תרגיל בית 2 – למידה חישובית 1
096411

<u>שמות המגישים</u>:
רוני פרידמן, ת.ז: 205517097
מור לוינבוק, ת.ז: 205451123

שאלה 1:

1) הHard SVM מספק לנו מודל, הפותר את בעיית האופטימיזציה:

$$w_0 = argmin \|w\|^2 \quad s.t \ \forall i \quad y_i\langle w, x_i\rangle \geq 1$$

והמודל שמסופק הינו:

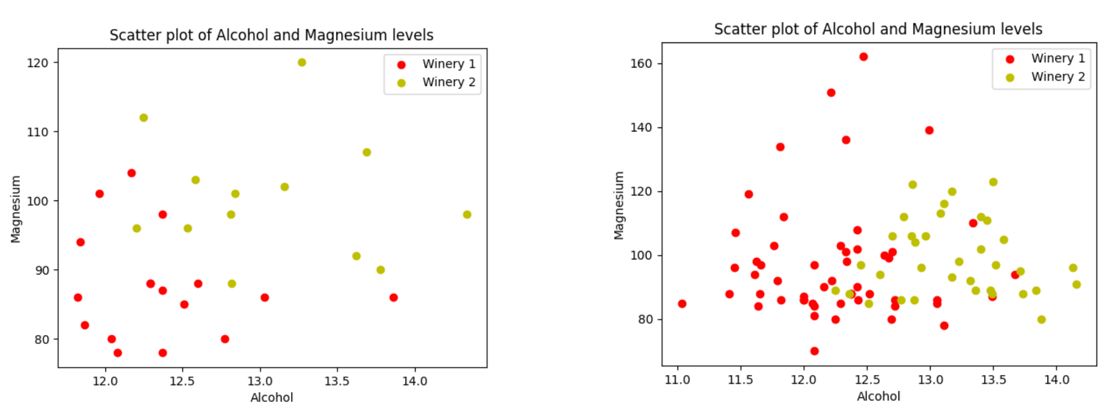
$$\widehat{W} = \frac{w_0}{\|w_0\|}$$

כלומר, המודל יספק לנו פתרון רק במידה ומדובר בדאטה פריד לינארית.
מאחר ולפי גרף הנקודות שקיבלנו מכל חלק של המדגם, ניתן ליראות ויזואלית כי הדאטה איננו פריד לינארית.
לכן, המודל לא יצליח לספק לנו פתרון.

```
def scatter_plot_df(target_list):
    fig, ax = plt.subplots()
    df1 = target_list[target_list['target'] == 1]
    df2 = target_list[target_list['target'] == 2]
    ax.set_xlabel('Alcohol')
    ax.set_ylabel("Magnesium")
    ax.scatter(df1['alcohol'], df1['magnesium'], c='r
')
    ax.scatter(df2['alcohol'], df2['magnesium'], c='y
')
    ax.legend(labels=['Winery 1', 'Winery 2'])
    plt.title("Scatter plot of Alcohol and Magnesium 
levels")
    plt.show()
```

```
# part 1:
scatter_plot_df(train_df)
scatter_plot_df(val_df)
```

נקבל את הגרפים הבאים (סט האימון מצד שמאל):



3. הוכחה:
נגדיר :

$$margin = M = \min_{i\in[m]} |\langle \widehat{w}, x_i \rangle|$$

יהי w_0 פתרון לבעיית האופטימיזציה הריבועית, המוצגת בסעיף 1.
נרצה להראות:

$$M = \frac{1}{\|w_0\|}$$

נגדיר את המרחק בין w_0 ל-support vectors כך שלכל x שהוא support vector מתקיים:

$$\langle w_0, x \rangle = 1$$

נוכל להגדיר זאת, היות וw_0 מיוצג על ידי נורמל, שנוכל להכפיל בכל קבוע שנרצה, והוא עדיין יהווה מישור מפריד, ופתרון לבעיה.
הגודל של M מוגדר בתור המרחק של x מהמישור המפריד, כלומר:

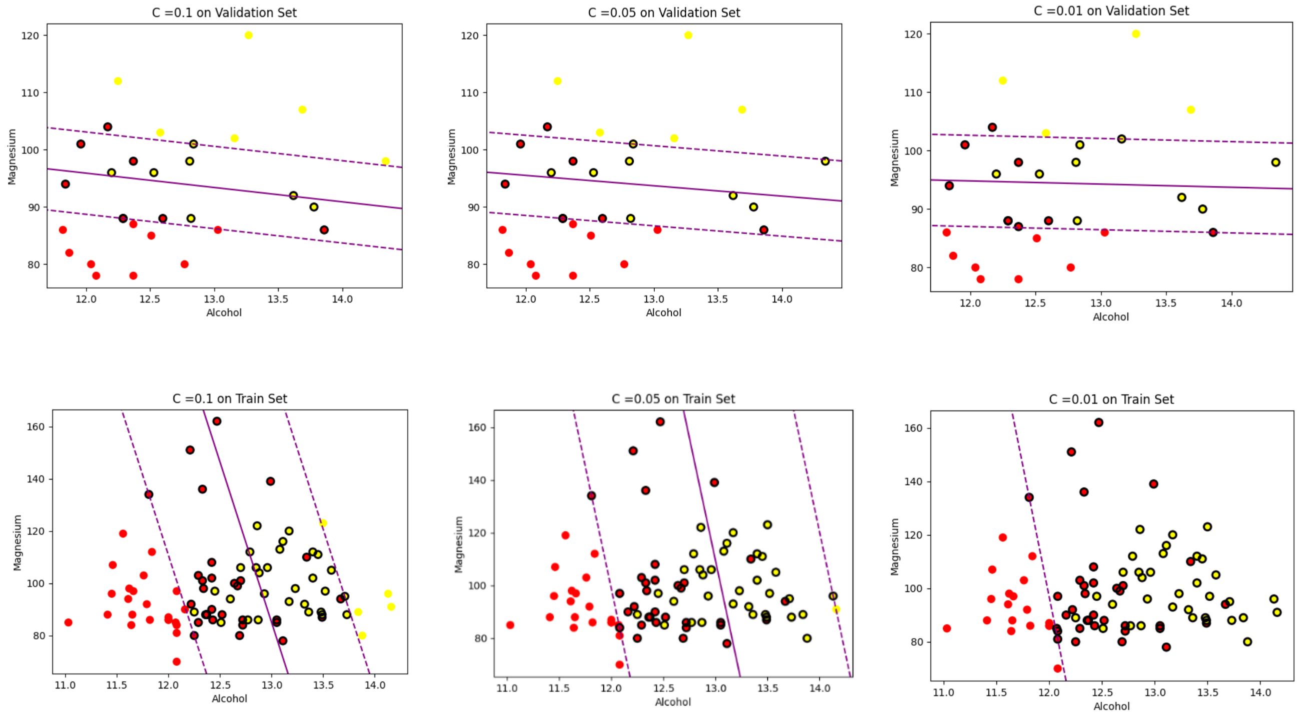
$$M = \frac{\langle w_0, x \rangle}{\|w_0\|} = \frac{1}{\|w_0\|}$$

כנדרש.

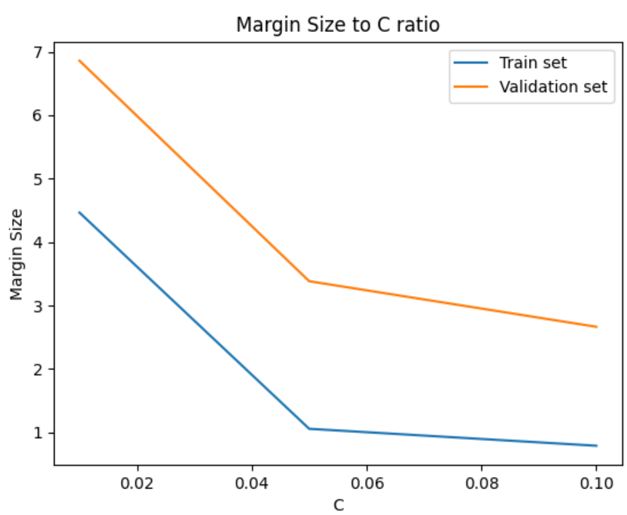
נציג את הקוד עבור כל החלקים 2-5 לא כולל חלק 3. תחילה נצרף את הקוד אשר השתמשנו בו להדפיס את הגרפים, כולל הקוד מתרגול 4, ואז את הגרפים:

```
def plot_svc_decision_function(model, ax=None,
plot_support=True):
    """Plot the decision function for a 2D SVC"""
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # create grid to evaluate model
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = model.decision_function(xy).reshape(X.shape)

    # plot decision boundary and margins
    ax.contour(X, Y, P, colors='purple', levels=[-1,
    0, 1], alpha=1,
                linestyles=['--', '-', '--'])

        # plot support vectors
        if plot_support:
            ax.scatter(model.support_vectors_[:, 0],
                       model.support_vectors_[:, 1],
                       s=50, linewidth=2, facecolors='
    none', edgecolor='black')
        ax.set_xlim(xlim)
        ax.set_ylim(ylim)
        ax.set_xlabel('Alcohol')
        ax.set_ylabel("Magnesium")
        plt.show()
```

```
    # Parts 2-5:
    train_list = []
    val_list = []
    for i in range(len(train_df['alcohol'].tolist
())):
        train_list.append(np.array((train_df['alcohol
'].tolist()[i], train_df['magnesium'].tolist()[i])))
    for i in range(len(val_df['alcohol'].tolist())):
        val_list.append(np.array((val_df['alcohol'].
tolist()[i], val_df['magnesium'].tolist()[i])))
    c = [0.01, 0.05, 0.1]
    margin_size_train = []
    margin_size_val = []
    error_rate_train = []
    error_rate_val = []
    for i in c:
        model1 = SVC(kernel='linear', C=i)
        model1.fit(train_list, train_df['target'].
tolist())
        error_rate_train.append(1-model1.score(
train_list, train_df['target'].tolist()))
        error_rate_val.append(1-model1.score(
val_list, val_df['target'].tolist()))
        margin_size_train.append(1/np.linalg.norm(
model1.coef_))
        plt.title("C =" + str(i) + " on Train Set")
        plt.scatter(train_df['alcohol'].tolist(),
train_df['magnesium'].tolist(),
                    c=train_df['target'].tolist(), s
=50, cmap='autumn')

        plot_svc_decision_function(model1)
        model2 = SVC(kernel='linear', C=i)
        model2.fit(val_list, val_df['target'].tolist
())
        margin_size_val.append(1/np.linalg.norm(
model2.coef_))
        plt.scatter(val_df['alcohol'], val_df['
magnesium'],
                    c=val_df['target'], s=50, cmap='
autumn')
        plt.title("C =" + str(i) + " on Validation
Set")
        plot_svc_decision_function(model2)
    # part 4 print
    plt.plot(c, margin_size_train, label="Train set"
)
    plt.plot(c, margin_size_val, label="Validation

set")
    plt.xlabel("C")
    plt.ylabel("Margin Size")
    plt.title("Margin Size to C ratio")
    plt.legend()
    plt.show()

    # part 5 print
    plt.plot(c, error_rate_train, label="Train set")
    plt.plot(c, error_rate_val, label="Validation
set")
    plt.xlabel("C")
    plt.ylabel("Error Rate")
    plt.title("Error Rate to C ratio")
    plt.legend()
    plt.show()
```



2. הגרפים מוצגים.
4. הגרף המתקבל:



פונקציית המטרה הינה:

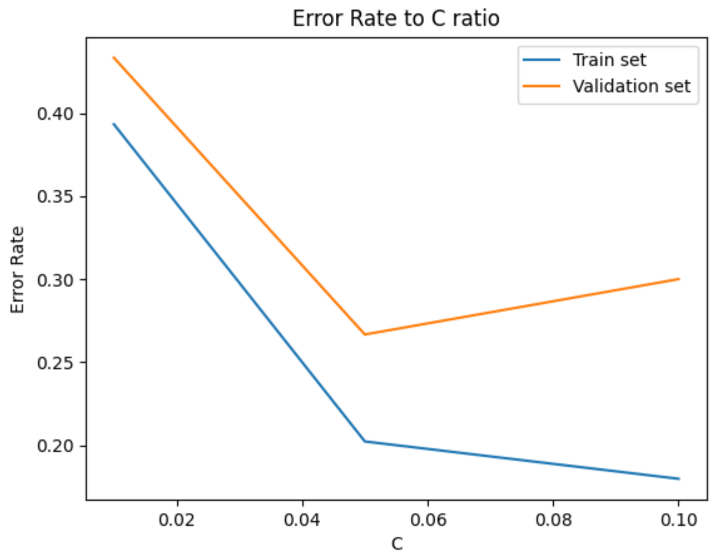
$$\arg\min_{w}\|w\|^2 + C * \sum_{i=1}^{m} \max\left(0,1 - y_i\langle w, x_i\rangle\right)$$

ניתן לראות כי כאשר C גדל, אנו נותנים פחות חשיבות לגודל המקסימלי של הmargin ובמקום זה מתמקדים בלא לבצע סיווגים שגויים, היות וכל סיווג כזה, יגרור קנס גדול יותר ככל שC גדל.

מסיבה זו, גודל הmargin יהיה קטן יותר, היות והטריידאוף בפונקציית המטרה היא

מיקסום של הmargin לעומת כמות הסיווגים השגויים – כאשר C גדול, כפי שאמרנו, נבחר להקטין את הmargin על מנת לבצע פחות סיווגים שגויים.

5. הגרף הינו:

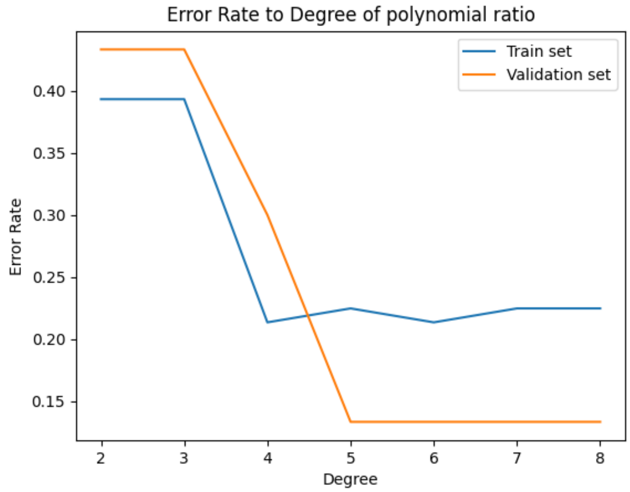


תחילה נתייחס לTraining Set – כאשר C קטן, הMargin גדול, ולכן הוא מאפשר הרבה סיווגים לא נכונים. ניתן לראות כי המידע לא מופרד בצורה מדוייקת לפי הגרף המתקבל (המישור המפריד לא ניראה בגרף, ניתן לראות כי הצד השמאלי של הMargin כולל בתוכו הרבה דוגמאות צהובות, שאמורות להיות מסווגות כאדומות).
כאשר C גדל, הMargin אומנם קטן גם, אך אנחנו מאפשרים פחות סיווגים לא נכונים, מה שבסך הכל, מוריד את הטעות של המסווג על הדוגמאות.
לפי הגרף של ה Validation Set – כאשר הגדלנו את C בפעם הראשונה, הטריידאוף השתלם, ובסך הכל המסווג סיווג טוב יותר למרות שהMargin קטן יותר.
לאחר מכן, כשהגדלנו את C שוב ל – 0.1 , הקטנו את הMargin אך מספר הדוגמאות שסווגו כלא נכונות בסט האימון, לא השתנה בהרבה. דבר זה, אומר שהטריידאוף לא השתלם, והמסווג יהיה פחות מדוייק מאשר כש $C = 0.05$

הקוד עבור סעיפים 6+7:

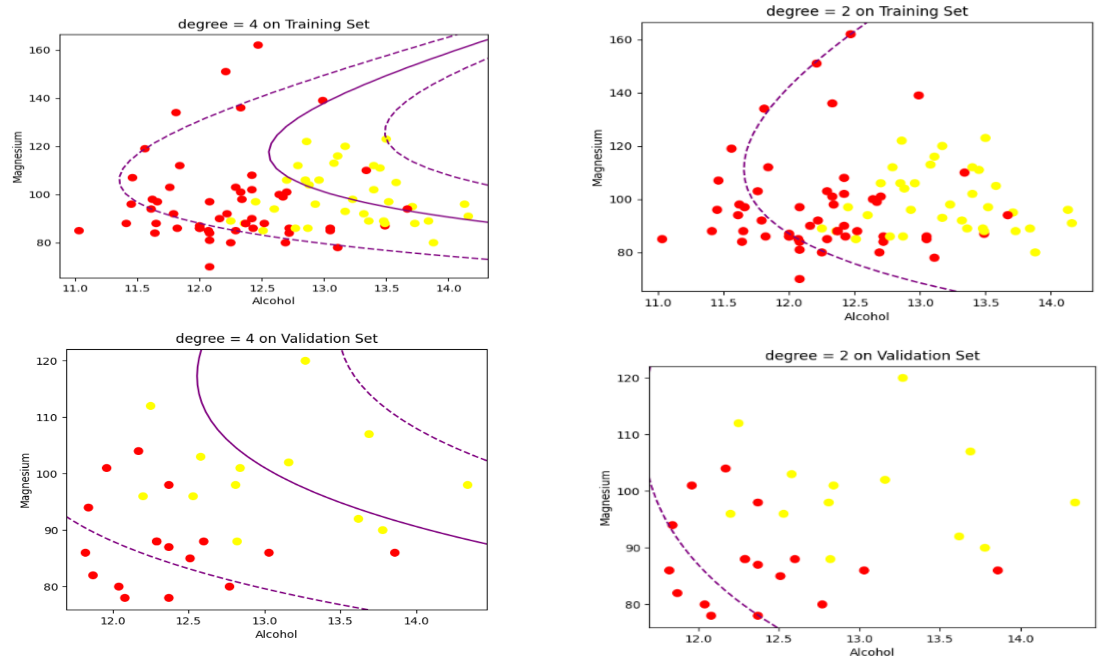
```
    # part 6 + 7
    degree = [i for i in range(2, 9)]
    error_rate_train = np.zeros([7, ])
    error_rate_val = np.zeros([7, ])
    for deg in degree:
        model = SVC(kernel='poly', C=1, degree=deg)
        model.fit(train_list, train_df['target'].
tolist())
        train_error = 1-model.score(train_list,
train_df['target'].tolist())
        val_error = 1-model.score(val_list, val_df['
target'].tolist())
        error_rate_train[deg-2] = train_error
        error_rate_val[deg-2] = val_error
    max_min_train = [np.argmax(error_rate_train)+2,
np.argmin(error_rate_train)+2]

    # printing the line graph, and the 4 degrees
graphs
    plt.plot(degree, error_rate_train, label="Train
set")
    plt.plot(degree, error_rate_val, label="
Validation set")
    plt.xlabel("Degree")
    plt.ylabel("Error Rate")
    plt.title("Error Rate to Degree of polynomial
ratio")
    plt.legend()
    plt.show()

    for i in max_min_train:
        model = SVC(kernel='poly', C=1, degree=i)
        model.fit(train_list, train_df['target'].
tolist())
        plt.scatter(train_df['alcohol'].tolist(),
train_df['magnesium'].tolist(),
                    c=train_df['target'].tolist(), s
=50, cmap='autumn')
        plt.title("degree = " + str(i) + " on
Training Set")
        plot_svc_decision_function(model,
plot_support=False)
        plt.scatter(val_df['alcohol'].tolist(),
val_df['magnesium'].tolist(),
                    c=val_df['target'].tolist(), s=
50, cmap='autumn')
        plt.title("degree = " + str(i) + " on
Validation Set")
        plot_svc_decision_function(model,
plot_support=False)
```

6.הגרף שהתקבל:



ניתן לראות שבאופן כללי, דרגה גבוהה יותר משפרת את ביצועי המסווג, עד לרמה מסויימת – את זה אנחנו למדים מהגרף של הTraining Set שקיימים בו גם עליות של הError Rate מידי פעם.
לדעתנו, הסיבה לכך היא המידע שבו אנחנו משתמשים.
המידע הנתון לנו, נתון לנו על ידי זוג פרמטרים בלבד. כאשר אנחנו משתמשים בפונקצייה פולינומית על מנת להעלות את נק' הדגימה למימד גבוה יותר, אנחנו מקבלים מידע אשר תכונתיו חוזרות על עצמו, היות והכניסות בוקטורים החדשים, הגיעו מאותם פרמטרים, אשר (אנו מאמינים) מגיעים מאותה התפלגות (כלומר הרנדומליות המגולמת בכל כניסה של הוקטורים החדשים, מגיעה מאותן זוג התפלגויות, של הכניסות המקוריות).
ולכן, אנחנו מגיעים למצב שבו השגיאה לא תרד (ואולי אף תעלה) ככל שנעלה בדרגה.

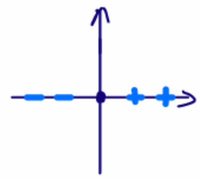
7. גילינו שעבור סט האימון (ניתן לראות בגרף בסעיף 6) הטעות הכי נמוכה מתקבל כאשר $degree = 4$ והכי גבוהה כאשר $degree = 2$.
הגרפים המתקבלים הינם:



שאלה 3:

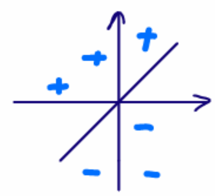
1. נקבל שגיאת סיווג 0 רק במידה ומדובר בדאטא פריד לינארית.

<u>עבור d=1:</u>



המישור המפריד הינו הנקודה 0.

<u>עבור d=2:</u>



המישור המפריד הינו ישר המעבר בראשית הצירים.

2. נתונות שתי נקודות: $x_1=\overset{(-1)}{(p,0)}$, $x_2=\overset{(1)}{(0,q)}$ עם התיוגים $y_1=-1, y_2=1$

נרצה למצוא את p, q שעבורם ישנו פתרון עבור Hard SVM:

$$\arg\min_{w} \|w\|^2$$
$$\text{s.t} \quad \forall i\in\{1,\dots,m\}: y_i\langle w,x_i\rangle \ge 1$$

נציב את הנקודות באילוצי הבעיה:

<u>$x_1=(p,0)$, $y_1=-1$:</u>

$$-1\cdot\left[(w_0,w_1)\begin{pmatrix}p\\0\end{pmatrix}\right] = -1(w_0 p+0) = -w_0 p \ge 1 \Rightarrow w_0 p \le -1 \Rightarrow \boxed{w_0 \le -\frac{1}{p}}$$

<u>$x_2=(0,q)$, $y_2=1$:</u>

$$1\cdot\left[(w_0,w_1)\begin{pmatrix}0\\q\end{pmatrix}\right] = 0+w_1 q = w_1 q \ge 1 \Rightarrow \boxed{w_1 \ge \frac{1}{q}}$$

כעת, נסתכל על פונקציית המטרה. נרצה להביא למינימום את:

$$\|w\|^2 = w_0^2+w_1^2$$

הביטוי $w_0^2+w_1^2\ge 0$ וכך גם כל אחד מהביטויים הריבועיים $w_1^2\ge 0$, $w_0^2\ge 0$,
ולכן, כדי להביא למינימום את פונק' המטרה נשאף למזער את כל אחד
מהביטויים הללו ולכן נרצה למצוא את $|w_1|, |w_0|$ הקטנים ביותר.

נבחן את כל המקרים האפשריים של p,q:

1) עבור p=0 וגם q=0: לא קיים מישור מפריד, כי קיימת רק נק' אחת.

עבור $p \neq 0 \; and \; q = 0$: קל לראות כי המשור המפריד יהיה מהצורה:

$$w_0 = \begin{cases} (r,0): 0 < r < p, & if \; p > 0 \\ (r,0): p < r < 0, & if \; p < 0 \end{cases}$$

באופן סימטרי, עבור $p = 0 \; and \; q \neq 0$:

$$w_0 = \begin{cases} (0,r): 0 < r < q, & if \; q > 0 \\ (0,r): q < r < 0, & if \; q < 0 \end{cases}$$

2) עבור שאר האפשרויות שהפרמטרים p,q לא שווים 0 נקבל כי הפתרון האופטימלי שבעצם הינו $W=(-\frac{1}{p},\frac{1}{q})$.

המשמעות היא כי זהו הפתרון שמביא למינימום את $\|w\|$! ו-$\|w\|$ לכן זה הפתרון שמביא למינימום את פונקציית המטרה: $\|w\|^2=\frac{1}{p^2}+\frac{1}{q^2}$.

3. אם היינו מאמנים שוב את המודל הרי שהיינו מקבלים מפריד זהה. הסיבה לכך: היא שהמפריד מתקבל לפי ה- Support Vectors שהן הנקודות שה- margin שלהם המינימלי למישור המפריד ומאחר והנקודות הנוספות לא משנות את ה-Support vectors, הרי שהמישור המפריד יישאר זהה.

4)ככל שλ גדל הביטוי יושפע הרבה יותר על ידי $\lambda\|w\|$ כלומר, החלק בביטוי של $\frac{1}{m}\sum_{i=1}^{m}\max\{0,1-y_i\langle x_i,w\rangle\}$ משפיע פחות על הביטוי. לכן האלגוריתם ירשה לעצמו להגדיל את כמות הסיווגים הלא נכונים, וכך יוכל להגדיל את הMargin.

כאשר λ קטן, החלק המשפיע יותר בביטויי יהיה $\frac{1}{m}\sum_{i=1}^{m}\max\{0,1-y_i\langle x_i,w\rangle\}$ ולכן הביטויי יגדל מאוד על כל סיווג לא נכון שהמסווג יבצע. האלגוריתם ינסה אם כך, לסווג כמה שפחות טעויות על נק' האימון, ובשביל כך הוא יקטין מאוד את הMargin.

כלומר – אנחנו מצפים לראות Margin גדול יותר עבור $\lambda = 200$ וMargin קטן יותר עבור $\lambda = 2$.

<u>שאלה 4:</u>

1. לפי Representers Theorm נרצה למצוא עבור בעייה מהסוג:

$$\min_w f(\langle w,\psi(x_1)\rangle, \dots \langle w,\psi(x_m)\rangle) + R(\|w\|)$$

פתרון, אשר יתקבל מאלגוריתם הPerceptron. הבעיה שאנחנו מנסים לפתור עם הPerceptron תוגדר כ:

$$\min_w \sum_{i=1}^{m} \max\{0,1-\langle w,\psi(x_i)\rangle\}$$

שזהו מקרה פרטי של HARD SVM רק ללא התחשבות בMargin. נגדיר אם כך

$$R(\|w\|) = 0$$

$$f = \frac{1}{m}\sum_{i=1}^{m} \max\{0,1-\langle w,\psi(x_i)\rangle\}$$

ניתן לראות כי f מקיימת את תנאי המשפט, היות והיא תלוייה אך ורק במכפלות הפנימיות בין המישור המפריד לנקודות $\psi(x_i)$, ו-R הינה פונקצייה שלכל w מחזירה ערך קבוע, ולכן הפונקצייה הינה מונוטונית עולה חלש.

לכן, קיים פתרון אופטימלי, אשר ניתן לביטויי כצירוף לינארי של הנקודות:

$$w = \sum_{i=1}^{m} \alpha_i \psi(x_i)$$

כמו שהראנו בהרצאה, מתקיים כי:

$$\langle w,\psi(x_i)\rangle = \sum_{j=1}^{m} \alpha_i \langle \psi(x_j),\psi(x_i)\rangle$$

נרצה שהאלגוריתם יספק לנו w שמקיים:

$$y_i\langle w,\psi(x_i)\rangle = y_i\sum_{j=1}^{m} \alpha_i \langle \psi(x_j),\psi(x_i)\rangle \geq 1 \quad \forall i \in [m]$$

אם הוא מסווג נק' כל שהיא x_i בצורה לא נכונה, זה אומר כי:

$$\max\{0,1-\langle w,\psi(x_i)\rangle\} = 1-\langle w,\psi(x_i)\rangle \to 1-\sum_{j=1}^{m}\alpha_i K(x_i,x_j) \geq 0$$

אנו יודעים שהנקודות במקומות קבועים ולכן $K(x_i,x_j)$ קבוע ואי שלילי. על מנת שנקבל כי הסיווג נכון, נרצה להגדיל את הביטויי $\sum_{j=1}^{m}\alpha_i K(x_i,x_j)$ שהוא החלק השלילי בביטויי.

זה אומר, שנצטרך להגדיל את α_i עבור סיווג שגויי. <u>נבחר להעלות אותו כל פעם ב1.</u>

כלומר, האלגוריתם יהיה:

Input: Function K, and $S^{\psi} = \{(\psi(x_i), y_i)\ \forall i \in [m]\}$

Initialize: $\alpha \in \mathbb{R}^m,\ \ \alpha = (0,0,..,0)$
for t=1,2,…

If $y_i \sum_{j=1}^{m} \alpha_i K(x_i, x_j) \le 1$

$\alpha_i = \alpha_i + 1$

Return α

על מנת לסווג נקודה חדשה x^*, נעשה:

$$L(x^*, \alpha) = sign\left(1 - y_i \sum_{j=1}^{m} \alpha_i K(x^*, x_j)\right)$$

.2

<u>כיוון =>:</u>נניח שהאלגוריתם מתכנס. ונניח בשלילה, שS^{ψ} לא פריד לינארית.
כלומר, לכל מפריד לינארי w, קיימת נקודה $(\psi(x_i), y_i)$ שמקיימת:

$$y_i \langle w, \psi(x_i) \rangle < 0$$

לפי הלולאה באלגוריתם שכתבנו בסעיף 1, אם קיימת נקודה כזו, היא גם תקיים:

$$y_i \sum_{j=1}^{m} \alpha_i K(x_i, x_j) = y_i \langle w, \psi(x_i) \rangle < 0 \le 1$$

כלומר, האלגוריתם לא יתכנס, וימשיך בלולאה בגלל נקודה זו.
אם לכל מפריד ולכל וקטור α מצב זה יקרה, האלגוריתם לעולם לא יתכנס, בסתירה.

<u>כיוון <=:</u>
נתון שS^{ψ} פריד לינארית. האלגוריתם שלנו, מחפש וקטור α כך שמתקיים:

$$y_i \sum_{j=1}^{m} \alpha_i K(x_i, x_j) = y_i \langle w, \psi(x_i) \rangle \ge 1 \quad \forall i \in [m]$$

נגדיר:

$$w^{(t)} = \sum_{i=1}^{m} \alpha_i^{(t)} \psi(x_i)$$

שאותו קיבלנו באמצעות וקטור ה$\alpha^{(t)}$ שהאלגוריתם שלנו מספק בזמן t.
כעת, $w^{(t)}$ מישור (לאו דווקא מפריד) במרחב של הנקודות $\psi(x_i)$.

נניח בשלילה כי האלגוריתם לעולם לא מתכנס. כלומר ניתן להסיק, כי לכל t נקבל כי $w^{(t)}$ אינו מישור מפריד עבור הנקודות S^{ψ}.

אך, אם נריץ את האלגוריתם *Perceptron* בינארי, הרגיל, על קבוצת הנקודות S^{ψ} נקבל ש$w(t)$ שלו, שקול ל-$w^{(t)}$ של האלגוריתם מסעיף 1.

זאת כי הוא מאותחל ל-0 בכל כניסה בהתחלה, ובכל צעד, יתקיים:

$$w(t+1) = w(t) + \psi(x_i)y_i$$

כלומר אנחנו נחסר או נוסיף את $\psi(x_i)$ המסווג לא נכון, ב1. באופן דומה,

$$w^{(t)} = \sum_{i=1}^{m} \alpha_i^{(t)} \psi(x_i)$$

כלומר צעד באלגוריתם *Perceptron* הרגיל, יהיה צעד על המישור $w^{(t)}$

הנחנו כי האלגוריתם מסעיף 1 לא מתכנס <= כלומר $w^{(t)}$ לא מישור מפריד לכל t <=

<= האלגוריתם $Perceptron$ על S^{ψ} לא מתכנס גם הוא , וזאת סתירה.