעסוק באלגוריתם Perceptron בו נציג את המודל של האלגוריתם [3, 4], חלוקת האלגוריתם לכלל החלטה ולתהליך הלמידה, האינטואיציה המתמטית העומדת מאחורי אופן הפעולה של תהליך הלמידה ושל כלל ההחלטה [5], הוכחת התכנסות האלגוריתם בצורה מתמטית ופורמלית [6, 7], ולבסוף נבחן את האלגוריתם על מערך הנתונים המוכר על פרח האיריס.

פרק 3 - יעסוק באלגוריתם SVM וגרסאותיו השונות [8] – "הפרדה קשיחה" (hard margin), "הפרדה רכה" (soft margin) ושיטת הגרעין (kernel method). תחילת הפרק יעסוק בהצגת בעיית האופטימיזציה עבור הפרדה קשיחה וננתח את הקיום והיחידות של פתרון הבעיה [9, 10], לאחר מכן, נציג את הבעייתיות בגרסה זו של האלגוריתם וננסה להתגבר על הבעייתיות באמצעות גרסת ההפרדה הרכה. נציג מימוש לפתרון בעיית ההפרדה הרכה באמצעות שיטת ה- Batch Gradient Descent [11]. לטובת שימוש בשיטה זו, נוכיח בפרק זה גם את קמירות בעיית ההפרדה הרכה [9, 10]. לבסוף, נציג את שיטת הגרעין העוסקת במציאת מסווגים עבור דאטה שניתן להפרדה בצורה שונה מזו הליניארית, על ידי העתקת המרחב למרחב חדש שבו יינתן פתרון לינארי. לבסוף, נציג הרצה של האלגוריתם על דאטה מוגרל בעל שתי מחלקות אשר ניתנות להפרדה על ידי מעגל. נציין, כי במימוש נתמקד בשיטת הגרעין אשר שונה במעט מתעלול הגרעין (שעל האחרון נפרט בקצרה) וזאת מאחר שמטרתנו בפרויקט היא למצוא מסווג עבור בעיות סיווג שונות ולהציג את מהלך ההעתקה של הבעיה למרחב החדש. לעומת זאת, עיקרו של תעלול הגרעין הוא בשיפור זמן הריצה של התוכנית.

פרק 4 - יעסוק ביישום אלגוריתם SVM לטובת סיווג טקסט [12], נפרט על תהליך טיוב הנתונים טרם הרצת האלגוריתם. תוצג התיאוריה העומדת מאחורי הפיכת הטקסט לווקטור בעל רכיבים ממשיים על ידי השיטה tf-idf [13] כדי שנוכל להפעיל את האלגוריתם. נממש יישום זה על מאגר מידע של ציוצים מהרשת החברתית Twitter וננסה באמצעות SVM ללמד את המחשב להבחין בין ציוץ בעל אופי גזעני לבין ציוץ שאינו בעל אופי גזעני. נשווה את תוצאות התוכנית שלנו למול תוצאות הפונקציה המובנית של האלגוריתם SVM מתוך ספריית sklrean בשפת התכנות python.

1.2 רקע – ניתוח מודלי סיווג

כפי שצוין קודם, האלגוריתמים שאותם נבחן בפרויקט זה הם מתחום למידת מכונה. יש צורך לבחון את נכונותם של האלגוריתמים במהלך תהליך הלמידה ולאחריו. על כן, נתייחס לתהליך הלמידה ולאחר מכן נבחן את פתרון הבעיה שהאלגוריתם סיפק. לשם כך, בטרם הרצת האלגוריתמים נחלק את מאגר הנתונים לשני מאגרים – מאגר אימון ומאגר בחינה. תהליך הלמידה של כל אלגוריתם יבוצע על מאגר האימון וכדי לבחון את השיפור במהלך הלמידה נשרטט (בחלק מהאלגוריתמים) את פונקציית המטרה ואת אחוז הדיוק (אחוז הסיווגים הנכונים שהאלגוריתם ביצע מתוך סך דגימות מאגר האימון) כפונקציה של מספר האיטרציות, ונחקור את תכונות הגרפים שהתקבלו. לדוגמא, כאשר נשתמש בשיטת ה-Gradient Descent הידועה לשם מציאת מינימום עבור פונקציית מטרה, נצפה לקבל מגמה כללית של ירידת ערך פונקציית המטרה ככל שנתקדם במספר האיטרציות. לאחר סיום תהליך הלמידה נרצה לבחון את הפתרון שהאלגוריתם סיפק, על דאטה שלא היה מעורב בתהליך הלמידה (דאטה לא "מזוהם"). לשם כך, נשתמש בפתרון שקיבלנו ונריץ תחת פתרון זה את מאגר הבחינה. הפתרון עתיד לתת לנו סיווג עבור כל דגימה ממאגר הבחינה (סיווג מנובא) ונשווה סיווג זה למול הסיווג המקורי של אותה דגימה. נבנה מטריצת ערפול (Confusion Matrix) וממנה נוציא מדדים להערכת נכונות המודל [1].

מטריצת ערפול:

Predicted\Actual	Positive Class	Negative Class	sum
Positive Class	TP	FP	TP+FP
Negative Class	FN	TN	FN+TN
sum	TP+FN	FP+TN	TP+FP+FN+TN

- TP(True Positive) – כמות הדגימות שסווגו כחיוביות ואכן סיווגם האמיתי הוא חיובי.
- FP(False Positive) – כמות הדגימות שסווגו כחיוביות, אך סיווגם האמיתי הוא שלילי.
- FN(False Negative) – כמות הדגימות שסווגו כשליליות, אך סיווגם האמיתי הוא חיובי.
- TN(True Negative) – כמות הדגימות שסווגו כשליליות ואכן סיווגם האמיתי הוא שלילי.

ישנם מדדים רבים שניתן לחשב מתוך מטריצת הערפול כדי לבדוק את נכונות הפתרון שהאלגוריתם סיפק (כגון – TPR, Precision, Recall, לפירוט נוסף יש לעיין במאמר [1]). אנו נתמקד במדד Accuracy ולכן נפרט עליו:

Accuracy – אחוז הדיוק, חישוב המייצג את האחוז בו סיווגנו נכון (הסיווג המנובא תואם לסיווג האמיתי) למול כלל הדגימות שאותם אנחנו בוחנים. הנוסחא לחישוב אחוז הדיוק:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

(*) הערה: באלגוריתם SVM, נבחן את ערך ה- Accuracy גם במהלך הלמידה ביחס למאגר האימון בכל איטרציה על מנת לשרטט גרף של אחוז הדיוק כפונקציה של מספר האיטרציה וזאת על מנת לראות את מגמת השיפור במהלך הלמידה.

2 אלגוריתם Perceptron

2.1 הפרדה ליניארית של נתונים

כפי שציינו במבוא, אלגוריתם ה-Perceptron מתמקד בבעיות סיווג אשר האובייקטים הנכללים במאגר המידע ניתנים להפרדה ליניארית המפרידה בין המחלקות. נסביר את המושג אובייקטים הניתנים להפרדה ליניארית. תחילה, נסביר מושג זה במישור R^2 ולאחר מכן נרחיב את המושג למימדים אחרים. נניח שקיימים אובייקטים המסווגים בצורה בינארית כלומר 1- ו-1. נרצה לבחון את האובייקטים באמצעות שני מאפיינים ולכן נוכל לתאר את הדגימות במערכת צירים XY כלומר במישור האוקילידי $R^2 = \{(x_1, x_2)|x_1, x_2 \in R\}$. המושג הפרדה ליניארית מתאר מצב שבו ניתן למצוא ישר $w_1x_1 + w_2x_2 + bias = 0$ כך ש- $w_1, w_2, bias \in R$

(ניתן לרשום גם בצורה $x_2 = ax_1 + b$) אשר מייצר הפרדה בין האובייקטים בעלי תווית סיווג 1 לבין האובייקטים בעלי תווית סיווג 1-. במילים אחרות, כל הנקודות שנמצאות מעל הישר או עליו (כלומר, מקיימות $w_1x_1 + w_2x_2 + bias \geq 0$) יהיו נקודות השייכות לקבוצה עם התווית *1* ואילו כל הנקודות מתחת לישר (כלומר, מקיימות $w_1x_1 + w_2x_2 + bias < 0$) יהיו הנקודות השייכות לקבוצה עם התווית *1-*. במרחב האוקלידי התלת מימדי - $R^3 = \{(x_1, x_2, x_3)|x_1, x_2, x_3 \in R\}$ אובייקטים הניתנים להפרדה בצורה ליניארית יופרדו על ידי מישור $w_1x_1 + w_2x_2 + w_3x_3 + bias = 0$ כך ש- $w_1, w_2, w_3, bias \in R$.

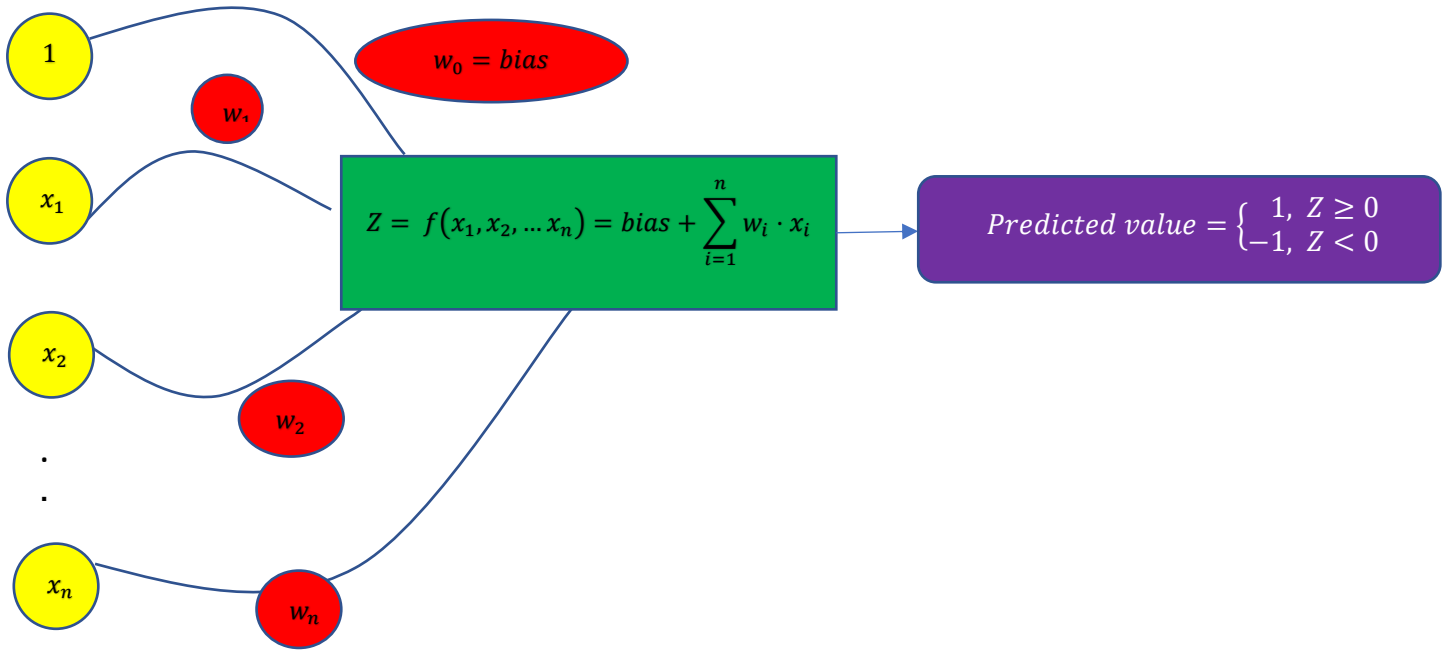
במרחב R^n ההפרדה מתבצעת על ידי *hyper-plane* – היפר מישור, באופן דומה.

2.2 הצגת המודל

<u>המטרה</u>: סיווג בינארי בין שני אובייקטים, 1- או 1.

<u>הנחות המודל</u>: האובייקטים שאותם אנחנו רוצים לבחון ניתנים להפרדה בצורה ליניארית באמצעות הפרמטרים עליהם אנחנו מבססים את מבחן ההחלטה.

<u>מידול ויזואלי</u>:



2.3 תרגום הבעיה למרחב R^{n+1}

לשם הנוחות, נסמן -

$$X = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{pmatrix}, \ W = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix}$$

ניתן להסתכל על הבעיה הנתונה כבעיה במרחב R^{n+1}, כאשר כל נקודות הדאטה נמצאות על העל מישור $x_0 = 1$ והיפר המישור המפריד עובר דרך ראשית הצירים. עבור הבעיה המתורגמת משוואת ההיפר מישור תהיה - $\sum_{i=0}^{n} w_i \cdot x_i = 0$ או באופן פשוט - $\mathrm{W}^{\mathrm{T}} X = 0$.

2.4 כלל החלטה Perceptron

נבצע הפרדה בין אלגוריתם ה-Perceptron לבין אלגוריתם הלמידה שלו.
תחילה נסביר את אלגוריתם ההחלטה של ה-Perceptron שלמעשה מייצג את כלל ההחלטה עבור סיווג דגימה מסוימת. כרגע, נצא מנקודת הנחה שמצאנו את היפר המישור המפריד וכל שנותר לנו להחליט עבור דגימה מסוימת לאיזה מן המחלקות היא שייכת וזאת נבצע על פי כלל ההחלטה.

2.4.1 אלגוריתם ההחלטה

$W_t \leftarrow \ weight\ vector\ we\ found$

$Pick\ random\ X_i\ data\ point$

$if\ W_t^T X_i \geq 0\ then$:

$\quad \hat{y} = 1;\ \ \#predicted\ value\ is\ positive(P)$

$else$:

$\quad \hat{y} = -1;\ \ \#predicted\ value\ is\ Negative(N)$

end

(*) נניח ללא הגבלת הכלליות ש- P זה קבוצת האובייקטים בעלי תווית 1, ו- N קבוצת האובייקטים בעלי תווית (1-). (מהמילים Positive/Negative)

2.4.2 הגיאומטריה שעומדת מאחורי כלל ההחלטה

הסבר זה מתייחס לבעיה המתורגמת במישור ה-R^{n+1}. עבור כל דגימה נרצה לחזות לאיזה קבוצת אובייקטים היא שייכת (כלומר מה התווית של הדגימה) ונעשה זאת על פי המבחן הבא : אם הזווית בין ווקטור המאפיינים X של הדגימה לבין ווקטור המשקולות W, מעל 90 מעלות אזי מבחן ההחלטה יקבע שהתווית היא "1-" כלומר שלילי ולהיפך אם הזווית בין ווקטור המאפיינים לבין ווקטור המשקולות הזווית תהיה שווה או קטנה מ- 90 מעלות אז האלגוריתם יקבע שהתווית היא "1" כלומר דגימה חיובית . בצורה זאת, אנחנו יוצרים מצב בו ההיפר-מישור המאונך לווקטור המשקולות הוא ההיפר-מישור אשר מפריד בין הדגימות.

2.4.3 הקשר שבין המכפלה הסקלרית לבין הזוויות שבין הווקטורים

באלגוריתם של כלל ההחלטה לעיל, ניתן לראות כי מבחן ההחלטה מבוצע על פי המכפלה הסקלרית ואולם בהסבר הגיאומטרי התייחסנו לזוויות שבין הווקטורים. אם כך, נידרש להסביר את הקשר ביניהם – למעשה, הביטוי $W_t^T X_i$ מייצג את המכפלה הסקלרית $< W_t, X_i >$. נראה כי קיים יחס ישר בין המכפלה הסקלרית לבין הזווית בין הווקטורים X_i ו W_t:

$$\cos(\theta) = \frac{< W_t, X_i >}{||W_t|| \cdot ||X_i||} \underbrace{\Rightarrow}_{||W_t||,||X_i|| \geq 0}$$

$$\Rightarrow sign(\cos(\theta)) = sign\left(\frac{<W_t, X_i>}{||W_t|| \cdot ||X_i||}\right) = sign(< W_t, X_i >)$$

(*) הערה: הפונקציה $\cos(\theta)$ המוגדרת בקטע $[0, \pi]$ כאשר θ נמדדת ברדיאנים, מקבלת ערכים אי שליליים בקטע $\left[0, \frac{\pi}{2}\right]$ וערכים שליליים בקטע $\left[\frac{\pi}{2}, \pi\right]$.

<u>הסבר גיאומטרי נוסף העומד מאחורי כלל ההחלטה:</u>

ניתן פרספקטיבה גיאומטרית במרחב ה- R^n על אותו מבחן החלטה.

לשם כך, נבצע סימונים שונים במעט -

נגדיר פונקציה $f(x_1, x_2, \dots, x_n) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + bias$ אשר משטח הרמה $f = 0$ מייצג היפר-מישור במרחב R^n.

כל הנקודות הנמצאות על ההיפר מישור מקיימות $f(x_1, x_2, \dots, x_n) = 0$. וכל הנקודות שנמצאות "מעל" ההיפר-מישור או עליו מקיימות:

$$f(x_1, x_2, \dots, x_n) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + bias \geq 0$$

ואילו כל הנקודות הנמצאות "מתחת" להיפר-מישור, מקיימות:

$$f(x_1, x_2, \dots, x_n) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + bias < 0$$

מאחר שמשטח הרמה $f = 0$ מגדיר היפר מישור נרצה שההיפר מישור הזה יהיה ההיפר המישור המפריד בין האובייקטים. כלומר, כל הנקודות ש"מעל" ההיפר-מישור יהיו מסווגות לקבוצת האובייקטים בעלי התווית *1*, ואילו הנקודות ש"מתחת" להיפר-מישור יהיו מסווגות לקבוצת האובייקטים בעלי תווית *-1*.

2.5 אלגוריתם הלמידה

```
P ← inputs with label (y_i = 1)
N ← inputs with label (y_i = −1)
Initailize W randomly (zero vector)
While ! convergence do
        Pick random X ∈ P ∪ N;
        if X ∈ P and W^T X < 0 then
                W = W + X
        end
        if X ∈ N and W^T X ≥ 0 then
                W = W − X
        end
end
```

(*) נניח ללא הגבלת הכלליות ש- P זה קבוצת האובייקטים בעלי תווית 1, ו- N קבוצת האובייקטים בעלי תווית (1-). (מהמילים positive/negative)

<u>טבלת החלטה:</u>

Label	y_i $(real\ value)$	$W^T X$	$\hat{y}$ **(predicted value)**	$y_i = \hat{y}$	$y_i(W^T X)$	**Update**
P	1	≥ 0	1	True	+ (or zero)	No
P	1	< 0	-1	False	-	Yes
N	-1	≥ 0	1	False	- (or zero)	Yes
N	-1	< 0	-1	True	+	No

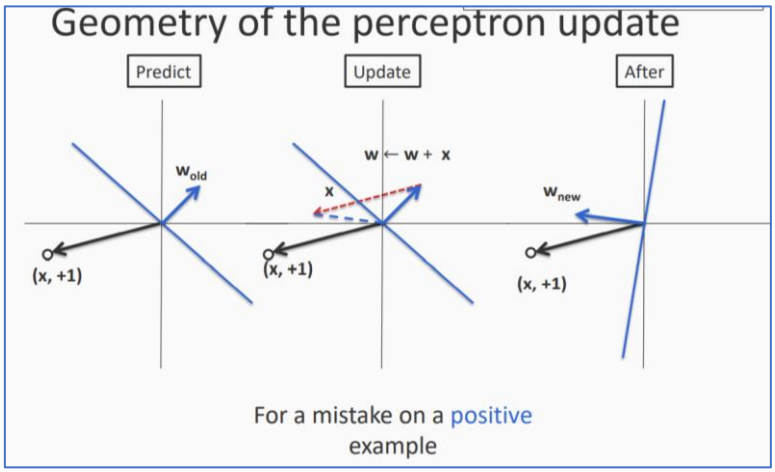
2.5.1 אלגוריתם הלמידה – אינטואיציה

כל דגימה שנכניס תעבור את מבחן ההחלטה ולפי תוצאתו ייקבע ערך החיזוי של הדגימה (כלומר אם הדגימה חיובית או שלילית). לאחר מכן, נבדוק האם מבחן ההחלטה תואם את מבחן המציאות. אם קיים פער בין ערך החיזוי לבין התווית האמיתית של הדגימה (כלומר טעינו בחיזוי) אז נבצע תיקון לווקטור המשקולות כך שהסיכוי לטעות ביחס לאותה דגימה שבה בוצעה הטעות יהיה קטן יותר ובתיאור גיאומטרי - נייצר ווקטור משקולות חדש אשר הזווית בינו לבין ווקטור המאפיינים של אותה דגימה שבה טעינו, תהיה קטנה יותר במקרה של טעות בדגימה חיובית ותהיה גדולה יותר במקרה של טעות בדגימה שלילית. בהמשך נראה שהווקטור החדש אכן מייצר

זווית קטנה/גדולה יותר ביחס לווקטור המאפיינים של אותה דגימה (כלומר הסיכוי שנטעה ביחס לאותה דגימה יהיה קטן יותר).

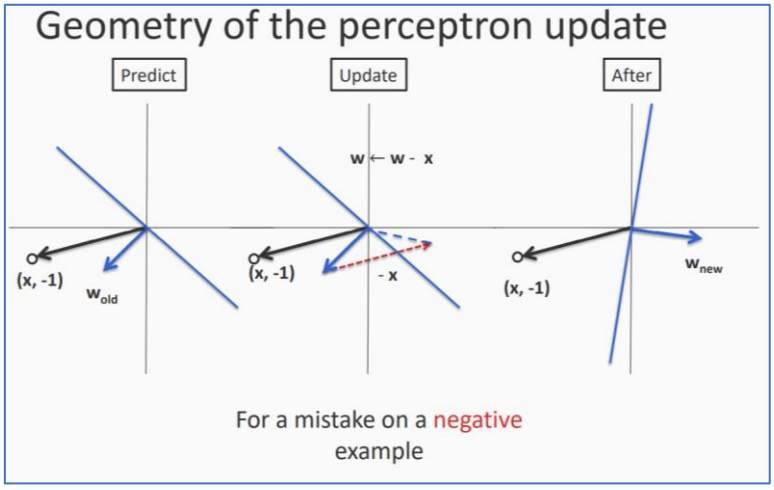
<u>סקיצה הממחישה את ההסבר הגיאומטרי:</u>

עבור טעות בדגימה חיובית:



The University of UTAH. (Spring 2018). The Perceptron Algorithm, Machine learning (p.25).

עבור טעות בדגימה שלילית:



The University of UTAH. (Spring 2018). The Perceptron Algorithm, Machine learning (p.31).

<u>דרך נוספת להסתכלות גיאומטרית:</u>

בפרק של כלל ההחלטה הראינו כי ישנן שתי דרכי הסתכלות גיאומטריות באשר לכלל ההחלטה. כעת, נראה שניתן להסתכל על אלגוריתם הלמידה גם מהמבט של הגדרת היפר מישור (במישור R^n) עבור הבעיה המקורית.

נסמן את המכפלה הסקלרית כפונקציה:

$$f(X) \underbrace{=}_{x_0=1} f(1, x_1, x_2, \dots, x_n) = W^t X = bias + \sum_{i=1}^{n} w_i x_i$$

הפונקציה הנ״ל היא פונקציה ליניארית שהנקודות בהן היא מתאפסת מגדירות היפר-מישור במרחב R^n אשר מתיימר להפריד בין הדגימות.

עבור כל דגימה (X_i, y_i) נבדוק אם הצבת ווקטור המאפיינים שלה $\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ בפונקציה תקנה לנו ערך חיובי או שלילי. ערך חיובי משמעו שהנקודה נמצאת מעל או על ההיפר מישור המוגדר כרגע, כלומר $W^t X \geq 0$. ואילו ערך שלילי משמעו שהנקודה נמצאת מתחת להיפר-מישור, כלומר מתקיים $W^t X < 0$, ולפיכך נסווג את הנקודות. במקרה של טעות נרצה לתקן את ווקטור המשקולות ורכיב ה- *bias* כך שהצבת ווקטור המאפיינים בפונקציה החדשה יגדיל את הסיכוי שנקודת הדגימה בה טעינו תהיה בצד הנכון של הסיווג. בשינוי ווקטור המשקולות תיווצר פונקציה חדשה כך שעבור טעות בדגימה חיובית נרצה שבהצבת ווקטור המאפיינים של אותה דגימה ערך הפונקציה יגדל ובכך נגדיל את סיכוייה להיות מעל ההיפר-מישור ובטעות בדגימה שלילית נרצה שבהצבת ווקטור המאפיינים בפונקציה החדשה ערך הפונקציה יהיה קטן יותר ובכך נגדיל את סיכוי הדגימה להיות מתחת להיפר-מישור.

2.6 ניסוח מתמטי פורמלי לאלגוריתם הלמידה של ה- Perceptron

יש לחזור על הפעולות קלט ועיבוד עד להתכנסות האלגוריתם (בהמשך נסביר על משמעות ההתכנסות):

<u>קלט:</u>

תהיינה רצף של דגימות - $(X_1, y_1), (X_2, y_2), \ldots, (X_m, y_m)$ כך ש- $X_i = \begin{pmatrix} x_{i0} \\ \vdots \\ x_{in} \end{pmatrix} \in R^{n+1}$

כך ש- $x_{i0} = 1$ ו- $y_i \in \{-1,1\}$, לכל $i = 1,2,3,\ldots,m$. t מסמן את מספר הטעויות שעשה אלגוריתם ה- Perceptron במהלך הלמידה.

<u>עיבוד:</u>

1. נאתחל את ווקטור המשקולות W_0 לווקטור האפס:

$$W_0 = \vec{0} \in R^{n+1}$$

2. עבור כל דגימה (X_i, y_i):

$$\hat{y}_i = sign(W_t^T X_i)$$

$$\#\hat{y}_i \text{ is the predicted value (N or P)}$$

3. אם מתקיים: $\hat{y}_i \neq y_i$ (זה יקרה כאשר $y_i \cdot (W_t^T X_i) \leq 0$):

עדכן:

$$W_{t+1} = W_t + (y_i \cdot X_i)$$

4. אחרת, כלומר $\widehat{y_i} = y_i$:

לא נקדם את t שהרי לא הייתה טעות, כלומר ווקטור המשקולות יישאר כפי שהוא.

5. תנאי עצירה – כאשר כל הדגימות מסווגות נכון:

$$y_i \cdot (W_t^T X_i) > 0 \quad , \forall i \in \{1, \dots, m\}$$

<u>פלט:</u>

הפרדה ליניארית בין הדגימות לפי סיווג האובייקטים.

<u>ביסוס מתמטי עבור האינטואיציה שעומדת מאחורי אלגוריתם הלמידה</u>

(*) הערה: נציין כי ביסוס זה עבור האינטואיציה אינה מהווה הוכחה להתכנסות האלגוריתם. נבחן את האינטואיציה ממבט של זוויות בין ווקטורים כלומר עבור הבעיה המוגדרת במישור R^{n+1}.

תחילה נסביר כי יש יחס הפוך בין זוויות בין ווקטורים לבין מכפלה סקלרית:

$$\cos(\theta) = \frac{< W_t, X_i >}{||W_t|| \cdot ||X_i||} \Rightarrow \theta = arcos\left(\frac{< W_t, X_i >}{||W_t|| \cdot ||X_i||}\right)$$

(*) תזכורת: $< W_t, X_i >= W_t^T X_i$.

פונקציית $y = arcos(x)$ היא פונקציה יורדת בקטע *[-1,1]* ולכן כאשר הביטוי $\frac{<W_t,X_i>}{||W_t||\cdot||X_i||}$ גדל כך הזווית קטנה ולהיפך. נצא מנקודת הנחה כי לכל $t \geq 0$ מתקיים $||W_t|| = 1$, כלומר לאחר כל איטרציה יבוצע נרמול לווקטור המשקולות. כמו גם, בשינוי ווקטור המשקולות אין אנו משנים את ווקטור המאפיינים של הדגימה בה טעינו ולכן $||X_i||$ נשאר כפי שהוא.

נניח שביצענו טעות בדגימה חיובית (כלומר הניבוי היה שלילי), כלומר יתקיים:

$$y_i = +1 \quad and \quad W_t^T X_i < 0$$

נבצע עדכון לווקטור המשקולות לפי האלגוריתם:

$$W_{t+1} = W_t + X_i$$

*נזכיר שבמקרה של טעות עבור דגימה חיובית נרצה שהזווית בין ווקטור המשקולות החדש לבין ווקטור המאפיינים של הדגימה תהיה קטנה יותר. ובכן נראה שזה אכן מתקיים:

$$(\#)\ W_{t+1}^T X_i = (W_t + X_i)^t X_i = W_t^T X_i + (X_i)^T X_i = W_t^T X_i + \left(||X_i||\right)^2 \underbrace{\geq}_{||X_i||_2 \geq 0} W_t^T X_i$$

לכן,

$$\theta_{new} = arcos\left(\frac{W_{t+1}^T X_i}{||W_{t+1}|| \cdot ||X_i||}\right) \underbrace{\leq}_{(\#)} arcos\left(\frac{W_t^T X_i}{||W_t|| \cdot ||X_i||}\right) = \theta_{old}$$

כעת, נראה מקרה בו האלגוריתם ביצע טעות בדגימה שלילית (כלומר הניבוי היה חיובי), ומבחינה מתמטית:

$$y_i = -1 \quad and \quad W_t^T X_i \geq 0$$

נבצע עדכון לווקטור המשקולות לפי האלגוריתם:

$$W_{t+1} = W_t - X_i$$

*נזכיר שבמקרה של טעות עבור דגימה שלילית נרצה שהזווית בין ווקטור המשקולות החדש לבין ווקטור המאפיינים של הדגימה תהיה גדולה יותר. ובכן נראה שזה אכן מתקיים:

$$(\#\#)\ W_{t+1}^T X_i = (W_t - X_i)^T X_i = W_t^T X_i - (X_i)^T X_i = W_t^T X_i - (||X_i||)^2 \underset{||X_i||_2 \geq 0}{\leq} W_t^T X_i$$

לכן,

$$\theta_{new} = arcos\left(\frac{W_{t+1}^T X_i}{||W_{t+1}|| \cdot ||X_i||}\right) \underset{(\#\#)}{\geq} arcos\left(\frac{W_t^T X_i}{||W_t|| \cdot ||X_i||}\right) = \theta_{old}$$

2.7 התכנסות האלגוריתם

<u>משמעות התכנסות האלגוריתם</u> – במספר סופי של איטרציות נמצא ווקטור W המייצר הפרדה ליניארית בין האובייקטים.

<u>משפט</u>: אם הנתונים ניתנים להפרדה בצורה ליניארית אזי האלגוריתם יתכנס.

<u>הגדרות חשובות</u>:

<u>היפר מישור</u>: $H_w = \{X:\ W^T X = 0\}$.

<u>קבוצת מאגר הנתונים</u>: $D = \{(X_i, y_i) | X_i \in R^{n+1}, y_i \in \{-1,1\}\}$ נסמן - $|D| = m$.

<u>מרחב הדגימות</u>: יהי R מספר ממשי כך שמתקיים $||X_i|| \leq R$ לכל $1 \leq i \leq m$, אזי מרחב הדגימות הוא $S = \{X:\ ||X|| \leq R\}$. נוסיף ונאמר כי $D \subseteq S \subseteq R^{n+1}$ כאשר אנו מריצים את האלגוריתם בבחינת n מאפיינים עבור כל דגימה.

<u>שוליים</u>: המרחק המינימלי בין נקודות מהדאטה להיפר המישור.

יהי H_w היפר-מישור המפריד בין הדגימות, אזי השול יוגדר באופן הבא:

$$\gamma(S, w) = min\ \ distance_{1 \leq i \leq m}(X_i, H_w)$$

כאשר

$$distance(X, H_w) = \min\{||X - X'||: X' \in H_w\} = \frac{| < X, W > |}{||W||} = \frac{|W^T X|}{||W||}$$

<u>קבוצת העל מישורים המפרידים:</u>

$$E = \{W | y_i W^t X_i > 0 \ \forall (X_i, y_i) \in D, ||W|| = 1\}$$

<u>השוליים של מאגר המידע:</u> השוליים האפשריים המקסימליים מכל היפר מישור שמפריד/מסווג את הדאטה בצורה ליניארית.

$$\gamma = \gamma(D) = max_{W \in E} \ \gamma(D, W)$$

2.7.1 משפט החסם על מספר הטעויות

תהיינה דגימות נתונים - $(X_1, y_1), (X_2, y_2), \ldots, (X_m, y_m)$ כך ש- $X_i \in R^{n+1}$ ו- $y_i \in \{-1,1\}$,
ומתקיים: $||X_i|| \leq R$ לכל $i = 1,2,3, \ldots, m$. נניח שקיים ווקטור יחידה $u \in R^{n+1}$
($||u|| = 1$), כך שעבור איזשהו $\gamma \in R$, $\gamma > 0$
מתקיים $y_i(u^T X_i) \geq \gamma$ לכל $1 \leq i \leq m$. אז אלגוריתם ה-Perceptron יעשה לכל היותר $\left(\frac{R}{\gamma}\right)^2$ טעויות במהלך "האימון" (אימון הלמידה).
(*) נציין כי טעות תוגדר במקרה שבו האלגוריתם ניבא ערך שונה מהערך האמיתי של הדגימה.

כלומר:
1. קיים מאגר דאטה מסווג בינארית עם m דגימות קלט.
2. האובייקטים במאגר ניתנים להפרדה ליניארית.
אז:
האלגוריתם ימצא את ההיפר-מישור המפריד תוך מספר סופי של צעדים. (כלומר האלגוריתם יתכנס בזמן סופי)

2.7.2 התכנסות האלגוריתם – הגדרות ומושגים

כפי שצוין לעיל, נגדיר תחילה את ווקטור המשקולות להיות ווקטור האפס:

$$W_0 = \vec{0}$$

נזכיר שכל הנקודות מוכללות ב"כדור" שרדיוסו R, כלומר מתקיים:

$$||X_i|| \leq R$$

לכל $1 \leq i \leq m$.

האובייקטים/הדגימות ניתנות להפרדה ליניארית עם שוליים ששווים ל-γ , כלומר קיים ווקטור u שמפריד את הנתונים כך ש- $||u|| = 1$ ומתקיים:

$$distance(X_i, H_u) = \frac{|u^T X|}{||u||} \underbrace{=}_{||u||=1} y_i(u^T X_i) \geq \gamma$$

לכל $1 \le i \le m$.

(*) הערה: להוכחת המשפט נצטרך להשתמש בשתי טענות ולכן תחילה ננסח ונוכיח את הטענות.

2.7.2.1 טענה 1

אחרי t טעויות מתקיים:

$$u^T W_t \ge t\gamma$$

2.7.2.1.1 הוכחת טענה 1 (באמצעות אינדוקציה):

בסיס האינדוקציה (נניח t=0):

$$u^T W_0 \underbrace{=}_{W_0=\vec{0}} 0 = 0 \cdot \gamma \ge 0 \cdot \gamma \Rightarrow u^T W_0 \ge 0 \cdot \gamma$$

הנחת האינדוקציה – נניח שהטענה נכונה עבור t=k-1:

$$u^T W_{k-1} \ge (k-1)\gamma$$

הוכחת האינדוקציה – נוכיח עבור t=k:

$$u^T W_k = u^T\left(W_{k-1} + y_j X_j\right) = \underbrace{u^T W_{k-1}}_{\ge (k-1)\gamma} + \underbrace{y_j u^T X_j}_{\ge \gamma} \ge (k-1)\gamma + \gamma = k \cdot \gamma \Rightarrow$$

$$u^T W_k \ge k\gamma$$

2.7.2.2 טענה 2

אחרי t טעויות מתקיים:

$$\left|\left|W_t\right|\right|^2 \le t \cdot R^2$$

2.7.2.2.1 הוכחת טענה 2 (באמצעות אינדוקציה)

בסיס האינדוקציה (נניח t=0):

$$\left|\left|W_0\right|\right|^2 = \left|\left|\vec{0}\right|\right|^2 \le 0 \cdot R^2 \Rightarrow \left|\left|W_0\right|\right|^2 \le 0 \cdot R^2$$

הנחת האינדוקציה - נניח שהטענה נכונה עבור t=k-1:

$$\left|\left|W_{k-1}\right|\right|^2 \le (k-1) \cdot R^2$$

הוכחת האינדוקציה – נוכיח עבור t=k:

$$\left|\left|W_k\right|\right|^2 = \left|\left|W_{k-1} + y_i X_i\right|\right|^2 = \underbrace{\left|\left|W_{k-1}\right|\right|^2}_{\le (k-1)\cdot R^2} + \underbrace{2y_i(W_{k-1}X_i)}_{\substack{\le 0 \\ (2.7.2.2.1a)\text{Explanation below}}} + \underbrace{\left|\left|X_i\right|\right|^2}_{\left|\left|X_i\right|\right| \le R}$$

$$\le (k-1) \cdot R^2 + R^2 = k \cdot R^2$$

$(2.7.2.2.1a)$ מקרה 1 – טעינו בדגימה חיובית אזי מתקיים $y_i = 1$ וגם $W_{k-1}X_i < 0$. מקרה 2 – טעינו בדגימה שלילית אזי מתקיים $y_i = -1$ וגם $W_{k-1}X_i \geq 0$. בשני המקרים מתקיים $y_i(W_{k-1}X_i) \leq 0$, ולכן מתקיים - $2y_i(W_{k-1}X_i) \leq 0$.

2.8 התכנסות האלגוריתם – הוכחה פורמלית

$$||W_t||^2 \underbrace{\leq}_{\text{טענה 2}} t \cdot R^2 \Rightarrow ||W_t|| \leq R\sqrt{t} \Rightarrow$$

$$\Rightarrow R\sqrt{t} \geq ||W_t|| \underbrace{\geq}_{2.8a} u^T W_t \underbrace{\geq}_{\text{טענה 1}} t\gamma \Rightarrow$$

מכך קיבלנו:

$$\Rightarrow R\sqrt{t} \geq t\gamma \Rightarrow \sqrt{t} \leq \frac{R}{\gamma} \Rightarrow$$

$$\Rightarrow t \leq \left(\frac{R}{\gamma}\right)^2$$

ובמילים אחרות, האלגוריתם יעשה מספר סופי של טעויות. כלומר, האלגוריתם יתכנס בזמן סופי.

(2.8a) אי שוויון קושי שוורץ:

$$u^T W_t = \underbrace{||u||}_{1} \cdot ||W_t|| \cdot \underbrace{\cos(a)}_{\leq 1} \leq ||W_t||$$

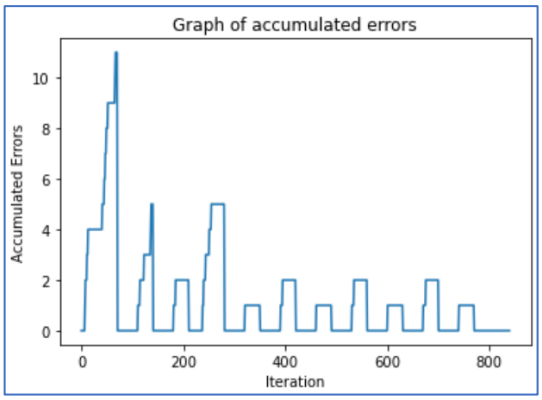
2.9 בחינת ה- Perceptron על דאטה פרח האיריס

<u>דאטה האיריס</u> – מערך נתונים המכיל מידע על 3 סוגים של פרח האיריס. הזנים הם virginica, setosa, versicolor כאשר הפיצ'רים הנתונים הם האורך והרוחב הנתונים בסנטימטרים של עלי הכותרת ועלי הגביע של כל פרח. סך כל מערך הנתונים הוא 150 פרחים – 50 מכל סוג, מערך הנתונים נלקח מאתר Kaggle, הקישור מצורף ברשימת המקורות. בפרויקט זה מימשתי את אלגוריתם הפרספטרון כפי שניתן לראות בפרק - מימוש הקוד, והרצתי את האלגוריתם על מאגר הנתונים של פרח האיריס. הרצת האלגוריתם בוצעה עבור הזנים setosa ו- virginica, כאשר הפיצ'רים הנבחרים בהרצה הראשונה – רוחב של עלי הכותרת ורוחב של עלי הגביע ובהרצה השנייה - רוחב עלה הגביע ואורך עלה הכותרת. כלומר, בחנו מאגר המכיל 100 פרחים – 50 מכל סוג. מתוך 100 הפרחים בוצעה חלוקה למאגר אימון ומאגר בחינה ביחס של 3:7 בהתאמה (מאגר אימון כולל 70 פרחים). בתוצאות ההרצה ניתן לראות גרף המכיל את נקודות מאגר האימון ואת המישור המפריד שנמצא על ידי האלגוריתם. בנוסף לכך, גרף המציג את מספר "הטעויות" המצטבר שבוצעו בתהליך הלמידה כאשר בכל מעבר מחודש על כל המאגר איפסנו מספר זה ובעצם התכנסות האלגוריתם נקבעת כאשר בוצע מעבר מחודש על כלל מאגר האימון ללא טעויות כלל (כלומר עד שהאלגוריתם יסווג נכון את כלל דגימות האימון). כמו כן, ניתן לראות את תוצאות האלגוריתם עבור מאגר הבחינה.

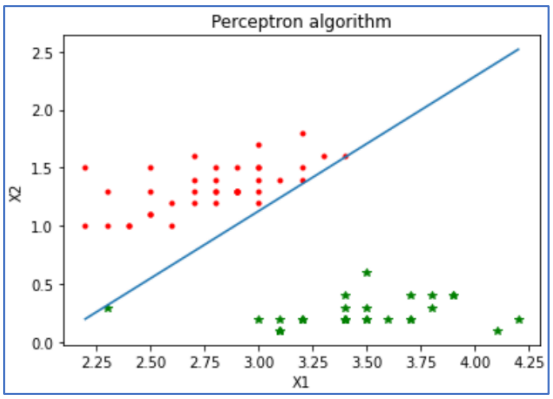
תוצאות ההרצה:

הרצה לפי המשתנים - רוחב עלי כותרת (ציר X2) ורוחב עלי גבעול (ציר X1):

גרף של מספר הטעויות המצטבר (בתהליך הלמידה):



גרף מאגר אימון והמישור המפריד:

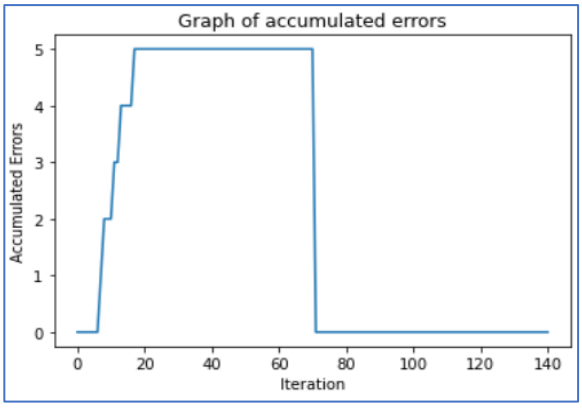


תוצאות המודל עבור מאגר הבחינה:

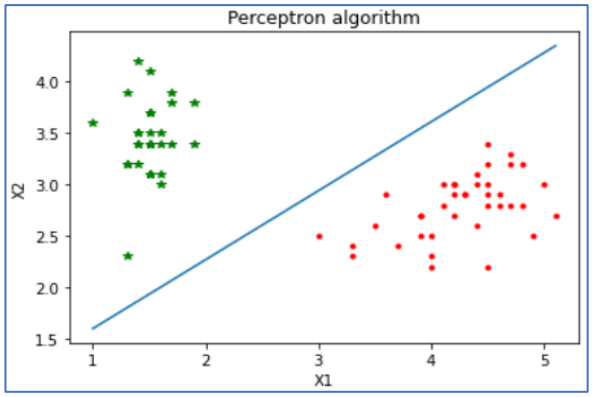
```
The boundary decision line is: y=1.16x-2.35
The weight vector is: [-16.27818724  14.04285008]
The bias balue is:  33

the confusion matrix is:
[[11  0 11]
 [ 0 19 19]
 [11 19 30]]
the accuracy rate is:
100.000%
```

הרצה לפי המשתנים - רוחב עלי גבעול (ציר X2) כותרת ואורך עלי כותרת (ציר X1):

גרף של מספר הטעויות המצטבר (בתהליך הלמידה):



גרף מאגר אימון והמישור המפריד:



תוצאות המודל עבור מאגר הבחינה:

```
The boundary decision line is: y=0.67x+0.93
The weight vector is: [ 3.62512222 -5.39871637]
The bias balue is:  5

the confusion matrix is:
[[11  0 11]
 [ 0 19 19]
 [11 19 30]]
the accuracy rate is:
100.000%
```

ניתוח תוצאות ומסקנות:

1. ניתן לראות שבהרצה הראשונה נדרשו 11 מעברים על מאגר האימון עד להתכנסות האלגוריתם, כלומר רק מעבר ה-12 על מאגר האימון שוב לא בוצעו טעויות כלל ($11 \cdot 7 = 770$, סה"כ היו 840 איטרציות). לעומת זאת, בהרצה השנייה כבר במעבר השני על מאגר האימון לא בוצעו טעויות כלל ואף בהרצה זו סך הטעויות היה 5 וכבר לאחר כ-20 איטרציות לא בוצעו טעויות כלל. מכך נסיק בוודאי שבחירת המאפיינים עלולה לשנות את זמן הריצה של התוכנית.
2. ניתן לראות שבהרצה הראשונה עבור המאפיינים הנבחרים רוחב עלי כותרת ורוחב עלי גביע ישנן נקודות קרובות מאוד למישור המפריד (דבר אשר עלול ליצור התאמת יתר) ואילו בהרצה השנייה עבור המאפיינים רוחב עלי גביע ואורך עלי כותרת, המישור מפריד בצורה ברורה יותר את הדאטה.
3. ההיפר מישורים המשמשים לקביעת זן הפרח הם:

$$petal\_width = 1.16 \cdot sepal\_width + 2.35$$

$$sepal\_width = 0.67 \cdot petal\_length + 0.93$$

4. בחינת המודל – מטריצת הערפול שקיבלנו בשתי ההרצות הייתה זהה והיא:

Predicted\real	Versicolor	Setosa	sum
Versicolor	11	0	11
Setosa	0	19	19
sum	11	19	30

למעשה, בבחינת המודל קיבלנו דיוק של 100%. ברור שזו תוצאה טובה, אך נסייג בזה כי המאגר שלקחנו הינו קטן ויתכן כי אינו מייצג במיוחד את כלל אוכלוסיית הפרחים השייכים לשני הזנים הללו.

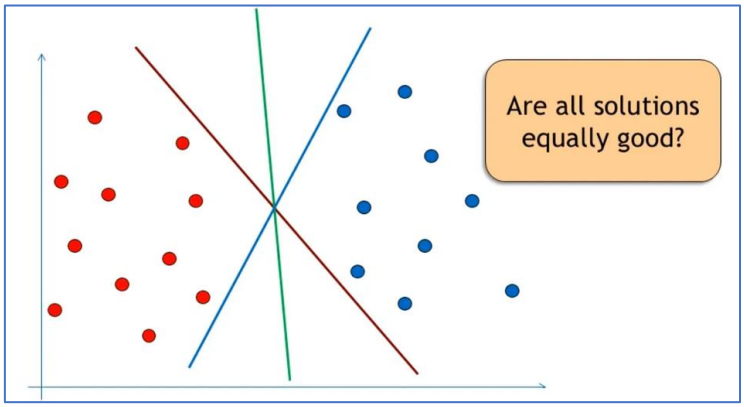
3 אלגוריתם SVM, מכונת ווקטורים תומכים

3.1 SVM Hard Margin – הפרדה קשיחה

מידול הבעיה:

כעת, נרצה למצוא היפר-מישור אשר מקנה לנו הפרדה קשיחה. כלומר, מישור המסווג את הדוגמאות בצורה שבה הוא יוצר את המרווח המקסימלי שניתן בינו לבין הדגימות הקרובות לו בשתי קבוצות הסיווג.

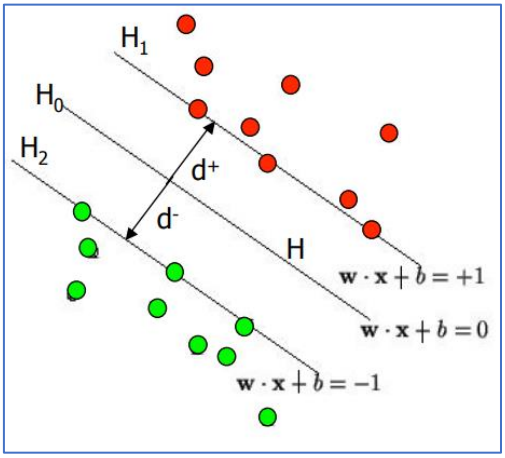
דוגמא:



Atul Agarwal, Sep 24,2019, Support Vector Machine - Formulation and Derivation, Medium

כפי שניתן לראות בדוגמא שלושת הישרים מהווים ישרים מפרידים בין הדגימות הכחולות והדגימות האדומות. ובכן, קיימים אינסוף ישרים כאלו אשר יוכלו לשמש כמסווגים בין הדגימות.
אם כך, נידרש לשאלה איזה מבין הישרים הוא המסווג הטוב ביותר עבורנו? התשובה לכך היא – הישר היוצר לנו את המרווח הגדול ביותר בין הנקודות הקרובות אליו מבין כל קבוצת סיווג.
כלומר, הישר שמייצר את השוליים הגדולים ביותר.
לשם כך, ננסה למצוא את ההיפר מישור העונה על הדרישות.

<u>סקיצה:</u>



Anshultrivedi, Nov 14,2020, Overview fo Support Vector Machines, Medium

המישור המסווג הוא $H_0: W^T X + b = 0$, וכל דגימה ירוקה $(y_i = -1)$ מקיימת $W^T X + b \leq -1$ וכל דגימה אדומה $(y_i = +1)$ מקיימת $W^T X + b \geq 1$, אנחנו מחפשים את ההיפר-מישור שיסווג לנו את הדגימות כך שהמרחק בין המישור $H_1: W^T X + b = 1$ לבין המישור $H_2: W^T X + b = -1$, יהיה המקסימלי ביותר. למעשה, H_1 ו- H_2 הם מישורים המקבילים למישור המפריד H_0 ומתלכדים עם דגימה אחת לפחות מכל מחלקה, כלומר H_1 מתלכד עם דגימה אדומה אחת לפחות ו- H_2 עם דגימה ירוקה אחת לפחות. הדגימות המתלכדות עם מישורי השוליים H_1 ו- H_2 נקראות ווקטורים תומכים שהרי כל נקודה היא ווקטור והן משחקות תפקיד מכריע בהגדרת המישור המפריד ובכך "תומכות" בהפרדה בין המחלקות.

אם כך, ננסה להבין מהו המרחק בין שני המישורים H_1 ו- H_2:

נניח שאנחנו נמצאים על המישור H_0 נבדוק כמה "יחידות צעדים" נצטרך ללכת כדי להגיע למישור H_1. ראשית, ברור שנלך בכיוון ווקטור הנורמל למישור H_0 שהרי אנו רוצים למצוא את המרחק (מרחק זה הקטע הכי קצר שמחבר בין המישורים), לכן נלך בכיוון ווקטור היחידה - $\frac{W}{||W||}$.

$$W^T\left(X + d^+ \cdot \frac{W}{||W||}\right) + b = 1 \Rightarrow W^T X + d^+ \cdot W^T \frac{W}{||W||} + b = 1$$

$$\Rightarrow W^T X + d^+ \cdot W^T \frac{W}{||W||} + b = 1 \Rightarrow \left\{W^T W = ||W||^2\right\} \Rightarrow$$

$$\Rightarrow W^T X + d^+ \cdot \frac{||W||^2}{||W||} + b = 1 \Rightarrow W^T X + b + d^+ \cdot ||W|| = 1 \Rightarrow \{W^T X + b = 0\} \Rightarrow$$

$$\Rightarrow d^+ \cdot ||W|| = 1 \Rightarrow d^+ = \frac{1}{||W||}$$

באותו אופן, נחשב את d^-.

לכן, גודל השוליים:

$$d = d^+ + d^- = \frac{1}{||W||} + \frac{1}{||W||} = \frac{2}{||W||}$$

נרצה למצוא את הישר הממקסם את המרחק d, זו בעיה מתמטית בתחום האופטימיזציה.

3.1.1 הגדרת בעיית האופטימיזציה

$$Max: \frac{2}{||W||}$$

$s.t:$

$W^T X_i + b \geq 1 \ (for\ positive\ sample)$

$W^T X_i + b < -1 \ (for\ negative\ sample)$

(*) הערות:

1. המשתנים עליהם אנו מבצעים את האופטימיזציה הם W, b שאותם אנו רוצים למצוא.
2. מיקסום הביטוי - $\frac{2}{||W||}$ זה לדרוש מינימום עבור $||W||$.

בצורה נוחה יותר:

$$min: \frac{1}{2}||W||^2 = \frac{1}{2}W^T W$$

$s.t:$

$y_i(W^T X_i + b) \geq 1 \ (i = 1, \dots, m)$

3.1.2 קיום ויחידות הפתרון + קמירות הבעיה

<u>פונקציה קמורה - הגדרה:</u>

פונקציה f מוגדרת בתחום I, נקראת קמורה אם לכל $x, y \in I$ ולכל $\lambda \in [0,1]$ מתקיים אי השוויון:

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$$

<u>קבוצה קמורה - הגדרה:</u>

תת קבוצה S של מרחב ווקטורי V נקראת קבוצה קמורה אם לכל $W_1, W_2 \in S$ ולכל $\lambda \in [0,1]$ ממשי, מתקיים:

$$\lambda W_1 + (1-\lambda)W_2 \in S$$

<u>פונקציה קמורה במובן החזק – הגדרה:</u>

פונקציה f מוגדרת בתחום קמור I, נקראת קמורה במובן החזק אם ורק אם לכל $x, y \in I$ כך ש- $x \neq y$ ולכל $\lambda \in (0,1)$ מתקיים אי השוויון:

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda)f(y)$$

בהמשך נראה שההגדרה מתקיימת עבור פונקציית המטרה של בעיית SVM הפרדה קשיחה.

3.1.2.1 טענה – יחידות הפתרון

יהי תחום קמור I. אם פונקציה f קמורה במובן החזק אזי קיים לכל היותר מינימום מקומי אחד של f בתחום I. כתוצאה מכך, אם קיים מינימום גלובלי של f בתחום I , אזי מינימום זה הוא יחיד.

3.1.2.1.1 הוכחה של טענה 3.1.2.1

יהי תחום קמור I ופונקציה f קמורה במובן החזק. נניח בשלילה כי קיימים שתי נ״ק מינימום מקומי ונסמן אותן ב- x^* ו- $\tilde{x}$ כך ש- $x^* \neq \tilde{x}$. נניח ללא הגבלת הכלליות כי מתקיים $f(x^*) \leq f(\tilde{x})$. נסתכל על קבוצת הווקטורים – $x_\lambda = \lambda x^* + (1-\lambda)\tilde{x}$ כך ש- $\lambda \in (0,1)$.

$$f(x_\lambda) = f(\lambda x^* + (1-\lambda)\tilde{x}) \underbrace{<}_{f\ strictly\ convex} \lambda f(x^*) + (1-\lambda) f(\tilde{x}) \underbrace{\leq}_{f(x^*)\leq f(\tilde{x})}$$
$$\leq \lambda f(\tilde{x}) + (1-\lambda) f(\tilde{x}) = f(\tilde{x})$$

אם נסתכל על סביבה של $\tilde{x}$ ונבחר x_λ כך ש- $\lambda \to 0$, אזי x_λ נמצא בתוך הסביבה. אם כך, מצאנו x_λ שנמצאת בסביבה של $\tilde{x}$ אך מתקיים $f(x_\lambda) < f(\tilde{x})$. זה עומד בסתירה לכך ש- $\tilde{x}$ הוא מינימום מקומי. הוכחה זו תקפה גם למינימום גלובלי שהרי כל מינימום גלובלי הוא מינימום מקומי. מכך נסיק שאם קיים מינימום גלובלי אזי הוא יחיד.

3.1.2.2 קמירות של פונקציית המטרה בבעיית SVM הפרדה קשיחה

נוכיח כי הביטוי $W^T W = ||W||^2$ קמור - יהיו W_1, W_2

$$\begin{aligned}
&\lambda||W_1||^2 + (1-\lambda)||W_2||^2 - ||\lambda W_1 + (1-\lambda)W_2||^2 \\
&\quad = \lambda||W_1||^2 + (1-\lambda)||W_2||^2 - \lambda^2||W_1||^2 - (1-\lambda)^2||W_2||^2 \\
&\quad - 2\lambda(1-\lambda)W_1^T W_2 \\
&\quad = \lambda(1-\lambda)||W_1||^2 - 2\lambda(1-\lambda)W_1^T W_2 \\
&\quad + \lambda(1-\lambda)||W_2||^2 \underbrace{=}_{Write\ (3.1.2.2a)} \lambda(1-\lambda)||W_1 - W_2||^2 \geq 0
\end{aligned}$$

לכן, קיבלנו:

$$\lambda||W_1||^2 + (1-\lambda)||W_2||^2 \geq ||\lambda W_1 + (1-\lambda)W_2||^2$$

כלומר,

$$\lambda f(W_1) + (1-\lambda) f(W_2) \geq f(\lambda W_1 + (1-\lambda)W_2)$$

נוכיח שפונקציית המטרה קמורה במובן החזק:

עבור $\lambda \in (0,1)$ ו- $W_1 \neq W_2$, נקבל:

$$\lambda\left||W_1|\right|^2 + (1-\lambda)\left||W_2|\right|^2 - \left||\lambda W_1 + (1-\lambda)W_2|\right|^2 =$$
$$\underbrace{=}_{From\ (3.1.2.2a)} \lambda(1-\lambda)\left||W_1 - W_2|\right|^2 > 0$$

כלומר, עבור $\lambda \in (0,1)$ ו- $W_1 \neq W_2$, נקבל:

$$\lambda\left||W_1|\right|^2 + (1-\lambda)\left||W_2|\right|^2 > \left||\lambda W_1 + (1-\lambda)W_2|\right|^2 \Rightarrow$$
$$\Rightarrow \lambda f(W_1) + (1-\lambda)f(W_2) > f(\lambda W_1 + (1-\lambda)W_2)$$

לסיכום, $W^T W$ היא פונקציה קמורה במובן החזק.

3.1.2.3 קמירות התחום בבעיית SVM הפרדה קשיחה

תחילה, נגדיר את התחום:

$$S = \left\{(W,b): y_i(W^T X_i + b) \geq 1\ , \forall i \in \{1,\dots,m\}\right\}$$

יהיו $(W_1,b_1),(W_2,b_2) \in S$, כלומר –

$$y_i(W_1^T X_i + b_1) \geq 1\ , \forall i \in \{1,\dots,m\}$$
$$y_i(W_2^T X_i + b_2) \geq 1\ , \forall i \in \{1,\dots,m\}$$

כעת, נסמן $(W_\lambda,b_\lambda) = \lambda(W_1,b_1) + (1-\lambda)(W_2,b_2)$ ונראה כי $(W_\lambda,b_\lambda) \in S$:

$$y_i\left(W_\lambda^T X_i + b_\lambda\right) = y_i\left((\lambda W_1 + (1-\lambda)W_2)^T X_i + (\lambda b_1 + (1-\lambda)b_2)\right)$$
$$= y_i \lambda W_1^T X_i + (1-\lambda)W_2^T X_i + \lambda b_1 + (1-\lambda)b_2$$
$$= \lambda \underbrace{y_i(W_1^T\ X\_i + b_1)}_{\geq 1} + (1-\lambda)\underbrace{y_i(W_2^T\ X\_i + b_2)}_{\geq 1}$$
$$\geq \lambda \cdot 1 + (1-\lambda)\cdot 1 = 1 \Rightarrow y_i\left(W_\lambda^T X_i + b_\lambda\right) \geq 1 \Rightarrow (W_\lambda,b_\lambda) \in S$$

כלומר, התחום בבעיית האופטימיזציה של אלגוריתם SVM בהפרדה קשיחה הוא קמור.

3.1.2.4 מסקנה

לבעיית האופטימיזציה SVM בגרסת הפרדה קשיחה קיים לכל היותר פתרון אחד.

3.1.2.5 משפט ויירשטראס

תהי פונקציה $f(x)$ רציפה המוגדרת על קבוצה קומפקטית I, אזי היא מקבלת ערך מינימלי.

3.1.2.6 טענה – קיום הפתרון עבור בעיית SVM הפרדה קשיחה

קיים לפחות פתרון אחד לבעיית האופטימיזציה SVM הפרדה קשיחה.

3.1.2.6.1 הוכחה

נסמן את השטח הפיזיבלי –

$K = \{(W,b): y_i(W^T X_i + b) \geq 1 \,, \forall (X_i, y_i) \in D\}$

לפונקציית המטרה קיים אינפימום שהרי קיים לה חסם תחתון:

$$f(W,b) = \frac{1}{2} W^T W = \frac{1}{2} ||W||^2 \geq 0$$

לכן, קיימת סדרה $(W_n, b_n)_{n\in N} \subseteq K$ *(Minimizing Sequence)* כך שמתקיים:

$$\lim_{n\to\infty} f(W_n, b_n) = \inf\{f(W,b): (W,b) \in K\} = \gamma$$

- <u>מקרה I</u>: הסדרה $(W_n, b_n)_{n\in N}$ חסומה.

  לכן, קיימת תת סדרה מתכנסת $(W_{n'}, b_{n'})_{n\in N} \subseteq (W_n, b_n)_{n\in N}$ כך ש-

$$(W_{n'}, b_{n'})_{n\in N} \overset{n\to\infty}{\Rightarrow} (\overline{W}, \overline{b})$$

מכיוון ש- K היא קבוצה סגורה, $(\overline{W}, \overline{b}) \in K$. מאחר שפונקציית המטרה $f(W,b)$ רציפה,

אזי $\lim_{n\to\infty} f(W_n, b_n) = f(\overline{W}, \overline{b})$. יתרה מזאת,

$$\lim_{n\to\infty} f(W_{n'}, b_{n'}) = \lim_{n\to\infty} f(W_n, b_n) = \gamma$$

כלומר, קיבלנו $f(\overline{W}, \overline{b}) = \gamma$. ומכך נסיק ש- $(\overline{W}, \overline{b})$ הוא אינפימום ששייך ל- K ולכן הוא מינימום.

- <u>מקרה II</u>: הסדרה $(W_n, b_n)_{n\in N}$ לא חסומה.

  לכן, קיימת תת סדרה $(W_{n'}, b_{n'})_{n\in N} \subseteq (W_n, b_n)_{n\in N}$ כך ש-

$$\lim_{n\to\infty} ||(W_{n'}, b_{n'})|| = +\infty$$

אם כך, נבחין בין שני תתי מקרים:

<u>מקרה $II.a$</u>: תת הסדרה $\{b_{n'}\}_{n\in N}$ חסומה.

מכך נסיק ש- $\lim_{n\to\infty} ||W_{n'}|| = +\infty$. לכן, מתקיים:

$$\lim_{n\to\infty} f(W_{n'}, b_{n'}) = \lim_{n\to\infty} \frac{1}{2} ||W_n'||^2 = \infty$$

אבל זה עומד בסתירה לעובדה ש-

$$\lim_{n\to\infty} f(W_{n'}, b_{n'}) = \lim_{n\to\infty} f(W_n, b_n) = \gamma < \infty \;, \qquad K \text{ non empty set}$$

- <u>מקרה $II.b$</u>: תת הסדרה $\{b_{n'}\}_{n\in N}$ לא חסומה.

  אזי קיימת תת סדרה $\{b_{n''}\}_{n\in N} \subseteq \{b_{n'}\}_{n\in N}$ לא חסומה כך ש-

$$\lim_{n\to\infty} |b_{n''}| = \infty$$

לפחות אחת משתי האפשרויות הבאות קיימות: קיימת תת סדרה אינסופית $b_{n'''}$ של ערכים חיוביים או לחילופין קיימת תת סדרה אינסופית $b_{n'''}$ של ערכים שליליים.

נניח שקיימת תת סדרה $\{b_{n'''}\}_{n\in N} \subseteq \{b_{n''}\}_{n\in N}$ כך ש- $b_{n'''} > 0$ לכל $n''' \in N$. כעת, נבחר דגימה (X_{iN}, y_{iN}) אשר התווית שלה שלילית (*Negative*), כלומר $y_{iN} = -1$.

מתקיים:

$$(II.b.\#)\ W_{n'''}^T X_{iN} + b_{n'''} \le -1$$

מכך נסיק:

$$|b_{n'''}| \underbrace{=}_{b_{n'''}>0} b_{n'''} \underbrace{\le}_{(II.b.\#)} -1 - W_{n'''}^T X_{iN}$$

$$\le -1 + \left|\left|W_{n'''}^T X_{iN}\right|\right| \underbrace{\le}_{Cauchy\ Schwartz} -1 + \left|\left|W_{n'''}\right|\right| \cdot ||X_{iN}||$$

אך מאחר שאנו יודעים כי $\lim_{n\to\infty} |b_{n''}| = \infty$ אז $\lim_{n\to\infty} ||W_{n'''}|| = \infty$. אבל זו סתירה לעובדה ש-

$$\lim_{n\to\infty} f(W_{n''''}, b_{n'''}) = \lim_{n\to\infty} \frac{1}{2}\left|\left|W_n'''\right|\right|^2 = \gamma < \infty$$

באופן דומה, עבור האפשרות השנייה בה קיימת תת סדרה $\{b_{n'''}\}_{n\in N} \subseteq \{b_{n''}\}_{n\in N}$ כך ש- $b_{n'''} < 0$ לכל $n''' \in N$. נבחר דגימה (X_{iP}, y_{iP}) אשר התווית שלה חיובית (*Positive*), כלומר $y_{iP} = 1$. מתקיים:

$$(II.b.\#\#)\ W_{n'''}^T X_{iP} + b_{n'''} \ge 1$$

מכך נסיק:

$$|b_{n'''}| \underbrace{=}_{b_{n'''}<0} -b_{n'''} \underbrace{\le}_{(II.b.\#\#)} -1 + W_{n'''}^T X_{iP}$$

$$\le -1 + \left|\left|W_{n'''}^T X_{iP}\right|\right| \underbrace{\le}_{Cauchy\ Schwartz} -1 + \left|\left|W_{n'''}\right|\right| \cdot ||X_{iP}||$$

אך מאחר שאנו יודעים כי $\lim_{n\to\infty} |b_{n''}| = \infty$ אז $\lim_{n\to\infty} ||W_{n'''}|| = \infty$. אבל זו סתירה לעובדה ש-

$$\lim_{n\to\infty} f(W_{n''''}, b_{n'''}) = \lim_{n\to\infty} \frac{1}{2}\left|\left|W_n'''\right|\right|^2 = \gamma < \infty$$

לסיכום, הוכחנו כי רק מקרה I יכול להתקיים שלפיו קיים פתרון לבעיית האופטימיזציה של אלגוריתם SVM הפרדה קשיחה.

3.1.2.7 מסקנה (נובע מטענות 3.1.2.4, 3.1.2.6)

לבעיית האופטימיזציה באלגוריתם SVM – הפרדה קשיחה, קיים פתרון יחיד.

3.1.3 הבעיה הדואלית

ננסה להפוך את בעיית האופטימיזציה הראשונית הנ"ל:

$$min: \frac{1}{2}||W||^2 = \frac{1}{2}W^TW$$

$$s.t:$$

$$y_i(W^TX_i + b) \geq 1 \ (i = 1, \dots, m)$$

לבעיה דואלית.

ונשתמש בשיטת לגרנג' כדי לנתח משמעויות.

<u>פונקציית לגרנג':</u>

$$L_p = \frac{1}{2}W^TW + \sum_{i=1}^{m} a_i\big(1 - y_i(W^TX_i + b)\big)$$

(*) הערות:

- המשתנים בהם פונקציה L_p תלויה הם - W, b, α_i.(כלומר - $L_p = L_p(W, b, \alpha_i)$)
- נבהיר כי למעשה אנו משתמשים במכפלה פנימית המוגדרת בצורה הבאה:

$$< u, v >= u^Tv$$

שאותה אנו מגדירים על ווקטורים עם רכיבים ממשיים ולכן מתקיים:

$$< u, v > = u^Tv = v^Tu = < v, u >$$

$$L_P = \frac{1}{2}W^TW + \sum_{i=1}^{m} a_i\big(1 - y_i(W^TX_i + b)\big) \Rightarrow$$

$$\Rightarrow L_P = \frac{1}{2} < W, W > + \sum_{i=1}^{m} a_i\big(1 - y_i(< W, X_i > + b)\big)$$

$$\Rightarrow L_P = \frac{1}{2} < W, W > + \sum_{i=1}^{m} a_i - \sum_{i=1}^{m} a_i\big(y_i(< W, X_i > + b)\big)$$

כעת, נחשב נגזרות (ביחס למשתנים W ו-b).

$$\frac{\partial L_P}{\partial W} = W - \sum_{i=1}^{m} a_i y_i, X_i = 0 \Rightarrow W = \sum_{i=1}^{m} a_i y_i X_i$$

$$\frac{\partial L_P}{\partial b} = \sum_{i=1}^{m} a_i y_i = 0$$

יש לשים לב, כי עבור הווקטורים הנמצאים על המישורים המפרידים H_1 ו- H_2 נרצה שכופלי הלגרנג' עבורם יתאפסו שהרי הם לא אמורים לתרום לנו לחישוב ווקטור המשקולות ועל השונות (bias). כעת, נציב את זה ב-L_P כדי לקבל פונקציה שתלויה בכופלי הלגרנג' בלבד.

נהפוך את הבעיה לצורה הדואלית:

$$L_D = \frac{1}{2} < W, W > + \sum_{i=1}^{m} a_i - \sum_{i=1}^{m} a_i\big(y_i(< W, X_i > + b)\big) =$$

$$L_D = \frac{1}{2} < \sum_{i=1}^{m} a_i y_i X_i \,, \sum_{j=1}^{m} a_j y_j X_j > + \sum_{i=1}^{m} a_i - \sum_{i=1}^{m} a_i \left(y_i \left(< \sum_{j=1}^{m} a_j y_j X_j \,, X_i > + b \right) \right) =$$

$$= \frac{1}{2} < \sum_{i=1}^{m} a_i y_i X_i \,, \sum_{j=1}^{m} a_j y_j X_j > + \sum_{i=1}^{m} a_i - \sum_{i=1}^{m} a_i y_i < \sum_{j=1}^{m} a_j y_j X_j \,, X_i > + b \sum_{i=1}^{m} a_i y_i$$

$$= \left\{ \sum_{i=1}^{m} a_i y_i = 0 \right\} =$$

$$= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j y_i y_j < X_i, X_j > + \sum_{i=1}^{m} a_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j y_i y_j < X_i, X_j >$$

$$= \sum_{i=1}^{m} a_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j y_i y_j < X_i, X_j >$$

לכן, הבעיה הדואלית היא:

$$max: L_D = \frac{1}{2} \sum_{i=1}^{m} a_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j y_i y_j < X_i, X_j >$$

$s.t:$

$$\sum_{i=1}^{m} a_i y_i = 0$$

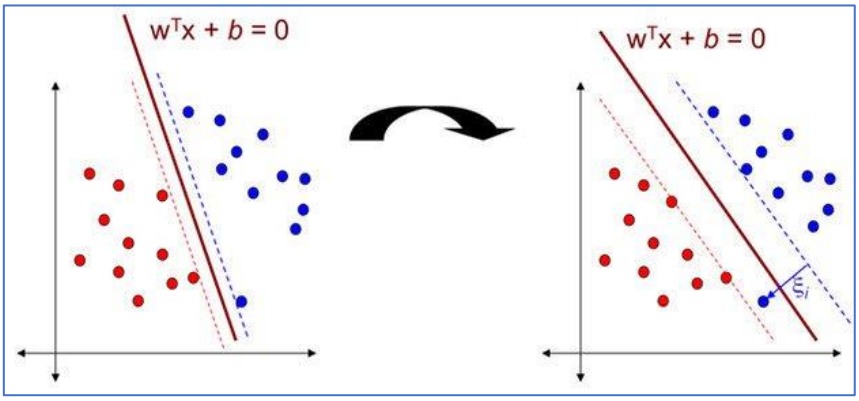
$$\forall i \; a_i \geq 0$$

3.1.4 הבעייתיות בהפרדה קשיחה

1. הפרדה קשיחה דורשת שכל הדגימות יהיו ניתנות להפרדה ליניארית לפי המחלקות שלהן.
2. הפרדה קשיחה לא יוצרת כלליות למודל (מייצרת overfitting). לדוגמא – אם קיימת דגימה שלא מייצגת את המדגם אזי האלגוריתם יבנה את מישור ההפרדה בהתייחס גם לאותה דגימה.

המחשה לסעיף 2 :

(מצד שמאל הפרדה קשיחה ומצד ימין הפרדה אחרת שנקראת הפרדה רכה)



Mubaris NK,14 October 2022, Support Vector Machines for Classification

אם כך, איך נפתור את הבעיות הללו? באמצעות הפרדה רכה...

3.2 הפרדה רכה (Soft Margin)

הפרדה רכה היא למעשה הפרדה שמאפשרת איזון בין הרצון ליצור שוליים רחבים למול הרצון שהדגימות יסווגו נכון. הפרדה זו מאפשרת את הימצאותם של דגימות "בצד הלא נכון" של המסווג (תכונה זו מאפשרת ליצור הפרדה גם במקרה שהנתונים לא ניתנים להפרדה ליניארית על ידי המחלקות).

3.2.1 בעיית האופטימיזציה של הפרדה רכה

בהפרדה רכה אנו "מענישים" את פונקציית המטרה במקרה של סיווג לא נכון.

$$min: \frac{1}{2}W^TW + C \cdot \sum_{i=1}^{m} \zeta_i$$

כאשר n זה אורך ווקטור המשקולות ($W \in R^n$), ו-m זה כמות הדגימות.

נהוג לסמן :

$$hinge\ function: \sum_{i=1}^{m} \zeta_i$$

הסבר:

ζ_i – המרחק של דגימה שלא נמצאת "מעל או מתחת לווקטורים התומכים בהתאם למחלקה שלה (אם דגימה חיובית – מעל הווקטור התומך ואם הדגימה שלילית מתחת לווקטור התומך).

באופן מתמטי $y_i(W^T X_i + b) \geq 1 - \zeta_i$ כאשר $\zeta_i \geq 0$.

לדוגמא, באיור לעיל בשרטוט הימני ניתן לראות כי קיימת דגימה כחולה שנמצאת מתחת להיפר המישור $W^T X + b = 1$, למרות שהיא אמורה להיות מעליו. את מרחקה מהיפר המישור הזה נסמן ב- ζ_i. נגדיר את המרחק הזה באופן הבא:

$$\zeta_{\mathrm{i}} = \max\{0, 1 - y_i(W^T X_i + b)\}\ ,\ \zeta_{\mathrm{i}} \geq 0.$$

מדוע נגדיר אותו באופן הנ"ל:

אנחנו נגיד כי דגימה סווגה בצורה נכונה אם היא נמצאת בצד הנכון של המסווג וגם לא נמצאת בתחום שבין הווקטורים התומכים. לכן, אם דגימה סווגה נכון המרחק ζ_{i} יהיה שווה לאפס.

אילו הדגימה לא סווגה נכון מתקיים:

$$y_i(W^T X_i + b) < 1 \Rightarrow 1 - y_i(W^T X_i + b) > 0$$

נגדיר את קבוצת "הסיווגים השגויים":

$$such\ that\colon E = \{\, j \mid y_j\big(W^T X_j + b\big) < 1\}$$

נוכל להגדיר גם בצורה הבאה:

$$hinge\ function\colon \sum_{i \in E}\big(1 - y_i(W^T X_i + b)\big)$$

פרמטר הענישה - $C\ parameter$:

פרמטר הענישה הוא פרמטר שאנו קובעים בהתאם למטרות שלנו. ככל ש- C יהיה גדול כך כמות הדגימות המסווגות באופן שגוי יקטן. לעומת זאת, ככל ש- C יהיה קטן כך רוחב השוליים יהיה גדול יותר.

3.2.2 קמירות הבעיה

בסעיף 3.1.2 הוכחנו כי בעיית האופטימיזציה של אלגוריתם SVM תחת הפרדה קשיחה מהווה בעיה קמורה. כעת, נראה כי גם בעיית האופטימיזציה של אלגוריתם SVM תחת הפרדה רכה מהווה בעיה קמורה. לשם כך, נצטרך להוכיח כי פונקציית המטרה היא פונקציה קמורה. כדי להוכיח זאת, נשתמש בתיאוריה מתמטית על קמירות של פונקציות.

3.2.2.1 טענה (סכום של פונקציות קמורות)

יהיו $f(x), g(x)$ פונקציות קמורה בתחום קמור I אזי הפונקציה $h(x) = f(x) + g(x)$ היא פונקציה קמורה בתחום I.

3.2.2.1.1 הוכחה

נניח ש- $f(x), g(x)$ פונקציות קמורות. אזי לכל $x_1, x_2 \in I$ ו- $\lambda \in [0,1]$ מתקיים:

$$f(\lambda x_1 + (1-\lambda)x_2) \le \lambda \cdot f(x_1) + (1-\lambda) \cdot f(x_2)$$

$$g(\lambda x_1 + (1-\lambda)x_2) \le \lambda \cdot g(x_1) + (1-\lambda) \cdot g(x_2)$$

כעת, נראה שהפונקציה $h(x) = (f+g)(x)$ קמורה בתחום I.

$$\begin{aligned}
h(\lambda x_1 + (1-\lambda)x_2) &= (f+g)(\lambda x_1 + (1-\lambda)x_2) \\
&= f(\lambda x_1 + (1-\lambda)x_2) + g(\lambda x_1 + (1-\lambda)x_2) \underset{f,g\ convex\ functions}{\le} \\
&\le \lambda \cdot f(x_1) + (1-\lambda) \cdot f(x_2) + \lambda \cdot g(x_1) + (1-\lambda) \cdot g(x_2) \\
&= \lambda \cdot \big(f(x_1) + g(x_1)\big) + (1-\lambda) \cdot \big(f(x_2) + g(x_2)\big) \\
&= \lambda \cdot \big((f+g)(x_1)\big) + (1-\lambda) \cdot \big((f+g)(x_2)\big) \\
&= \lambda \cdot \big(h(x_1)\big) + (1-\lambda) \cdot \big(h(x_2)\big)
\end{aligned}$$

כלומר, קיבלנו –

$$h(\lambda x_1 + (1-\lambda)x_2) \le \lambda \cdot h(x_1) + (1-\lambda) \cdot h(x_2)$$

3.2.2.2 טענה (מקסימום של פונקציות קמורות)

יהיו $f(x), g(x)$ פונקציות קמורה בתחום קמור I אזי הפונקציה $M(x) = \max\big(f(x), g(x)\big)$ היא פונקציה קמורה בתחום I.

3.2.2.2.1 הוכחה

יהיו $f(x), g(x)$ פונקציות קמורה בתחום קמור I. אזי לכל $x_1, x_2 \in I$ ו- $\lambda \in [0,1]$ מתקיים:

$$f(\lambda x_1 + (1-\lambda)x_2) \le \lambda \cdot f(x_1) + (1-\lambda) \cdot f(x_2)$$

$$g(\lambda x_1 + (1-\lambda)x_2) \le \lambda \cdot g(x_1) + (1-\lambda) \cdot g(x_2)$$

כעת, נראה שהפונקציה $M(x) = \max\big(g(x), f(x)\big)$ קמורה בתחום I.

$$\begin{aligned}
M(\lambda x_1 + (1-\lambda)x_2) &= \max\big(f(x_1 + (1-\lambda)x_2), g(x_1 + (1-\lambda)x_2)\big) \\
&= \{Suppose\ WLOG: g(x_1 + (1-\lambda)x_2) \ge f(x_1 + (1-\lambda)x_2)\} \\
&= f(x_1 + (1-\lambda)x_2) \le \lambda \cdot f(x_1) + (1-\lambda) \cdot f(x_2) \\
&\le \lambda \cdot \max\big(f(x_1), g(x_1)\big) + (1-\lambda) \cdot \max\big(f(x_2), g(x_2)\big) \\
&\le \lambda \cdot M(x_1) + (1-\lambda) \cdot M(x_2)
\end{aligned}$$

כלומר, קיבלנו –

$$M(\lambda x_1 + (1-\lambda)x_2) \leq \lambda \cdot M(x_1) + (1-\lambda) \cdot M(x_2)$$

3.2.2.3 קמירות בעיית ההפרדה הרכה

תזכורת: פונקציית המטרה של בעיית ההפרדה הרכה –

$$\frac{1}{2}W^T W + C \cdot \sum_{i=1}^{m} \max\{0, 1 - y_i(W^T X_i + b)\}$$

- $\frac{1}{2}W^T W$: בבעיית ההפרדה הקשיחה הוכחנו כי הביטוי מהווה פונקציה קמורה.
- $1 - y_i(W^T X_i + b)$: ביטוי זה מייצג פונקציה ליניארית וידוע כי פונקציה ליניארית היא פונקציה קמורה.
- $\max\{0, 1 - y_i(W^T X_i + b)\}$: מקסימום מבין שתי פונקציות קמורות הוא קמור לפי טענה 3.2.2.2 . (ברור שהפונקציה הקבועה 0 היא פונקציה קמורה)
- $C \cdot \sum_{i=1}^{m} \max\{0, 1 - y_i(W^T X_i + b)\}$: סכום של פונקציות קמורות היא פונקציה קמורה לפי טענה 3.2.2.1 .

לסיכום, בעיית האופטימיזציה של אלגוריתם SVM הפרדה רכה היא בעיה קמורה.

3.3 פתרון בעיית SVM באמצעות שיטת Batch Gradient Descent

לאחר שהוכחנו שזו בעיה קמורה, נוכל להשתמש בשיטת הגרדיאנט.

$$F(W, b) = \frac{1}{2}W^T W + C \cdot \sum_{i=1}^{m} \max\{0, 1 - y_i(W^T X_i + b)\}$$

לשם נוחות, נסמן ב - $w^{(j)}$ את הקואורדינטה ה-jית של ווקטור W .

(*) סימון: $X_i^{(j)}$ – הקואורדינטה ה-jית בווקטור X של הדגימה ה-i.

$$F(W, b) = \frac{1}{2}\sum_{j=1}^{n}\left(w^{(j)}\right)^2 + C \cdot \sum_{i \in E}\left(1 - y_i\left(\sum_{j=1}^{n} w^{(j)} x_i^{(j)} + b\right)\right)$$

$$\nabla F\left(b, w^{(1)}, w^{(2)}, \dots, w^{(n)}\right) = \nabla\left(\frac{1}{2}W^T W\right) + C \cdot \nabla hinge\_function$$

לשם נוחות, נסמן:

$$hinge\_function = \sum_{i \in E} h_i$$

כאשר $h_i = 1 - y_i\left(\sum_{j=1}^{n} w^{(j)} x_i^{(j)} + b\right)$.

נחשב נגזרות חלקיות עבור הדגימה ה-i:

$$\frac{\partial h_i}{\partial w^{(k)}} = \begin{cases} 0, & y_i(W^T X_i + b) > 1 \\ -y_i X_i^{(k)}, & y_i(W^T X_i + b) < 1 \end{cases}$$

$$\frac{\partial h_i}{\partial b} = \begin{cases} 0, & y_i(W^T X_i + b) > 1 \\ -y_i, & y_i(W^T X_i + b) < 1 \end{cases}$$

$$\frac{\partial(\frac{1}{2} W^T W)}{\partial w^{(k)}} = \frac{\partial(\frac{1}{2} \sum_{j=1}^{n} \left(w^{(j)}\right)^2)}{\partial w^{(k)}} = w^{(k)}$$

$$\frac{\partial(\frac{1}{2} W^T W)}{\partial b} = 0$$

נמצא נוסחה לקורדינטות הגרדיאנט עבור הדגימה ה-i:

$$\frac{\partial F_i}{\partial w^{(k)}} = w^{(k)} + C \cdot \frac{\partial h_i}{\partial w^{(k)}}$$

באופן מפורש (עבור הדגימה ה-i):

$$\frac{\partial F_i}{\partial w^{(k)}} = \begin{cases} w^{(k)}, & y_i(W^T X_i + b) > 1 \\ w^{(k)} + C \cdot (-y_i x_i^{(k)}), & y_i(W^T X_i + b) < 1 \end{cases}$$

$$\frac{\partial F_i}{\partial b} = \begin{cases} 0, & y_i(W^T X_i + b) > 1 \\ -C \cdot y_i, & y_i(W^T X_i + b) < 1 \end{cases}$$

נרשום בצורה ווקטורית:

$$\frac{\partial F_i}{\partial W} = \begin{cases} W, & y_i(W^T X_i + b) > 1 \\ W + C \cdot (-y_i X_i), & y_i(W^T X_i + b) < 1 \end{cases}$$

$$\frac{\partial F_i}{\partial b} = \begin{cases} 0, & y_i(W^T X_i + b) > 1 \\ -C \cdot y_i, & y_i(W^T X_i + b) < 1 \end{cases}$$

פתרון בעיית SVM באמצעות שיטת Batch Gradient Descent:

Batch Gradient Descent for SVM:

1. Initialize W, b – random values
2. Write $m \to |dataset|$
3. for k=1,…, number of iterations:
4. Initialize the gradient vectors: $\frac{\partial F}{\partial W} \to \vec{0}, \frac{\partial F}{\partial b} \to 0$
5. For each sample - $(X_i, y_i) \in dataset$:
6. Initialize the sum of hinge gradient vectors: $\frac{\partial h}{\partial W} \to \vec{0}, \frac{\partial h}{\partial b} \to 0$
7. If $y_i(W^T X_i + b) < 1$:
8. $\frac{\partial h}{\partial W} \to \frac{\partial h}{\partial W} + C \cdot (-y_i X_i)$
9. $\frac{\partial h}{\partial b} \to \frac{\partial h}{\partial b} + C \cdot (-y_i)$
13. $\frac{\partial F}{\partial W} \to W + \frac{\partial h}{\partial W}$
14. $\frac{\partial F}{\partial b} \to \frac{\partial h}{\partial b}$
15. $W \to W - lr \cdot \frac{\partial F}{\partial w} = (1 - lr) \cdot W - lr \cdot \frac{\partial h}{\partial W}$
16. $b \to b - lr \cdot \frac{\partial F}{\partial b} = b - lr \cdot \frac{\partial h}{\partial b}$

(*) הערה: lr $(learning\ rate)$ מסמן את שיעור הלמידה שזה למעשה גודל הצעד שנעשה בכיוון המנוגד לכיוון הגרדיאנט לצורך מציאת הערך המינימלי.

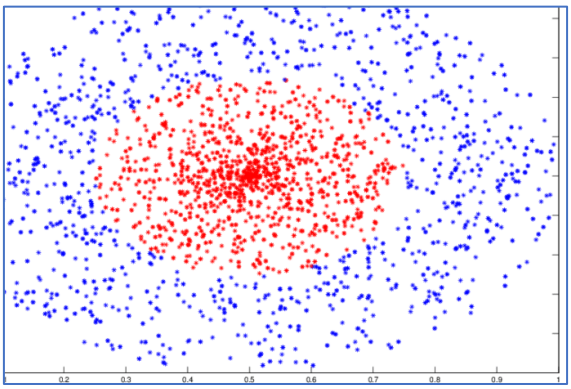
3.4 הסתייגות מאלגוריתם SVM תחת הפרדה רכה

אחד היתרונות של פתרון אלגוריתם SVM תחת "הפרדה רכה" למול "הפרדה קשיחה" היא שלא קיימת הדרישה לכך שיהיו מופרדים ליניארית. יחד עם זאת, אין זה אומר שאלגוריתם זה יספק לנו פתרון טוב לכל מקרה של בעיית סיווג אשר לא עומדת בדרישה כי המחלקות לא ניתנות להפרדה ליניארית. שהרי בסופו של דבר גם אלגוריתם Soft Margin – SVM מספק לנו "היפר-מישור" שמתיימר לייצר הפרדה בין הנתונים. אך, ישנן בעיות סיווג שלא מאפשרות סיווג על ידי "היפר מישור ליניארי".

דוגמא עבור המרחב - R^2:

נניח כי קיים לנו מאגר דגימות עם שתי מחלקות כך שבהסתמך על ווקטור הנתונים, המחלקות ניתנות להפרדה על ידי מעגל.

<u>המחשה</u>:



Mugizi Robert Rwebangira (2014). Learning by combining Native Features Similarity Functions. figure 4.3.

ברור שבמצב כזה כל אלגוריתם שמספק לנו פתרון לבעיית הסיווג על ידי מסווג ליניארי - קו ישר ("היפר מישור במרחב R^2) לא יספק לנו פתרון טוב לבעיה הנתונה.

ננסה לפתור בעיית סיווג זאת באמצעות אלגוריתם Soft Margin – SVM, על ידי שינוי מימד המרחב כך שבמרחב החדש נוכל להפריד בין הנתונים על ידי היפר מישור.

3.5 שיטת הגרעין (Kernel Method)

(*) הערה: נציין כי יש הבדל בין שיטת הגרעין לבין תעלול הגרעין.

שיטת הגרעין נועדה לפתור בעיוות סיווג שפתרונן לא ניתן על ידי מפריד ליניארי. שיטת הגרעין מעתיקה את דוגמאות האימון מהמרחב המקורי למרחב במימד אחר מתאים כך שבמרחב החדש ימצא מסווג ליניארי טוב שלא כמו במרחב המקורי. ההעתקה מבוצעת על ידי שינוי המכפלה הפנימית לפונקציית גרעין מתאימה (בהתאם לבעיית הסיווג).

3.5.1 שימוש בשיטת הגרעין

נגדיר פונקציה מתמטית:

$$\phi: R^n \to R^L \ such\ that\ L \geq n$$

פונקציה זו מקבלת את ווקטורי X, נזכיר ש - $X \in R^n$ וכמות הדגימות ברשותנו היא m.

כך שבמרחב R^L קיים "היפר-מישור" שהוא יכול להוות מסווג ליניארי של הדאטה "החדש" - $\{\phi(X_i), y_i\}_{i=1}^m$.

כלומר, קיים היפר-מישור $W^T\phi + b = 0$, כך שלכל i (עבור כל דגימה) מתקיים:

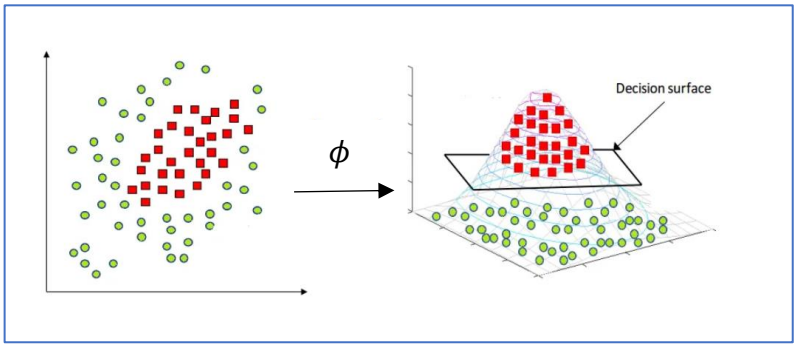
$$y_i \cdot (W^T\phi(X_i) + b) = y_i\left(b + \sum_{k=1}^{L} w_k\varphi_k(X_i)\right) \geq 0$$

(*) הערה: ווקטור המשקולות שאנו רוצים למצוא הוא ווקטור המשקולות עבור המרחב החדש ולכן $W \in R^L$. נציין כי φ_k – מייצג את הקואורדינטה ה-k בווקטור ϕ.

3.5.2 דוגמאות לשיטת הגרעין

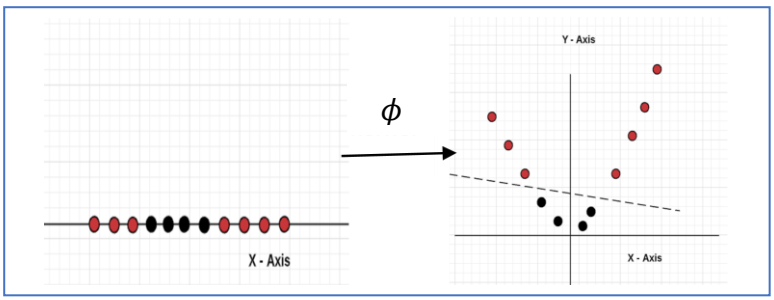
<u>המחשה להעתקות ϕ:</u>

1. העתקה לפונקציה שקווי הרמה שלה הם מעגל:



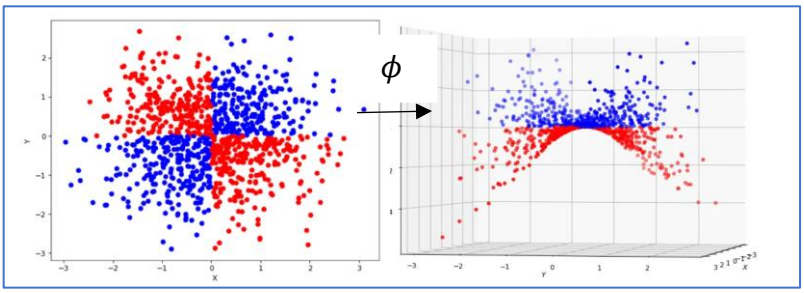
Zhang, G. (2018, November 11). "What is the kernel trick? Why is it important?" Medium

2. העתקה פולינומיאלית:



Ajitesh Kumar, (July 16,2020), Machine Learning - SVM Kernel Trick Example, Data analitics

3. העתקה המבצעת מכפלת קואורדינטות - $\phi(X) = \begin{pmatrix} x_1 \\ x_2 \\ x_1 \cdot x_2 \end{pmatrix}$:



Konstantin Kutzkov (Sep 2,2021), Explicit feature maps for non-linear kernel functions, medium

3.6 תעלול הגרעין בהפרדה קשיחה (Kernel Trick)

אחת הבעיות בשיטת הגרעין היא סיבוכיות זמן הריצה. כאשר יש לנו כמות דגימות גדולה ופונקציית הגרעין ϕ ממימד גבוה מאוד, דבר זה יגדיל את זמן הריצה באופן משמעותי. על כן, משתמשים בטריק הגרעין – נרחיב עליו במעט.

ניזכר בבעיה הדואלית עבור SVM "הפרדה קשיחה":

$$max: L_D = \frac{1}{2}\sum_{i=1}^{m} a_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} a_i a_j y_i y_j < X_i, X_j >$$

$s.t:$

$$\sum_{i=1}^{m} a_i y_i = 0$$

$\forall i \; a_i \geq 0$

למעשה, השתמשנו במכפלה הסקלרית $< X_i, X_j >$, כעת כאשר בעיית הסיווג לא ליניארית נוכל להחליף מכפלה זו ל- $< \phi(X_i), \phi(X_j) >$. נסמן -

$k_{i,j} = k(X_i, X_j) < \phi(X_i), \phi(X_j) >$ ונרשום את הבעיה בצורה החדשה:

$$max: L_D = \frac{1}{2}\sum_{i=1}^{m} a_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} a_i a_j y_i y_j k_{i,j}$$

$s.t:$

$$\sum_{i=1}^{m} a_i y_i = 0$$

$\forall i \; a_i \geq 0$

בדרך זו, אין אנו צריכים לבצע העתקה עבור כל דגימה למימד החדש, אלא לחשב את המכפלות הפנימיות שבין הדגימות. דבר זה מועיל מאוד ומשפר את זמן הריצה בעיקר במקרים בהם כמות הפיצ'רים גבוה.

<u>דוגמא לחשיבות תעלול הגרעין עבור העתקה פולינומיאלית בסיבוכיות זמן הריצה:</u>

נניח שקיים לנו מאגר נתונים אשר ניתן להפרדה ליניארית לאחר הטרנספורמציה הבאה:

$$\phi: R^n \to R^{n^2},\ \ \phi(X) = \begin{pmatrix} x_1x_1 \\ x_1x_2 \\ \vdots \\ x_1x_n \\ x_2x_1 \\ \vdots \\ x_2x_n \\ x_3x_1 \\ \vdots \\ x_3x_n \\ \vdots \\ x_nx_1 \\ \vdots \\ x_nx_n \end{pmatrix}$$

כעת, ננסה ליישם את תעלול הגרעין:

נניח ש- $X, Z \in R^n$. נגדיר את הגרעין באופן הבא:

$$k(X,Z) = (X^TZ)^2$$

ונראה כי $k(X,Z) = \phi(X)^T\phi(Z)$:

$$k(X,Z) = (X^TZ)^2 = \left(\sum_{i=1}^{n} x_iz_i\right)\left(\sum_{j=1}^{n} x_jz_j\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} x_iz_ix_jz_j = \sum_{i=1}^{n}\sum_{j=1}^{n} x_ix_jz_iz_j$$
$$= \phi(x)^T\phi(Z)$$

כעת, ננתח את זמני הריצה:

סיבוכיות זמן הריצה של טרנספורמציה עבור כל דגימה היא $O(n^2)$ שהרי אנו מבצעים העתקה למרחב R^{n^2}. לעומת זאת, חישוב המכפלות הפנימיות באמצעות תעלול הגרעין "תעלה" $O(n)$ שהרי החישוב X^TZ היא $O(n)$ וברור שמ- X^TZ אנו מקבלים מספר ממשי שהעלאתו בריבוע היא $O(1)$. לכן, בדוגמא זו ניתן לראות בבירור את תרומתו של תעלול הגרעין על זמן הריצה של התוכנית.

(*) הערה: נציין כי פונקציה יכולה לשמש כגרעין אם ורק אם מטריצת הגרעין המוגדרת באופן הבא - $G_{i,j} = K(X_i, X_j)$ היא מטריצה סימטרית (כלומר, $G_{i,j} = G_{j.i}$) ומוגדרת אי שלילית (כלומר, לכל $X \in R^m$ מתקיים $X^TGX \geq 0$).

פונקציות גרעין שימושיות במיוחד:

גרעין ליניארי –

$$k(X_i, X_j) = X_i^T X_j + c$$

גרעין פולינומיאלי –

$$k(X_i, X_j) = \left(X_i^T X_j + c\right)^d$$

גרעין RBF –

$$k(X_i, X_j) = e^{-\gamma\left\|X_i - X_j\right\|^2}$$

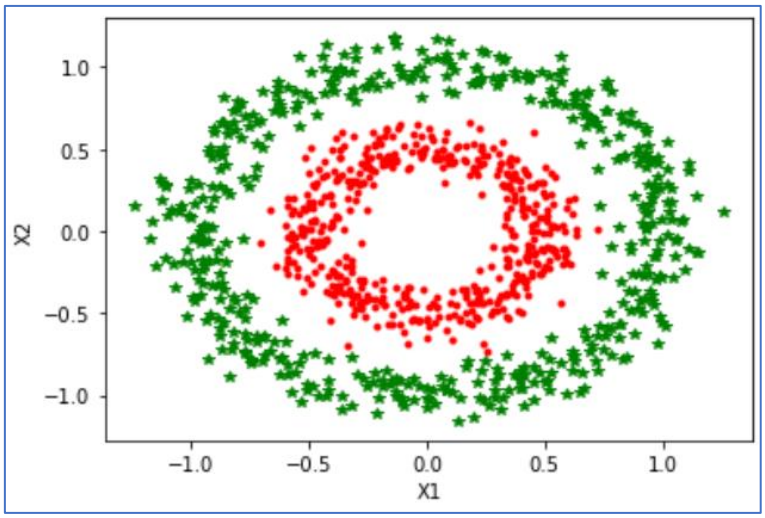
3.7 בחינת SVM על דאטה מוגרל

נבחן את אלגוריתם SVM על דאטה אשר מוגרל כשתי מחלקות הניתנות להפרדה על ידי מעגל שמרכזו בראשית הצירים. בפרויקט זה מימשתי את פתרון בעיית SVM באמצעות Batch Gradient Descent כפי שניתן לראות בפרק המימוש – קוד (כמות האיטרציות שבוצעו היא 3000). חלק זה כולל גם שימוש בהעתקה הבאה - $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \to \begin{pmatrix} x_1 \\ x_1^2 + x_2^2 \end{pmatrix}$ כדי להעביר את הבעיה לתחום בו המחלקות ניתנות להפרדה ליניארית. נציין כי ראוי היה לבצע את ההעתקה הבאה:

$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \to \begin{pmatrix} x_1 \\ x_2 \\ x_1^2 + x_2^2 \end{pmatrix}$, אך מתוך הרצון להישאר במימד 2 בוצעה "הטלה" על המישור (x_1, x_3) כדי להציג את ההפרדה בצורה נוחה. הדאטה הוגרל באופן שבו ישנן שתי מחלקות הניתנות להפרדה על ידי מעגל בראשית הצירים, 500 נקודות מכל מחלקה. בנוסף לכך, ניתן רעש לנקודות על מנת לאפשר למחלקות לא להיות מופרדות ליניארית לאחר העתקה באופן מושלם כדי לבחון את הרעיון העומד מאחורי *soft-margin*. בוצעה חלוקה של הדאטה למאגר אימון ומאגר בחינה ביחס של 3 :7 בהתאמה.

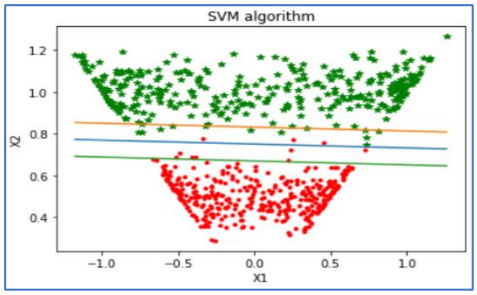
תוצאות ההרצה:

דאטה מקורי:

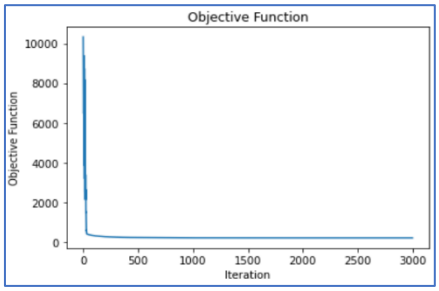


עבור C=15:

מאגר האימון וההיפר המישור המפריד:



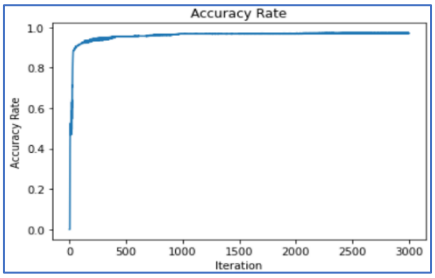
ערך פונקציית המטרה כפונקציה של מס' האיטרציות בתהליך הלמידה:



התוצאות עבור מאגר הבחינה:

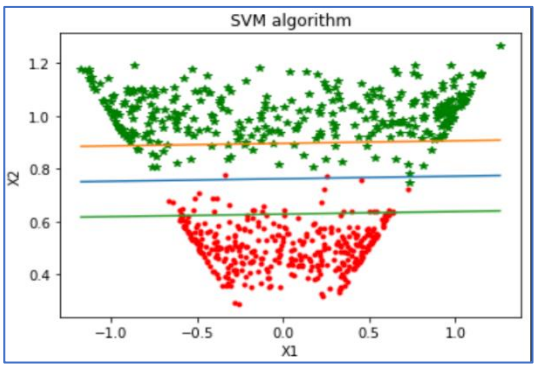
```
SVM's Circle Results:
The boundary decision line is: y=-0.02x+0.75
The weight vector is: [ -0.22987201 -12.34020096]
The bias balue is:  9.261428571428649

the confusion matrix is:
[[130   0 130]
 [  0 170 170]
 [130 170 300]]
the accuracy rate is:
100.000%
```

אחוז הדיוק כפונקציה של מס' האיטרציות בתהליך הלמידה:

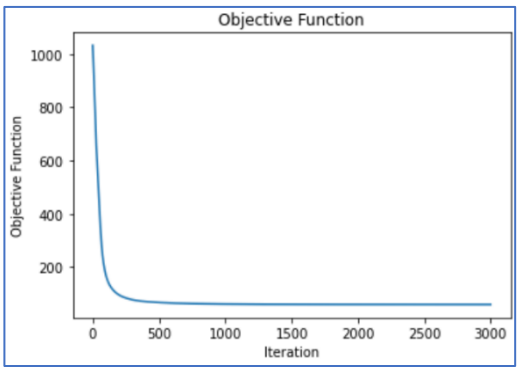


עבור C=1.5 :

מאגר האימון וההיפר המישור המפריד :



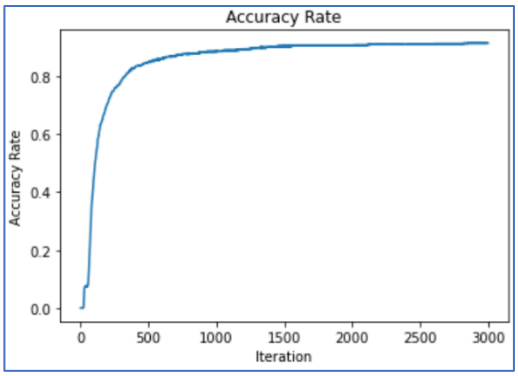
ערך פונקציית המטרה כפונקציה של מס' האיטרציות בתהליך הלמידה :



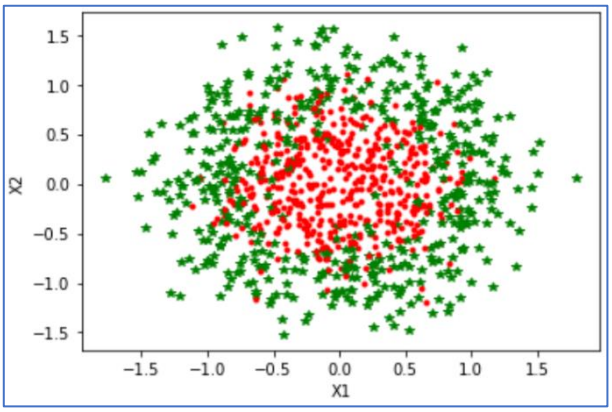
התוצאות עבור מאגר הבחינה :

```
SVM's Circle Results:
The boundary decision line is: y=0.01x+0.76
The weight vector is: [ 0.07207922 -7.46174161]
The bias balue is:  5.685428571428636

the confusion matrix is:
[[130   0 130]
 [  0 170 170]
 [130 170 300]]
the accuracy rate is:
100.000%
```

אחוז הדיוק כפונקציה של מס' האיטרציות בתהליך הלמידה :

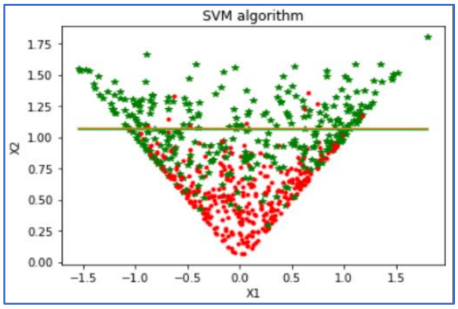


עבור דאטה מורעש פי 3 :

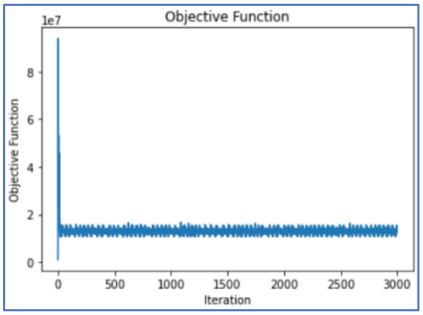


עבור C=15 :

מאגר האימון וההיפר המישור המפריד :



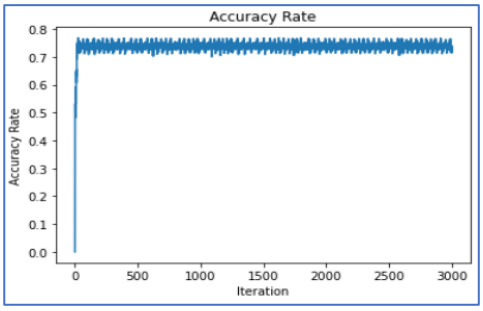
ערך פונקציית המטרה כפונקציה של מס' האיטרציות בתהליך הלמידה :



התוצאות עבור מאגר הבחינה :

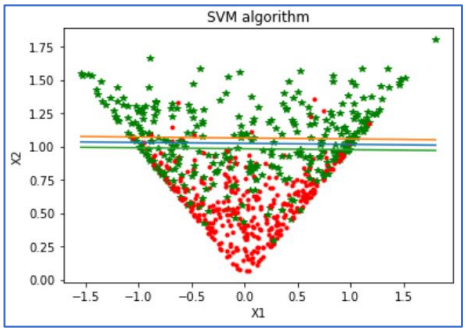
```
SVM's Circle Results:
The boundary decision line is: y=-0.00x+1.06
The weight vector is: [-1.50643052e-01 -2.14398156e+02]
The bias balue is:  228.29999999999814

the confusion matrix is:
[[125  93 218]
 [  5  77  82]
 [130 170 300]]
the accuracy rate is:
67.333%
```

אחוז הדיוק כפונקציה של מס' האיטרציות בתהליך הלמידה :

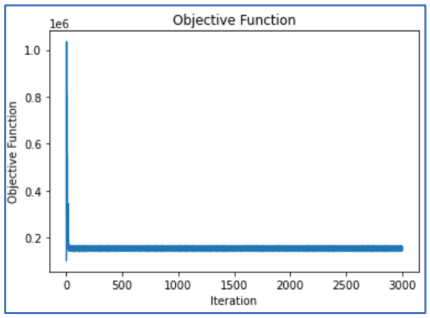


עבור C=1.5 :

מאגר האימון וההיפר המישור המפריד :



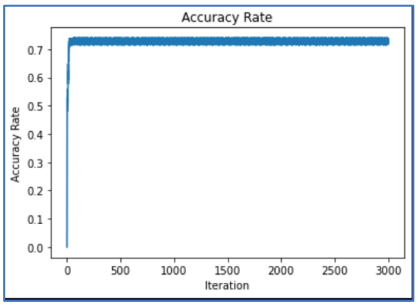
ערך פונקציית המטרה כפונקציה של מס' האיטרציות בתהליך הלמידה :



התוצאות עבור מאגר הבחינה :

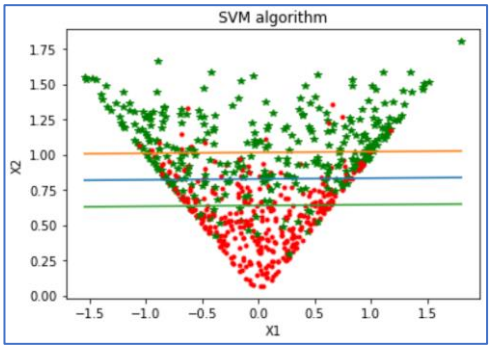
```
SVM's Circle Results:
The boundary decision line is: y=-0.01x+1.02
The weight vector is: [ -0.17763098 -24.77397495]
The bias balue is:  25.32857142857079

the confusion matrix is:
[[123  83 206]
 [  7  87  94]
 [130 170 300]]
the accuracy rate is:
70.000%
```

אחוז הדיוק כפונקציה של מס' האיטרציות בתהליך הלמידה :

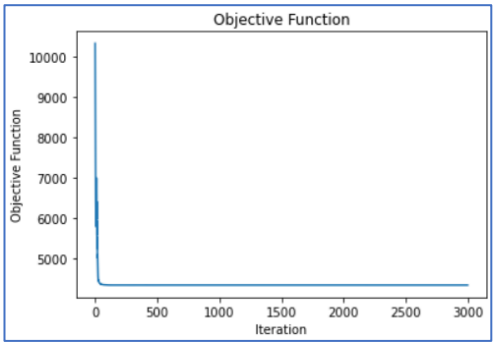


עבור C=0.15

מאגר האימון וההיפר המישור המפריד:



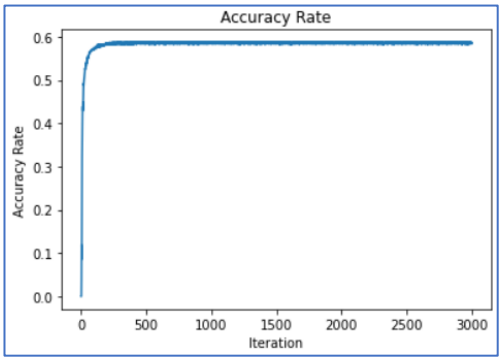
ערך פונקציית המטרה כפונקציה של מס' האיטרציות בתהליך הלמידה:



התוצאות עבור מאגר הבחינה:

```
SVM's Circle Results:
The boundary decision line is: y=0.01x+0.83
The weight vector is: [ 0.0309227  -5.31333573]
The bias balue is:  4.3885714285714315

the confusion matrix is:
[[113  30 143]
 [ 17 140 157]
 [130 170 300]]
the accuracy rate is:
84.333%
```

אחוז הדיוק כפונקציה של מס' האיטרציות בתהליך הלמידה:



ניתוח תוצאות:

1. ניתן לראות שככל שפרמטר הענישה גדול כך השוליים צרים וזאת מאחר שאנחנו מענישים את הפונקציה באופן משמעותי על כל טעות, כלומר מתייחסים בחומרה לכל טעות.
2. בכל המקרים קיבלנו כי פונקציית המטרה אשר נמצאת במגמת ירידה באופן כללי ואילו שיעור הדיוק נמצא בעלייה.
3. עבור הרצת האלגוריתם במקרה של דאטה מורעש באופן מועט, ניתן לראות כי שיעור הדיוק ופונקציית המטרה למול מספר האיטרציות יוצרות פונקציות חלקות יחסית. כמו גם, שינוי בפרמטר ההענשה לא הביא לתוצאות שונות כל כך ואף בשני המקרים התקבל אחוז דיוק של 100% עבור מאגר הבחינה.
4. הרצת האלגוריתם עבור דאטה מורעש פי 3 - ככל שהוקטן פרמטר הענשה הגענו לתוצאות טובות יותר בבחינת המודל. ניתן להסביר זאת מאחר שכאשר מגדילים את פרמטר הענשה אנו יוצרים מצב של התאמת יתר עבור מאגר הלמידה ואילו יתכן כי מאגר הבחינה שונה במעט ולכן התאמת היתר פוגעת באחוזי הדיוק. כמו גם, עבור פרמטר הענשה קטן יחסית 0.15 קיבלנו כי במהלך הלמידה שיעור הדיוק ופונקציית המטרה למול מספר האיטרציות הן פונקציות חלקות יותר ואילו עבור פרמטרי ההענשה הגדולים יותר קיבלנו פונקציות שאינן חלקות.

4 יישום לסיווג טקסט באמצעות אלגוריתם SVM

סיווג טקסט הוא טכניקה בתחום למידת מכונה (לרוב מסוג מונחית) המקצה קטגוריות מוגדרות מראש עבור טקסט פתוח. נעשה שימוש נרחב במסווגי טקסט על מנת לארגן מבנה ולקטלג כל סוג של טקסט - מטקסטים רפואיים ועד לפוסטים ברשתות החברתיות. סיווג טקסט הוא אחד מהמשימות הנפוצות והבסיסיות בתהליך עיבוד שפה טבעית עם יישומים רבים, כגון – ניתוח רגשות, תיוג נושאים, זיהוי ספאם וזיהוי כוונות. ישנם אלגוריתמים רבים ושונים לסיווג טקסט. אנחנו ננסה לבחון את אלגוריתם הסיווג עליו דיברנו SVM עבור סיווג טקסט.

4.1 אופן התהליך

1. ייצוג הטקסטים במבנה נתונים.
2. טיוב הנתונים – ביצוע מגוון פעולות על הטקסטים הנתונים שיעזרו לנו להפיק מידע נחוץ לטובת משימת הסיווג ולהפחתת מימדי הבעיה כדי לשפר זמן ריצה.
3. הפיכת קבצי הטקסט לווקטורים הכוללים מאפיינים נומריים.
4. חלוקת המבנה – חלוקה למאגר אימון ולמאגר בחינה.
5. הפעלת אלגוריתם סיווג SVM על מאגר האימון לטובת מציאת המסווג.
6. בחינת המודל – בחינת המודל לאחר מציאת המסווג באמצעות מאגר הבחינה.

4.2 תהליך טיוב הנתונים

- הסרת שורות ריקות.
- Lower Case – המרת כלל הטקסטים לאותיות קטנות.
  לדוגמא:

"Hello world! This is an example sentence" -> "hello world! this is an example sentence"

- Tokenization – המרת הטקסט לרשימה הכוללת כל מילה בתא ברשימה.
  לדוגמא:

"hello world! this is an example sentence" -> ['hello', 'world', '!', 'this', 'is', "n't", 'an', 'amazing', 'example', 'sentence', '.']

- Stop words and alphabet recognition – זיהוי מילים בעלות משמעות והסרת סימני הפיסוק.
  לדוגמא:

"hello world! this is an example sentence" -> ['hello', 'world', 'example', 'sentence']

- POS tag and lemmatized - זיהוי חלקי הדיבר של המילה (שם תואר, שם עצם או פועל) והוצאת שורש. חלק זה נועד ליצור מאפיינים בעלי משמעות ותורם להפחתת מימדי הבעיה שהרי שימוש בשורש "סופח" אל פיצ'ר זה את כל המילים השייכות לאותו

השורש. בנוסף לכך, חלק זה תורם בכלליות המודל (לא ספציפי למילים מסוימות). כמו כן, עוזר בזיהוי ההקשר של המילה במשפט.
לדוגמא:

"running" -> verb-> "run"
"better" -> adjective-> "good"

- בחירת K המילים התדירות ביותר. זה נעשה לצורך הקטנת המימדים של הבעיה. נציין כי K הוא מספר שרירותי הנתון לבחירה.
- יצירת ווקטורים באמצעות tfidf (על כך נפרט בהמשך).

4.3 ווקטוריזציה – יצירת ווקטורים מתוך טקסט

אלגוריתם SVM מושתת על ווקטורים המכילים מספרים. לכן, נרצה להעניק למילים ערכים נומריים על מנת שנוכל להפעיל את האלגוריתם. הפיכת המילים לערכים נומריים תיעשה בשיטת tf-idf. שיטה זו תעניק לכל מילה ערך נומרי המציין את חשיבות המילה בקבצי הטקסט הנתונים לנו.

- tf (Term Frequency) תדירות מונח – התדירות היחסית של מונח t בתוך מסמך d.
יחושב באופן הבא:

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

כאשר - $f_{t,d}$ מסמל את התדירות של המילה t במסמך d.
למעשה, מחשבים תדירות יחסית מתוך ההבנה שגדלי הטקסטים שונים.
ככל שמילה תופיע הרבה בטקסט מסוים ניתן להסיק שהיא בעלת משמעות רבה באותו טקסט.

- Idf (Inverse Document Frequency) תדירות מסמכים הופכית – מדד לכמות המידע שהמילה מספקת. ננתח את כמות הטקסטים בהם המילה t מופיע. ככל שמילה תופיע בהרבה טקסטים (כלומר נפוצה) אז כנראה שחשיבותה קטנה. לעומת זאת, נרצה להגדיל את משקלן של מילים אשר מופיעות לעיתים רחוקות.
יחושב באופן הבא:

$$idf(t,D) = \ln\left(\frac{|D|}{|\{d \in D: t \in d\}|}\right)$$

D - מייצג את אוסף כל הטקסטים/מסמכים. $|D|$ - כמות הטקסטים/מסמכים.
נשים לב שמילה המופיעה בכל המסמכים תקבל - $idf(t,D) = \ln\left(\frac{|D|}{|D|}\right) = \log(1) = 0$.
תיקונים עבור Idf (smoothing idf):
כדי למנוע חלוקה באפס, ישנן שיטות לתיקון הביטוי המתמטי עבור idf. חלוקה באפס

תתרחש כאשר ישנו ביטוי אשר לא מופיע כלל בקובץ.
התיקון הבסיסי:

$$idf(t,D) = \ln\left(\frac{|D|+1}{|\{d \in D: t \in d\}|+1}\right)$$

משמעות התיקון – יצירת טקסט חדש בתוך הקובץ המכיל את כלל הטקסטים, כך שהטקסט החדש מכיל כביכול את כל המילים שנבחרו לטובת הווקטוריזציה.
התיקון הנפוץ:

$$idf(t,D) = \ln\left(\frac{|D|+1}{|\{d \in D: t \in d\}|+1}\right)+1$$

למעט העובדה שתיקון זה מונע חלוקה באפס, מטרתו הנוספת היא לתת ערך נומרי שאינו אפס למילים שמופיעות בכל הטקסטים הנכללים בקובץ. שהרי אם מילה מופיע בכל הטקסטים אזי לפי התיקון הקודם נקבל ערך נומרי אפס עבור אותה מילה, אך מאחר שאנו רוצים לתת משמעות גם למילים כאלו נבקש לתקן זאת על ידי התיקון הנפוץ. ביישום פרק זה, נשתמש בתיקון הנפוץ.

- *Tf-idf* – מדד זה לוקח בחשבון את תדירות המונח במסמך מסוים ואת תדירות המסמכים ההופכית. ערך זה יגדל כאשר מילה מופיעה מספר רב של פעמים במסמך בודד, אך יקטן אילו מילה זה תופיע במספר רב של מסמכים.
יחושב באופן הבא:

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D)$$

- בניית הווקטור – נניח שנבחרו n מילים $(t_0, \dots t_{n-1})$ לבניית הווקטורים, אזי עבור כל מסמך נבנה ווקטור בעל n קורדינטות. לדוגמא, עבור מסמך טקסט d מתוך הקובץ D נבנה ווקטור בעל n כך שהקורדינטה ה- k שלו מייצגת את הערך $tfidf(t_k,d,D)$, כלומר הווקטור המתאים למסמך d ייראה כך:

$$\overrightarrow{V_d} = \begin{pmatrix} tfidf(t_0,d,D) \\ \vdots \\ tfidf(t_n,d,D) \end{pmatrix}$$

לאחר מכן, ננרמל (לפי נורמה 2) כל ווקטור,

$$\widehat{V_d} = \frac{1}{\sqrt{\sum_{i=1}^{n}\left(tfidf(t_i,d,D)\right)^2}} \cdot \begin{pmatrix} tfidf(t_0,d,D) \\ \vdots \\ fidf(t_n,d,D) \end{pmatrix}$$

כלומר, $\left\|\widehat{V_d}\right\|_2 = 1$.

4.3.1 דוגמא עבור תהליך טיוב ו-ווקטוריזציה:

1. נתון קובץ המכיל שני טקסטים:

```
Text 1: Applied mathematics contributes to the development of machine
learning and running better algorithms
Text 2: Optimization is a key area of applied mathematics and making
processes more efficient
```

2. Tokenization:

```
Tokenization 1: ['applied', 'mathematics', 'contributes', 'to', 'the',
'development', 'of', 'machine', 'learning', 'and', 'running', 'better',
'algorithms']
Tokenization 2: ['optimization', 'is', 'a', 'key', 'area', 'of',
'applied', 'mathematics', 'and', 'making', 'processes', 'more',
'efficient']
```

3. הסרת Stop Words:

```
Filtered 1: ['applied', 'mathematics', 'contributes', 'development',
'machine', 'learning', 'running', 'better', 'algorithms']
Filtered 2: ['optimization', 'key', 'area', 'applied', 'mathematics',
'making', 'processes', 'efficient']
```

4. POS tag and lemmatized:

```
Lemmatization 1: ['apply', 'mathematics', 'contributes', 'development',
'machine', 'learn', 'run', 'well', 'algorithm']
Lemmatization 2: ['optimization', 'key', 'area', 'apply', 'mathematics',
'make', 'process', 'efficient']

POS Tags 1: [('apply', 'VB'), ('mathematics', 'NNS'), ('contributes',
'NNS'), ('development', 'NN'), ('machine', 'NN'), ('learn', 'NN'),
('run', 'VBP'), ('well', 'RB'), ('algorithm', 'RB')]
POS Tags 2: [('optimization', 'NN'), ('key', 'JJ'), ('area', 'NN'),
('apply', 'NN'), ('mathematics', 'NNS'), ('make', 'VBP'), ('process',
'NN'), ('efficient', 'NN')]
```

5. המילים שנבחרו לבניית הווקטורים:

```
Feature Names (Words):
['algorithm', 'apply', 'area', 'contributes', 'development',
'efficient', 'key', 'learn', 'machine', 'make', 'mathematics',
'optimization', 'process', 'run', 'well']
```

6. הטקסטים לאחר טיוב:

```
Text 1 after preprocessing: apply mathematics contributes development
machine learn run well algorithm
Text 2 after preprocessing: optimization key area apply mathematics make
process efficient
```

7. ווקטוריזציה:

```
TF-IDF Vectors:

TFIDF vec 1: [0.35327777 0.25136004 0.         0.35327777 0.35327777 0.
 0.         0.35327777 0.35327777 0.         0.25136004 0.
 0.         0.35327777 0.35327777]

TFIDF vec 2: [0.         0.26868528 0.37762778 0.         0.
0.37762778
 0.37762778 0.         0.         0.37762778 0.26868528 0.37762778
 0.37762778 0.         0.        ]
```

תהליך הווקטוריזציה (סעיף 7 בדוגמא) באופן ידני עבור טקסט 1:

<u>עבור הטקסט הראשון:</u>

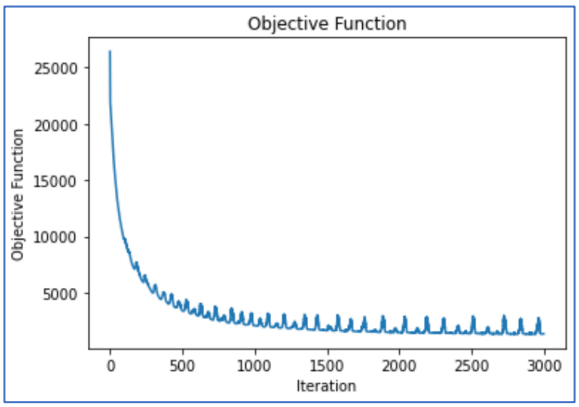
Feature Name	tf	IDF	Tf-idf (vec)	ערכים לאחר נרמול
t_0 = algorithm	$\frac{1}{9}$	$\ln\left(\frac{2+1}{1+1}\right)+1$	$\frac{1}{9}\left(\ln\left(\frac{2+1}{1+1}\right)+1\right)=0.156$	0.353278
t_1 = apply	$\frac{1}{9}$	$\ln\left(\frac{2+1}{2+1}\right)+1$	$\frac{1}{9}\left(\ln\left(\frac{2+1}{2+1}\right)+1\right)=\frac{1}{9}$	0.25136
t_2 = area	$\frac{0}{9}$	$\ln\left(\frac{2+1}{1+1}\right)+1$	$\frac{0}{9}\left(\ln\left(\frac{2+1}{1+1}\right)+1\right)=0$	0
t_3 = contributes	$\frac{1}{9}$	$\ln\left(\frac{2+1}{1+1}\right)+1$	$\frac{1}{9}\left(\log\left(\frac{2+1}{1+1}\right)+1\right)=0.156$	0.353278
t_4 = development	$\frac{1}{9}$	$\ln\left(\frac{2+1}{1+1}\right)+1$	$\frac{1}{9}\left(\log\left(\frac{2+1}{1+1}\right)+1\right)=0.156$	0.353278
t_5 = efficient	$\frac{0}{9}$	$\ln\left(\frac{2+1}{1+1}\right)+1$	$\frac{0}{9}\left(\log\left(\frac{2+1}{1+1}\right)+1\right)=0$	0
t_6 = key	$\frac{0}{9}$	$\ln\left(\frac{2+1}{1+1}\right)+1$	$\frac{0}{9}\left(\log\left(\frac{2+1}{1+1}\right)+1\right)=0$	0
t_7 = learn	$\frac{1}{9}$	$\ln\left(\frac{2+1}{1+1}\right)+1$	$\frac{1}{9}\left(\log\left(\frac{2+1}{1+1}\right)+1\right)=0.156$	0.353278
t_8 = machine	$\frac{1}{9}$	$\ln\left(\frac{2+1}{1+1}\right)+1$	$\frac{1}{9}\left(\log\left(\frac{2+1}{1+1}\right)+1\right)=0.156$	0.353278
t_9 = make	$\frac{0}{9}$	$\ln\left(\frac{2+1}{1+1}\right)+1$	$\frac{0}{9}\left(\log\left(\frac{2+1}{1+1}\right)+1\right)=0$	0
t_{10} = mahtematic	$\frac{1}{9}$	$\ln\left(\frac{2+1}{2+1}\right)+1$	$\frac{1}{9}\left(\log\left(\frac{2+1}{2+1}\right)+1\right)=\frac{1}{9}$	0.25136
t_{11} = optimization	$\frac{0}{9}$	$\ln\left(\frac{2+1}{1+1}\right)+1$	$\frac{0}{9}\left(\log\left(\frac{2+1}{1+1}\right)+1\right)=0$	0
t_{12} = process	$\frac{0}{9}$	$\ln\left(\frac{2+1}{1+1}\right)+1$	$\frac{0}{9}\left(\log\left(\frac{2+1}{1+1}\right)+1\right)=0$	0
t_{13} = run	$\frac{1}{9}$	$\ln\left(\frac{2+1}{1+1}\right)+1$	$\frac{1}{9}\left(\log\left(\frac{2+1}{1+1}\right)+1\right)=0.156$	0.353278
t_{14} = well	$\frac{1}{9}$	$\ln\left(\frac{2+1}{1+1}\right)+1$	$\frac{1}{9}\left(\log\left(\frac{2+1}{1+1}\right)+1\right)=0.156$	0.353278

4.4 יישום של SVM עבור סיווג טקסט

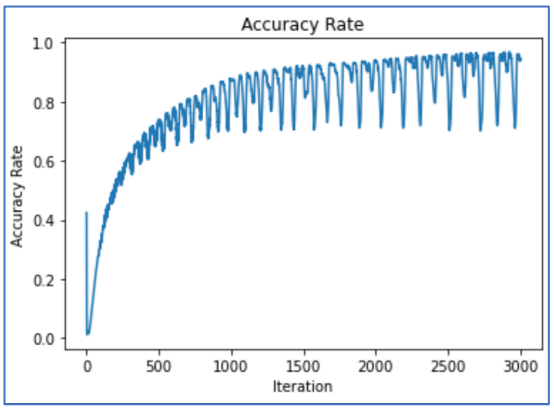
נבחן את אלגוריתם SVM אותו מימשנו עבור סיווג טקסט. ניקח מאגר ציוצים מהרשת החברתית Twitter אשר מסווגים לציוצים גזעניים ולא גזעניים. ננסה ללמד את המחשב באמצעות אלגוריתם SVM לזהות איזה ציוץ הוא בעל אופי גזעני ואיזה ציוץ לא בעל אופן גזעני. הדאטה כולל 2000 ציוצים מה- Twitter אשר מתוכם 1000 ציוצים בעלי אופן גזעני והשאר לא (מצורף בפרק נספחים ורשימת מקורות). נבצע טיוב לנתונים ונבצע ווקטוריזציה לטקסט בהתאם לשיטה tf-idf תוך שימוש בחבילות (המצוינות בפרק מימוש הקוד) השייכות לספריה ntlk. כל ווקטור יורכב מ-2865 מילים. לאחר מכן, נריץ את אלגוריתם SVM אשר אותו מימשנו. במקביל, נריץ את אלגוריתם SVM המובנה ב-python מתוך הספריה sklearn, וננסה להשוות בין התוצאות. שוב, החלוקה הייתה 3 :7 לטובת מאגר האימון.

<u>עבור C=15 :</u>

ערך פונקציית המטרה כפונקציה של מס' האטרציות בתהליך האימון :



אחוז הדיוק כפונקציה של מס' האיטרציות בתהליך האימון :



תוצאות הרצת פתרון המודל על מאגר הבחינה

```
SVM Accuracy by python function:  86.0 %
The weight vector is: [ 0.09092558  1.05210386  2.38886087 ...  4.37262029
1.50083818
 -0.66028833]
The bias balue is:  -0.9759113652609007

the confusion matrix is:
[[259  43 302]
 [ 35 263 298]
 [294 306 600]]
the accuracy rate is:
87.000%
```

ניתוח תוצאות:

1. ניתן לראות כי אלגוריתם SVM שימושי גם עבור סיווג טקסט. קיבלנו תוצאת דיוק גבוהה עבור מאגר הבחינה. כמו גם, ניתן לראות כי תוצאת הדיוק עבור *SVM* אותו מימשנו למול הפונקציה המובנית בספריה לא בעלת שגיאה גדולה. כדי לאשש זאת, היה ניתן לבצע מספר הרצות ולבדוק את השגיאה ולחשב את תוחלת השגיאות.
2. פונקציית המטרה למול מספר האיטרציות – ניתן לראות כי לא קיבלנו גרף חלק, יחד עם זאת המגמה היא ירידה כמו שהיינו מצפים שהרי אלגוריתם *SVM* במימוש *Batch Gradient Descent* מטרתו למצוא את המינימום של הפונקציה.
3. שיעור הדיוק במהלך הלמידה – ניתן לראות שגם גרף שיעור הדיוק למול מספר האיטרציות לא חלק במיוחד, אך גם בסיווג טקסט ניתן לראות כי המגמה היא עלייה כפי שהיינו מצפים שהרי ברצוננו לשפר את דיוק הסיווג.
4. השוואה מול הפונקציה המובנית בפייתון – בפלט של תוצאות מאגר הבחינה ניתן לראות גם את אחוז הדיוק שהתקבל עבור הרצת הפונקציה המובנית בפייתון. נשים לב, כי הפונקציה המובנית קיבלה אחוז דיוק של 86% ואילו אחוז הדיוק של האלגוריתם שלנו (הממומש) הוא 87%. נציין כי הרצנו את שתי הפונקציות עבור אותו פרמטר ההענשה *C=15*. אין להסיק מכך שמימוש האלגוריתם שלנו טוב יותר שהרי עבור הרצות נוספות ייתכן כי נקבל תוצאות שונות. יחד עם זאת, ההפרש באחוז הדיוק במימוש האלגוריתם למול הפונקציה המובנית לא היה משמעותי גם עבור הרצות נוספות שביצענו (אם יתרון קל לאחד מהם).

5 סיכום

בפרויקט זה, עסקנו באחת הבעיות החשובות מעולם למידת מכונה – Machine Learning והיא בעיית הסיווג. לבעיה זו המצויה בתחומים רבים, קיימות גישות שונות לפתרון ובפרויקט זה בחנו את גישת האלגוריתמים Perceptron ו-SVM. תחילה סקרנו את בעיית הסיווג הליניארי תוך שימוש באלגוריתמים הנתונים. לאחר מכן, הרחבנו את בעיית הסיווג הליניארי לבעיית סיווג כללית תוך שימוש באותם האלגוריתמים, אך עם ביצוע מניפולציות על המרחב על ידי העתקה למרחב שבו הנתונים ניתנים להפרדה ליניארית. התייחסנו לגרסאות השונות של אלגוריתם SVM ותרומתן למניעת תופעת התאמת היתר (Overfitting) ולפתרון בעיית סיווג עבור דאטה מורעש. בנוסף לכך, מוצגים מימושים לאלגוריתמים בשפת Python ואף בחינת מימושים אלו על מערכי נתונים שונים. כמו כן, הפרויקט מציג את אופן השימוש באלגוריתם SVM לטובת יישום עבור בעיית סיווג טקסט ומכיל בתוכו את האסטרטגיה העומדת מאחורי הפיכת הטקסט לווקטור בעל רכיבים ממשיים. עיקר הפרויקט מתעסק בתיאוריה המתמטית העומדת מאחורי בעיות הסיווג ופתרונן באמצעות האלגוריתמים הללו. התוצאות שהתקבלו בבחינת האלגוריתמים על מערכי הדאטה השונים ביססו את העובדה שהאלגוריתמים האלה מהווים שיטה לפתירת בעיות סיווג שונות הדגימו את יישום התיאוריה המתמטית. אחת המטרות העומדות מאחורי הפרויקט הנ"ל הייתה להציג את האופן בו תחומים רבים בעולם המתמטי (כגון – אלגברה, חדו"א, אופטימיזציה ועוד) משתלבים בנושא רלוונטי מתמיד כמו למידת מכונה. מטרתו העיקרית של ענף המתמטיקה השימושית היא ליישם תיאוריות מתמטיות ולתת בסיס תיאורטי לטובת יישומים אלו, וכך גם הייתה מטרתו של הפרויקט. כמחקר המשך אמליץ לחקור את זמן הריצה עבור פתרון בעיית SVM באמצעות אלגוריתמים שונים כגון:

Gradient descent, Stochastic Gradient Descent, Pegasos Algorithm, SMO.

כמו כן, אמליץ לבחון יישומים שונים של האלגוריתם SVM, למשל עבור סיווג שמע תוך התייחסות לאופן הפיכת השמע לווקטור בעל רכיבים ממשיים.

6 הכרת תודה

ראשית, ארצה להודות רבות לפרופסור חגי כתריאל שהנחה וליווה אותי לאורך כל הפרויקט, תודה על שהקדיש מזמנו היקר להכווין אותי ולדייק את הפרויקט כך שאהיה מרוצה ושלם מהתוצר הסופי.
שנית, ארצה להודות לכלל סגל המרצים מהם למדתי לאורך כל התואר, אין ספק שהידע ויכולת הלמידה שרכשתי מסגל המרצים מהווים אבני דרך לגיבוש התוצר הסופי של פרויקט זה.

7 רשימת מקורות

[1] Bherde, G., Telang, A., Bhuva, M., Gajra, H., & Patel, A. (2015). Credit Card Fraud Detection Using Perceptron Training Algorithm and Prevention Using One Time Password. *Recent and Innovation Trend in Computing and Communication*, *Volume: 3* (Issue: 4).

[2] Sarang Narkhede (May 2018). *Understanding Confusion Matrix. Medium.*

[3] Flach, P. (2012). *Machine Learning: The Art and Science of Algorithms that Make Sense of Data (pp.207-216).* Cambridge University Press.

[4] Hardt, M., & Recht, B. (2022). *Patterns, Predictions, and Actions: A Story About Machine Learning (pp.37-42).*

[5] Akshay L Chandra (Aug 22, 2018). *Perceptron Learning Algorithm: A Graphical Explanation Of Why it Works.* Medium.

[6] Vivek Srikumar (Spring 2018). *Machine Learning.* The University Of Utah.

[7] Avrim Blum, (January 25,2010). *15-859(B) Machine Learning Theory.*

[8] Andrew Ng. *CS229 Lecture notes.* Stanford University.

[9] Sergio Rem´ırez Miqu´elez (July 2023). *Support Vector Machines for Binary Classification: Theory and Practice.* GRADO EN MATEMATICAS.

[10] Thomas, G. (2018, May 29). *Convexity.* Stanford University.

[11] Vivek Srikumar (Spring 2018). *SVM: Training with Stochastic Gradient Descent.* The University Of Utah.

[12] Emil Westin (Spring 2020). *Authorship Classification Using The Vector Space Model and Kernel Methods.*

[13] Mukesh Chaudhary (Apr 24, 2020). *TF-IDF Vectorizer scikit-learn. Medium.*

קישורים לדאטה:

1. https://www.kaggle.com/datasets/rishabhjohri/racismdetectiondataset
2. https://www.kaggle.com/datasets/uciml/iris

8 מימוש הקוד

1. ספריות לשימוש:

```
#libraries:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets.samples_generator import make_blobs
from sklearn.datasets.samples_generator import make_circles
from sklearn import svm
from sklearn.model_selection import train_test_split
import nltk
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, svm
from sklearn.metrics import accuracy_score
import random
```

2. יצירת ווקטורים מהדאטה באקסל:

```
#this func reduce the data frame according to the classes we want to examine
def reduce_data_frame(df,Class_col_name,Positive_Class, Negative_Class):
    df_new = df
    rows = df_new.shape[0]
    Y_vec = df_new[Class_col_name]
    counter_positive = 0
    counter_negative = 0
    for i in range(0,rows):
        if Y_vec[i] == Positive_Class:
            counter_positive+=1
        elif Y_vec[i] == Negative_Class:
            counter_negative+=1
            df_new[Class_col_name][i] = -1 #change the negative class to (-1)
        else:
            df_new.drop(index = i, axis = 0, inplace = True)
    return df_new,counter_positive,counter_negative

#real data:
def get_data():
    df_name = str(input("DataBase name: "));  #for instance: iris.xlsx
    df = pd.read_excel(df_name)
    features_name = []
    num_of_features = int(input("How many features? "))
    class_coumn_name = str(input("What is the name of the class column? "))
    Positive_Class = int(input("What is the number of the positive class? "))
    Negative_Class = int(input("What is the number of the negative class? "))
    df_new,counter_positive,counter_negative = reduce_data_frame(df,class_coumn_name,Positive_Class, Negative_Class)
    X = np.zeros((len(df_new),num_of_features))
    Y = np.zeros(len(df_new))
    Y = df_new[class_coumn_name]
    for i  in range(0,num_of_features):
```

```
    parameter_name = str(input("Featrue name: "))
    features_name.append(parameter_name)
  print("")

  X = np.array(X)
  Y = np.array(Y)

  for i in range(0,num_of_features):
    X[:,i] = df_new[features_name[i]] #get the X vector from data

  return X,Y,num_of_features,features_name,Positive_Class,Negative_Class
```

3. יצירת Scatter Plot של מאגר האימון:

```
def plot_X_Y(X,Y):
  length_of_Data = len(Y);
  for i in range(0,length_of_Data):
    x1 = X[i][0];
    x2 = X[i][1];
    if Y[i] == -1:
      plt.plot(x1,x2,'g*');
    elif Y[i] == 1:
      plt.plot(x1,x2,'r.');
    #else:
    #   plt.plot(x1,x2,'b.');
  plt.xlabel("X1");
  plt.ylabel("X2");
```

4. שרטוט ההיפר-מישור המפריד על ידי אלגוריתם Perceptron:

```
def plot_HyperPlane_boundary(W,bias,num_features,X,Y):
  if num_features==2:
      min_x = X[:,0].min();
      max_x = X[:,0].max();
      x_plot = np.array([min_x,max_x]);
      y_plot = -bias/W[1] + -W[0]/W[1]*x_plot;
      plt.plot(x_plot,y_plot); #plot the boundary decision line
      plt.title("Perceptron algorithm")
      a = -W[0]/W[1];
      b = -bias/W[1];
      if b>0:
        print("The boundary decision line is: y=%.2fx+%.2f " %(a,b));
      else:
        print("The boundary decision line is: y=%.2fx%.2f " %(a,b));

  print("The weight vector is:",W);
  print("The bias balue is: ", bias);
  print("");
```

5. מימוש אלגוריתם Peceptron:

```
def Perceptron_algo(X,Y,num_features,Class0,Class1,W,bias):
  #initial W and bias: W = zero vec, bias = 0
  error_counter = 0
  err_vec = np.zeros(Y.shape[0])
  for i in range(0,Y.shape[0]):
    dot_product = W@X[i]+bias;  #dot product (w^T)*X , w = (bias w1 w2 ....) X = (1 x1 x2 ....)
    if Y[i]*dot_product<=0: #mistake while Yi*(W@X[i]+bias)<=0
      W = W + Y[i]*X[i];
      bias+=1;
```

```
        error_counter+=1 #update while there is an error
    err_vec[i] = error_counter
  return W,bias,error_counter,err_vec;

def convergence_pereptron(X,Y,num_features,Class0,Class1,W,bias):
  bias = 0
  errors_amount = 1;
  Total_err_vec = np.zeros(1)
  while errors_amount > 0:
    W,bias,errors_amount,err_vec = Perceptron_algo(X,Y,num_features,Class0,Class1,W,bias);
    Total_err_vec = np.concatenate((Total_err_vec,err_vec))
  plot_HyperPlane_boundary(W,bias,num_features,X,Y); #plot the hyperplane boundary
  plt.show()
  plt.plot(Total_err_vec)
  plt.title("Graph of accumulated errors")
  plt.xlabel("Iteration");
  plt.ylabel("Accumulated Errors");
  plt.show();
  return W,bias
```

6. שרטוט ההיפר-מישור המפריד על ידי אלגוריתם SVM:

```
#SVM part:
def plot_HyperPlane_boundary_SVM(W,bias,num_features,X,Y):
  if num_features==2:
        min_x = X[:,0].min();
        max_x = X[:,0].max();
        x_plot = np.array([min_x,max_x]);
        y_plot = -bias/W[1] + -W[0]/W[1]*x_plot;
        plt.plot(x_plot,y_plot); #plot the boundary decision line

        #upper bound:
        x_plot = np.array([min_x,max_x]);
        y_plot = (-bias-1)/W[1] - W[0]/W[1]*x_plot;
        plt.plot(x_plot,y_plot); #plot the boundary decision line

        #lower bound:
        x_plot = np.array([min_x,max_x]);
        y_plot = (-bias+1)/W[1] - W[0]/W[1]*x_plot;
        plt.plot(x_plot,y_plot); #plot the boundary decision line
        plt.title("SVM algorithm")
        a = -W[0]/W[1];
        b = -bias/W[1];
        if b>0:
          print("The boundary decision line is: y=%.2fx+%.2f " %(a,b));
        else:
          print("The boundary decision line is: y=%.2fx%.2f " %(a,b));

  print("The weight vector is:",W);
  print("The bias balue is: ", bias);
  print("");
```

7. מימוש אלגוריתם SVM באמצעות Batch Gradient-Decent:

```
def SVM_algo_GD(X,Y,num_features,Class0,Class1,W,bias):
  #initial W and bias: W = zero vec, bias = 0
  C = 0.15
  iterations = 3000
```

```
    lr = 1/5/Y.shape[0] #learning rate
    error_counter = 0
    err_vec = np.zeros(iterations)
    epochs = np.zeros(iterations)
    objective_function = np.zeros(iterations)
    obj = 0
    for j in range(0,iterations):
        sigma_W = np.zeros(num_features);
        sigma_b = 0;
        #compute the hinge function dervative for misclassified samples:
        for i in range(0,Y.shape[0]):
            dot_product = W@X[i]+bias;  #dot product (w^T)*X , w = (bias w1 w2 ....) X = (1 x1 x2 ....)
            if Y[i]*dot_product<=1: #max(0,1-y_i(W@X_i+bias))=1-y_i(W@X_i+bias)
                sigma_W = sigma_W + (-Y[i]*X[i])
                sigma_b = sigma_b + (-Y[i])
                error_counter += 1
                obj += 1-Y[i]*(W@X[i]+bias)
        objective_function[j] = 0.5*np.dot(W,W) + C*obj
        epochs[j] = j
        err_vec[j] = (Y.shape[0]-error_counter)/Y.shape[0]
        dJ_dW = (W +C*sigma_W)
        dJ_db = C*sigma_b
        W = W - lr*dJ_dW
        bias = bias - lr*dJ_db
        error_counter = 0
        obj = 0
    return W,bias,error_counter,epochs,objective_function,err_vec


def convergence_SVM(X,Y,num_features,Class0,Class1,W,bias):
    bias = 0.3
    errors_amount = 1
    t=1
    while t < 2:
        W,bias,errors_amoun,epochs,objective_function,err_vec = SVM_algo_GD(X,Y,num_features,Class0,Class1,W,bias);
        t += 1
    plot_HyperPlane_boundary_SVM(W,bias,num_features,X,Y); #plot the hyperplane boundary
    plt.show();
    plt.plot(epochs,objective_function)
    plt.title("Objective Function")
    plt.xlabel("Iteration");
    plt.ylabel("Objective Function");
    plt.show();
    plt.plot(epochs,err_vec)
    plt.title("Accuracy Rate")
    plt.xlabel("Iteration");
    plt.ylabel("Accuracy Rate");
    plt.show();
    return W,bias
```

8. יצירת כלל החלטה:

```
def decision_rule(W,bias,X,Y):
    dot_product = W@X+bias;
    if dot_product>=0:
        predicted_val = 1
    else:
        predicted_val = -1
    return predicted_val
```

9. יצירת תוצאות הרצת המודל על מאגר הבחינה (מטריצת ערפול):

```
def TestTheModel(W,bias,X,Y):
    confusion_matrix = np.array([[0,0,0],[0,0,0],[0,0,0]])
    for i in range(Y.shape[0]):
        confusion_matrix[2][2]+=1
        predicted_val = decision_rule(W,bias,X[i],Y)
        if predicted_val == 1 and Y[i] == 1:
            confusion_matrix[0][0]+=1  #a[pos][pos]+=1
            confusion_matrix[0][2]+=1  #for predicted val
            confusion_matrix[2][0]+=1  #for real val
        elif predicted_val == 1 and Y[i] == -1:
            confusion_matrix[0][1]+=1  #a[pos][neg]+=1
            confusion_matrix[0][2]+=1  #for predicted val
            confusion_matrix[2][1]+=1  #for real val
        elif predicted_val == -1 and Y[i] == 1:
            confusion_matrix[1][0]+=1  #a[neg][pos]+=1
            confusion_matrix[1][2]+=1  #for predicted val
            confusion_matrix[2][0]+=1  #for real val
        elif predicted_val == -1 and Y[i] == -1:
            confusion_matrix[1][1]+=1  #a[neg][neg]+=1
            confusion_matrix[1][2]+=1  #for predicted val
            confusion_matrix[2][1]+=1  #for real val
    success_rate = (confusion_matrix[0][0]+confusion_matrix[1][1])/confusion_matrix[2][2]
    return confusion_matrix, success_rate
```

10. העתקת מעגל למרחב שבו הדאטה ניתנת להפרדה ליניארית:

```
def circle_map(X,Y):
    x3 = np.zeros((X.shape[0],1))
    X = np.append(X,x3,axis = 1)
    for i in range(0,X.shape[0]):
        X[i][2] = ((X[i][0])**2+(X[i][1])**2)**0.5 #x3 = (x1^2+x2^2)^0.5 -> radius of the circle
    X = np.delete(X,1,axis = 1)
    return X,Y
```

11. טיוב נתונים עבור טקסט:

```
#text classification part:
def preprocessing(df):
    # 1 - Remove blank rows if any:
    df = df.dropna()
    df.reset_index(inplace=True, drop=True)

    # 2 - Change all the text to lower case:
    df['text'] = [entry.lower() for entry in df['text']]

    # 3 - Tokenization : turn the text to words (string) list:
    df['text']= [word_tokenize(entry) for entry in df['text']]

    # 4 - Remove Stop words, Non-Numeric and perfom Word Stemming/Lemmenting:

    # WordNetLemmatizer requires Pos tags to understand if the word is noun or verb or adjective etc.
    tag_map = defaultdict(lambda : wn.NOUN)
    tag_map['J'] = wn.ADJ
    tag_map['V'] = wn.VERB
    tag_map['R'] = wn.ADV
```

```
for index,entry in enumerate(df['text']):
#Declaring Empty List to store the words that follow the rules for this step
    Final_words = []
    # Initializing WordNetLemmatizer()
    word_Lemmatized = WordNetLemmatizer()
    # pos_tag function below will provide the 'tag' i.e if the word is Noun(N) or Verb(V) or something else.
    for word, tag in pos_tag(entry):
        # Below condition is to check for Stop words and if it include only alphabet letters
        if word not in stopwords.words('english') and word.isalpha():
            word_Final = word_Lemmatized.lemmatize(word,tag_map[tag[0]])
            Final_words.append(word_Final)
    # The final processed set of words for each iteration will be stored in 'text_final'
    df.loc[index,'text_final'] = str(Final_words)
return df
```

12. סיווג טקסט באמצעות המימוש שביצענו ל-SVM:

```
def TextClassification():
    #get data:
    df_origin = pd.read_excel("racismdetection.xlsx")

    #preprocessing:
    df = preprocessing(df_origin)

    Train_X, Test_X, Train_Y, Test_Y = model_selection.train_test_split(df['text_final'],df['Label'],test_size=0.3)
    Encoder = LabelEncoder()
    Train_Y = Encoder.fit_transform(Train_Y)
    Test_Y = Encoder.fit_transform(Test_Y)

    Tfidf_vect = TfidfVectorizer(max_features=5000)
    Tfidf_vect.fit(df['text_final'])
    Train_X_Tfidf = Tfidf_vect.transform(Train_X)
    Test_X_Tfidf = Tfidf_vect.transform(Test_X)

    #convert from sparse matrix to numpy arrays:    # Convert sparse TF-IDF matrices to dense arrays
    Train_X_Dense = Train_X_Tfidf.toarray()
    Test_X_Dense = Test_X_Tfidf.toarray()

    """
    #print(Tfidf_vect.vocabulary_)
    print("Hi")
    print("TF-IDF matrix: ", Train_X_Dense)
    """

    # Classifier - Algorithm - SVM
    # fit the training dataset on the classifier
    SVM = svm.SVC(C=1.5, kernel='linear', degree=1, gamma='auto')
    SVM.fit(Train_X_Tfidf,Train_Y)
    # predict the labels on validation dataset
    predictions_SVM = SVM.predict(Test_X_Tfidf)
    # Use accuracy_score function to get the accuracy
    print("SVM Accuracy by python function: ",accuracy_score(predictions_SVM, Test_Y)*100,"%")


    Train_Y[Train_Y == 0] = -1
    Test_Y[Test_Y == 0] = -1
```

```
    #My SVM:
    W = np.random.rand(Train_X_Dense.shape[1])
    W,bias = convergence_SVM(Train_X_Dense,Train_Y,Train_X_Dense.shape[1],-1,1,W,0)

    confusion_matrix, success_rate = TestTheModel(W,bias,Test_X_Dense,Test_Y)
    print("the confusion matrix is: ")
    print(confusion_matrix)
    print("the accuracy rate is: ")
    print("%.3f" %(success_rate*100) + "%")
    print()
```

13. בניית פונקציה ראשית והרצתה:

```
def main():
    """the answers to Perceptron part:
    iris.xlsx, 2, Class, 1, 0, choose of: SepalWidthCm , SepalLengthCm , PetalWidthCm , PetalLengthCm
    """

    #Perceptron Part -> run iris data:
    print("Perceptron's Results: ")
    #run for the iris data from excel:
    X,Y,num_features,features_name,Positive_Class,Negative_Class = get_data()
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, random_state=33)
    plot_X_Y(X_train,y_train)
    W = np.random.rand(num_features)
    bias = 0
    W, bias = convergence_pereptron(X_train,y_train,num_features,Negative_Class,Positive_Class,W,bias)
    confusion_matrix,success_rate = TestTheModel(W,bias,X_test,y_test)
    print("the confusion matrix is: ")
    print(confusion_matrix)
    print("the accuracy rate is: ")
    print("%.3f" %(success_rate*100) + "%")
    print()

    #SVM Part -> run make_blob data:
    print("SVM's Circle Results: ")
    #circle:
    plt.show();
    #generate random points - circle sepeartion
    X1,Y1 = make_circles(n_samples=(500,500), random_state=3, noise=0.09, factor = 0.5)
    Y1[Y1==0] = -1 #change the negative class to -1
    plt.show();
    plot_X_Y(X1,Y1)
    plt.show();
    X,Y = circle_map(X1,Y1)
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, random_state=33)
    plot_X_Y(X_train,y_train)
    W = np.zeros(2)
    bias = 0
    W, bias = convergence_SVM(X_train,y_train,2,-1,1,W,bias)
    confusion_matrix,success_rate = TestTheModel(W,bias,X_test,y_test)
    print("the confusion matrix is: ")
    print(confusion_matrix)
    print("the accuracy rate is: ")
    print("%.3f" %(success_rate*100) + "%")
    print()
    #text classification:
    TextClassification()

main()
```