למידת מכונה הרצאה 6

ניזכר באלגוריתם adaboost:

יש לנו אוסף של חוקים באיכות נמוכה, אנחנו נחבר אותם כדי לקבל חוק באיכות גבוהה. הבעיה היא: כמו שראינו שבוע שעבר, אם מחברים הרבה חוקים, המימד-VC גדל. אז החסם הכללה מורכב משני גורמים: הטעות האמפירית על המדגם, ועוד גורם שתלוי במימד-VC. אז מתחילים עם חוקים שיש להם טעות גבוהה ומימד נמוך, ומחברים אותם ומקבלים חוק עם טעות נמוכה ומימד גבוה. נרצה למצוא איזון בין הדברים האלה.

נזכיר את מהלך האלגוריתם:

קלט: קבוצת נקודות מתוייגות. מספר איטרציות רצוי k. קבוצת חוקים חלשים.

פלט רצוי: משקל (גודל חיובי או שלילי) לכל חוק.

נאתחל משקל שווה לכל **נקודה**. $D_1(x_i) = 1/n$

1. נרוץ k פעמים:
   a. נחשב את הטעות הממושקלת של כל החוקים (לפי משקלי הנקודות) $\epsilon_t(h) = \sum_{i=1}^{n} D_t(x_i)[h(x_i) \neq y_i]$.
   b. נבחר את החוק עם הטעות המינימלית.
   c. נגדיר משקל לחוק כפונקציה של הטעות (טעות נמוכה – משקל גדול).
   d. נעדכן את משקלי הנקודות לפי המשקל הקודם והחוק הנוכחי. נחלק בגורם מנרמל.

החוק הסופי הוא צ"ל של החוקים החלשים: $F(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$.

התיוג על נקודה חדשה יהיה: $H(x) = sign\big(F(x)\big)$.

חזרה על ניתוח האלגוריתם:

המשקל שהאלגוריתם קובע לכל מסווג (החוק הוא h_t – כלומר החוק שנבחר באיטרציה ה-t כי הוא היה בעל טעות מינימלית):

$$\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t(h_t)}{\epsilon_t(h_t)}$$

וראינו שאם הטעות נמוכה, המשקל גדול. ואם הטעות גבוהה (טעות קרובה לחצי) אז המשקל קרוב לאפס.
הטעות חסומה בחצי כי אחרת פשוט ניקח את החוק הנגדי.

השלב של עדכון משקלי הנקודות וחלוקה בגורם המנרמל:

$$D_{t+1}(x_i) = \frac{1}{Z_t} D_t(x_i) e^{-\alpha_t h_t(x_i) y_i}$$

אם התיוג של הנקודה (y_i) זהה לקביעה של החוק הנוכחי ($h_t(x_i)$), אז הסימן של המכפלה שלהם הוא חיובי. ובגלל המינוס שיש בחזקה, בעצם מקטינים את משקל הנקודה. כלומר אם צדקתי לגבי נקודה המשקל שלה יורד (כי כבר יש חוק שמתאים לה אז פחות קריטי למצוא עוד חוק). ואם טעיתי לגבי הנקודה, אז המשקל עולה כי חשוב שנמצא חוק שמתאים לה.

ניתחנו את המשקל של נקודה בכל איטרציה. בכל פעם מכפילים במספר גדול מ-1 או קטן מ-1:

$$D_{t+1}(x_i) = \frac{1}{Z_t} D_t(x_i) e^{-\alpha_t h_t(x_i) y_i} =$$

והגורם $D_t(x_i)$ נקבע לפי האיטרציה הקודמת, כלומר:

$$D_t(x_i) = \frac{1}{Z_{t-1}} D_{t-1}(x_i) e^{-\alpha_{t-1} h_{t-1}(x_i) y_i}$$

נציב:

$$D_{t+1}(x_i) = \frac{1}{Z_t}\frac{1}{Z_{t-1}} D_{t-1}(x_i) e^{-\alpha_{t-1} h_{t-1}(x_i) y_i} \cdot e^{-\alpha_t h_t(x_i) y_i}$$
$$= \frac{1}{Z_t Z_{t-1}} D_{t-1}(x_i) e^{-y_i\left(\alpha_t \cdot h_t(x_i) + \alpha_{t-1} h_{t-1}(x_i)\right)} =$$

ואפשר לחזור אחורה עד האיטרציה הראשונה:

$$= \frac{1}{Z_t Z_{t-1} \cdots Z_1} D_1(x_i) e^{-y_i \sum_{j=1}^{t} \alpha_j h_j(x_i)} =$$

נגדיר $Z = Z_t Z_{t-1} \cdots Z_1$, ונשים לב שהחזקה היא בדיוק הפונקציה שהאלגוריתם מגדיר להחלטה הסופית. וניזכר ש $D_1(x_i) = \frac{1}{n}$.

$$= \frac{1}{z} \cdot \frac{1}{n} e^{-y_i F(X_i)}$$

אז אם ניקח את סכום המשקלים הסופי על כל הנקודות (אנחנו יודעים שהוא צריך להיות שווה 1):

$$1 = \sum_i D_{t+1}(x_i) = \sum_{i=1}^{n} \frac{1}{Z} \cdot \frac{1}{n} e^{-y_i F(X_i)} = \frac{1}{Zn} \sum_{i=1}^{n} e^{-y_i F(X_i)}$$

ואם נבודד את Z, נמצא: $Z_t Z_{t-1} \cdots Z_1 = Z = \frac{1}{n} \sum_{i=1}^{n} e^{-y_i F(X_i)}$.

טענה: אחרי האלגוריתם, $\bar{e}(H) \leq Z$.

$$\bar{e}(H) \stackrel{\text{א}}{=} \frac{1}{n} \sum_{i=1}^{n} [H(x_i) \neq y_i] \leq \frac{1}{n} \sum_{i=1}^{n} e^{-y_i F(x_i)} = Z$$

א – מספר הנקודות שטעינו בהן, חלקי מספר הנקודות הכולל.

אנחנו רוצים לחסום את הטעות ע"י Z, כי Z קטנה באופן מעריכי (גם את זה נוכיח).

הוכחה: מתקיים:

$$F(x_i) \stackrel{\text{א}}{=} \operatorname{sign}(F(x_i)) \cdot |F(x_i)| \stackrel{\text{ב}}{=} H(x_i) \cdot |F(x_i)|$$

א – הגיון, ב – הגדרת $H(x)$.

נחלק למקרים:

מקרה א: אם טעינו בנקודה, $H(x_i) \neq y_i$. נחשב את התרומה של הנקודה הזו לכל סכום (לפני החלוקה ב-n):

צד שמאל: $\operatorname{error}(H(x_i)) = 1$

צד ימין: $e^{-y_i \cdot F(x_i)} = e^{-y_i \cdot H(x_i) \cdot |F(x_i)|} = e^{|F(x_i)|} \geq 1$

כלומר הנקודה תרמה יותר לצד ימין.

מקרה ב: אם צדקנו בנקודה, $H(x_i) = y_i$:

צד שמאל: $\operatorname{error}(H(x_i)) = 0$

צד ימין: $e^{-y_i \cdot F(x_i)} = e^{-y_i \cdot H(x_i) \cdot |F(x_i)|} = e^{-|F(x_i)|} \geq 0$

גם פה הנקודה תרמה יותר לצד ימין.

אז אנחנו רוצים למזער את Z. נשים לב:

$$Z_t = \sum_{i=1}^{n} D_t(x_i) e^{-y_i \alpha_t h(x_i)} = \sum_{x_i \in A} D_t(x_i) e^{-\alpha_t} + \sum_{x_i \in \bar{A}} D_t(x_i) e^{\alpha_t}$$

כאשר A הן הנקודות שסווגו נכון (ולכן $y_i \cdot h(x_i) = 1$), $\bar{A}$ הן הנקודות שטעינו בהן (ולכן $y_i \cdot h(x_i) = -1$).

נגזור את Z_t ונשווה לאפס:

$$0 = Z_t' = -\sum_{x_i \in A} D_t(x_i) e^{-\alpha_t} + \sum_{x_i \in \bar{A}} D_t(x_i) e^{\alpha_t}$$

נעביר אגף:

$$\sum_{x_i \in A} D_t(x_i) e^{-\alpha_t} = \sum_{x_i \in \bar{A}} D_t(x_i) e^{\alpha_t}$$

נחלק ב $e^{-\alpha_t}$ (כי הוא לא תלוי ב-i). כלומר בעצם מכפילים ב e^{α_t}:

$$\sum_{x_i \in A} D_t(x_i) = \sum_{x_i \in \bar{A}} D_t(x_i) e^{2\alpha_t}$$

בתוך צד ימין, נשים לב:

$$\sum_{x_i \in \bar{A}} D_t(x_i) = \epsilon_t(h_t)$$

הסכום הממושקל של הנקודות שטעינו בהן, שזה בדיוק הטעות של אותה איטרציה.

והסכום בצד השני זה $1 - \epsilon_t(h_t)$ (הסכום הממושקל של הנקודות שלא טעינו בהן), כי הסכום הממושקל של כל הנקודות הוא 1.

כלומר:

$$1 - \epsilon_t(h_t) = \epsilon_t(h_t) e^{2\alpha_t}$$

זה מתקיים כאשר:

$$\frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)} = e^{2\alpha_t} \quad \Longrightarrow \quad \ln \frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)} = \ln e^{2\alpha_t} \quad \Longrightarrow \quad \ln \frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)} = 2\alpha_t \quad \Longrightarrow \quad \alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)} \right)$$

לסיכום: הערך המינימלי של Z_t מתקבל כאשר $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)} \right)$, שזה בדיוק המשקל שהאלגוריתם נותן ל α_t.

כלומר האלגוריתם ממזער את Z_t בכל שלב.

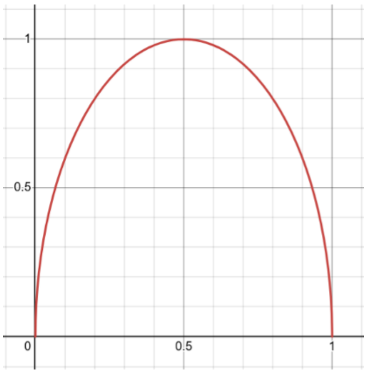
נציב את הערך של α_t ב-Z_t:

$$Z_t = \sum_{x_i \in A} D_t(x_i) e^{-\frac{1}{2} \ln \left(\frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)} \right)} + \sum_{x_i \in \bar{A}} D_t(x_i) e^{\frac{1}{2} \ln \left(\frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)} \right)} =$$

$$= e^{\frac{1}{2}\ln\left(\frac{\epsilon_t(h_t)}{1-\epsilon_t(h_t)}\right)} \sum_{x_i \in A} D_t(x_i) + e^{\frac{1}{2}\ln\left(\frac{1-\epsilon_t(h_t)}{\epsilon_t(h_t)}\right)} \sum_{x_i \in \bar{A}} D_t(x_i)$$

$$= \left(e^{\ln\left(\frac{\epsilon_t(h_t)}{1-\epsilon_t(h_t)}\right)}\right)^{\frac{1}{2}} \sum_{x_i \in A} D_t(x_i) + \left(e^{\frac{1}{2}\ln\left(\frac{1-\epsilon_t(h_t)}{\epsilon_t(h_t)}\right)}\right)^{\frac{1}{2}} \sum_{x_i \in \bar{A}} D_t(x_i) =$$

$$= (1-\epsilon_t(h_t))\sqrt{\frac{\epsilon_t(h_t)}{1-\epsilon_t(h_t)}} + \epsilon_t(h_t)\sqrt{\frac{1-\epsilon_t(h_t)}{\epsilon_t(h_t)}} = 2\sqrt{\epsilon_t(h_t)(1-\epsilon_t(h_t))}$$



הגרף של $2\sqrt{x(1-x)}$. ככל שהטעות קטנה (או גדולה מחצי ואז ניקח את הנגדי), הערך של Z_t יורד. עבור טעות קרובה לחצי, הערך מתקרב ל-1.

אז Z_t קטן באופן אקספוננציאלי, אם הטעות של החוקים קטנה (כי כל פעם מכפילים במספר קרוב לאפס).

אמרנו שאנחנו מעדיפים לא לחבר יותר מדי חוקים ביחד. אם הקטנו את הטעות האמפירית אבל המימד-VC גדל, זה מצב של *overfitting*.

ניתוח יותר פורמלי:

נציב: $\beta_t = \frac{1}{2} - \epsilon_t(h_t), \beta_t \in \left[0, \frac{1}{2}\right]$:

$$Z_t = \sqrt{1-4\beta_t^2} \le e^{-2\beta_t^2}$$

כלומר:

$$\text{error}(H) \le Z \le e^{-2\sum_{t=1}^{T}\beta_t^2}$$

קטן מהר יותר עם טעות קטנה יותר.

הנקודה העיקרית של *adaboost* היא שאפשר לחבר חוקים חלשים (ולא צריך הרבה מהם) ולקבל חוק איכותי (עם טעות אמפירית נמוכה).

SVM – Support Vector Machines

ראינו כמה אלגוריתמים למציאת מישור מפריד:

- *Brute force* – לפי הווקטורים שמגדירים את המישור, לנסות כל אפשרות. זמן ריצה אקספוננציאלי במימד.
- *Winnow*
- *Perceptron*

המימד-VC של מישור מפריד: עבור דאטא ב-d מימדים, $d+1$. אם יש מרווח γ, $\frac{O(1)}{\gamma^2}$. מימד VC גבוה מוביל ל *overfitting*.

נניח שאפשר להפריד את הדאטא. רוצים למצוא מישור מפריד עם מרווח גדול.

לא תמיד אפשר למצוא מרווח. נתחיל עם מקרה שאפשר להפריד – נקרא *hard margin*. מרווח קשיח שאחיד על S.

מה זה *support vector*? אמרנו שבהינתן נקודות, המישור עם המרווח הכי גדול נקבע ע"י מספר קבוע של נקודות. עבור נקודות בשני מימדים, הקו המפריד נקבע ע"י 3 נקודות. כל מימד שעולים מוסיף נקודה.

Hard-margin SVM:

בהינתן קבוצה S של נקודות $x_i \in S$, ותיוג $y_i \in \{-1,1\}$. אנחנו רוצים למצוא γ מקסימלי ו-w כך ש:

$$y_i[w \cdot x_i + b] \geq \gamma, \qquad \forall i, \qquad \|w\| = 1$$

w מייצג את המישור. כלומר, לכל נקודה, המכפלה הפנימית שלה עם הקו – שמייצג את המרווח שלה מהקו – גדול מ-γ. עבור נקודה עם תיוג חיובי y_i – חיובי – הנקודה בצד אחד של הקו. ועבור נקודות עם תיוג שלילי, הפוך. נשים לב:

אם $y_i = 1$, $w \cdot x_i \geq \gamma$.

אם $y_i = -1$, $w \cdot x_i \leq -\gamma$.

אפשר להתעלם מ-b כי זה רק הזזה של הקו יחסית לצירים. נגיד שיש לנו חוק שהוא קו שעובר לא דרך ראשית הצירים. אז אפשר "להזיז" את הנקודות כגוש אחד ביחס לראשית הצירים ואת הקו ביחד איתם, עד שהוא עובר דרך ראשית הצירים.

אפשר לייצג את הבעיה אחרת:

אם w מנורמל, אז לאי השוויון $y_i[w \cdot x_i + b] \geq \gamma$ יש משמעות – המרווח γ הוא ביחס לרדיוס 1. ואנחנו צריכים למצוא w שמקיים את אי השוויון עם γ מקסימלי.

ננסה שיטה אחרת: נקבע את $\gamma = 1$, חוסם מלמטה את $y_i[w \cdot x_i + b]$. ואנחנו רוצים למזער את הנורמה של w.

וזה פתרון שקול עד כדי הכפלה בקבוע. כי ככל שהמכפלה הפנימית גדולה, הנקודות רחוקות (בכיוון הנכון) מהמישור המפריד. אם הנורמה של w קטנה, זה אומר שמצאנו ישר שהנקודות רחוקות ממנו (כי המרחק גדול גם בלי שהנורמה של w גדולה). קל להגדיל את המרחק ע"י זה שנגדיל את הנורמה של w, תמיד נוכל למצוא w שמקיים את אי השוויון לכל x. החלק הקשה הוא למזער את w.

אנחנו רוצים למקסם את $\min_{x_i} \frac{|w \cdot x_i + b|}{\|w\|}$. כאשר: $|w \cdot x_i + b| \geq 1$. אז נמקסם את $\frac{1}{\|w\|}$, או נמזער את $\|w\|$.

אפשר לפתור את זה ע"י *convex optimization*, יקר מבחינה חישובית $O(n^3)$.

מישור מפריד עם טעות

כבר ראינו אלגוריתמים שעובדים בהנחה שאפשר להפריד בין הנקודות, וצריך רק למצוא את המישור המפריד. נתייחס עכשיו לבעיה מסובכת יותר – שאי אפשר להפריד לגמרי בין הנקודות. לכל מישור תהיה טעות כלשהי.

באופן כללי אנחנו יודעים שבהסתברות לפחות $1-\delta$,

$$e(h) \leq \bar{e}(h) + \sqrt{\frac{8d \ln \frac{2en}{d} + 8 \ln \frac{4}{\delta}}{n}}$$

אנחנו מכירים שבאופן כללי יש *Tradeoff*:

אם המרווח גדול, המימד $d = O(1)/\gamma^2$ קטן.

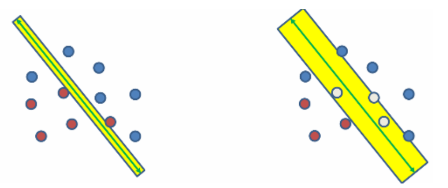
אם המרווח קטן, הטעות האמפירית קטנה.

לדוגמה ראינו ש*adaboost* יכול להוריד את הטעות האמפירית, אבל זה מגדיל את המימד-VC.

ננסה לקחת מרווח שכן יכול לטעות על כמה מהנקודות, וזה יאפשר לנו להגדיל את המרווח ולהקטין את המימד-VC. נרצה למצוא את המרווח שמקטין כמה שיותר את החסם – שתלוי גם בטעות וגם במימד-VC.

עבור אותו מישור, נשקול מרווחים שונים. למרווח הגדול יותר יש טעות אמפירית גדולה יותר (כל הנקודות שבתוך המרווח). אבל מימד VC נמוך יותר.

החסם של הטעות יכול בעצם להגיד לנו איזה מרווח עדיף.



מזעור טעויות:

תהי S קבוצה של n נקודות ב-d מימדים. רוצים למצוא מישור ב-$d-1$ מימדים עם כמה שפחות טעויות (אפילו בלי מרווח).

טענה: זה *NP-hard*.

הוכחה: רדוקציה מבעיית *Independent Set*.

נתון גרף $G=(V,E)$. רוצים למצוא תת קבוצה מקסימלית $V'\subseteq V$ של צמתים שאין ביניהם קשת. זו בעיה *NP-hard*, ואפילו קשה לקירוב.

רדוקציה: בהינתן בעיית *independent set* (להלן A), ניצור בעיית מישור מפריד שקולה (להלן B), כך שאם נפתור את בעיית המישור המפריד נפתור גם את בעיית ה *independent set*.

נשים נקודה אדומה עם משקל n על ראשית הצירים. עבור כל צומת בA, נגדיר לכל אחד מהם וקטור בסיס בB. הווקטורים האלה כחולים. אם יש צלע בין שני קודקודים בA, נשים ביניהם נקודה אדומה עם משקל n בB.

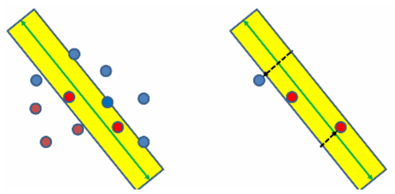
כלומר, כל מישור מפריד שנמצא ישים את כל האדומים בצד הנכון (כי לפספס אחד מהם זה כמו לפספס את כל הכחולים). וישאף לשים כמה שיותר כחולים בצד הנכון.

כל הנקודות הכחולות שבצד הנכון, אין ביניהן נקודה אדומה, כלומר אין בין הקודקודים האלה צלע בA, אז זו קבוצה בת"ל. המספר המקסימלי של נקודות כחולות בצד הנכון, זה הקבוצה בת"ל המקסימלית.

כלומר הדברים הבאים שקולים:

מישור מפריד שטועה על $k<n$ נקודות $\quad\Leftrightarrow\quad$ קבוצה בת"ל של $n-k$ קודקודים

חזרה ל*SVM*:



ופניק (*Vapnik*) וקורטז (*Cortes*) שקלו *SVM* כאשר קבוצת הנקודות לא ניתנת להפרדה.

הרעיון – *soft margin*. נאפשר טעויות. לכל נקודה x_i בצד הלא נכון, נשלם קנס ζ_i, לפי המרחק. רוצים למזער את סכום הקנסות.

בתמונה: נשלם קנס גם על הנקודה האדומה השמאלית, למרות שהיא בצד הנכון. והקנס על הנקודה הכחולה יהיה לפי המרחק מהמרווח, לא המרחק מהקו.

אנחנו רוצים:

$$\min\left(\frac{1}{n}\sum_i \zeta_i+\lambda\|w\|^2\right),\qquad s.t:\quad \forall i,\qquad y_i(w\cdot x_i)\geq 1-\zeta_i,\qquad \zeta_i\geq 0$$

נשים לב שזה דומה לתנאי שכבר ראינו: $y_i(w\cdot x_i)\geq 1$. ההבדל הוא שעכשיו יש תנאי קל יותר, כי צריך רק לעבור את $1-\zeta_i$. המרחק שנשאר לנו כדי לעבור את 1 זה הקנס שנשלם על הנקודה. הסכום $\frac{1}{n}\sum_i\zeta_i$ נקרא **קנס הציר** (***hinge loss***).

מה זה λ? אנחנו מנסים להקטין שני דברים במקביל. הפרמטר λ בעצם יקבע את ה *tradeoff* ביניהם. איך נבחר אותו?

נשתמש בטכניקת *cross validation*. זה טכניקה כללית לבחירת פרמטרים לאלגוריתם:

מחלקים את הדאטא לשתי קבוצות: **אימון** (*training*) **ווידוא** (*validation*). נריץ את האלגוריתם על האימון עם ערך λ מסויים, ונבדוק אותו עם הווידוא. נבחר את הלמדא הכי טוב.

מה הטעות האמיתית של *SVM*?

טענה: מתקיים:

$$imperical\ error=\bar{e}(h)\leq\frac{1}{n}\sum_i\zeta_i=hinge\ loss$$

כי מתקיים:

$$[h(x_i)\neq y_i] \le^{\text{א}} \gamma[h(x_i)\neq y_i] \le^{\text{ב}} \zeta_i$$

כאשר $[h(x_i)\neq y_i]$ הוא האינדיקטור $\{0,1\}$ לכך שטעינו בנקודה.

א – כי $\gamma=1$ כמו שקבענו. ב – כי אם טעינו על נקודה, המרחק שלה מהמרווה הוא לפחות γ. אז גם הקנס.

אפשר להשתמש בטכניקות שלנו כדי להוכיח חסם הכללה:

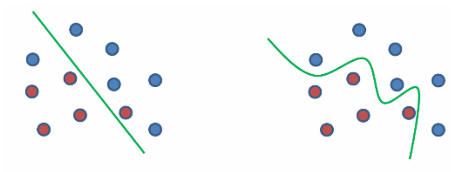
יהי w הווקטור שקיבלנו מSVM. אזי בהסתברות לפחות $1-\delta$,

$$e(h)\le \underbrace{\frac{1}{n}\sum_i \zeta_i}_{\bar{e}(h)\le} + \underbrace{\frac{4\|w\|}{\sqrt{n}}}_{\propto VC-dim} + \underbrace{(1+2\|w\|)\sqrt{\frac{2\ln\left(\frac{4\|w\|}{\delta}\right)}{n}}}_{\propto\|w\|}$$

נשים לב שהמבנה דומה לחסמי ההכללה שראינו. הטעות האמיתית, ועוד משהו שתלוי במימד-VC.

אז לסיכום: היתרון של SVM הוא שהוא מוצא מישור מפריד גם אם אין מישור מפריד בלי טעויות.

SVM **לא ליניארי –** ***nonlinear SVM***



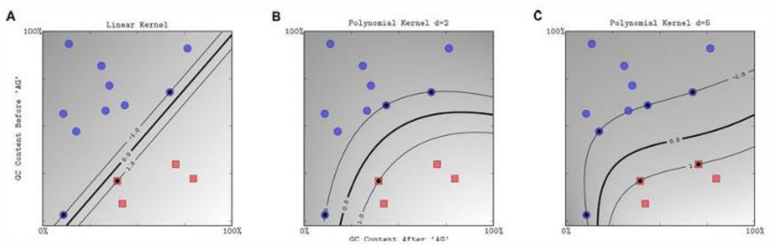
אפשר לכתוב את ה $soft\ margin\ SVM$ כך:

$$\text{Min}\left(\sum_{i=1}^n c_i-\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n y_ic_i(x_i\cdot x_j)y_jc_j\right),$$

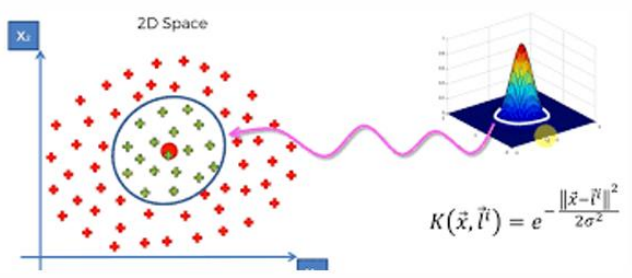
$s.t:\ \sum_{j=1}^n y_ic_i=0,\ 0\le c_i\le\frac{1}{2n\lambda}$. המכפלה הפנימית $(x_i\cdot x_j)$ נקראת $kernel$. לא נוכיח את זה (:

בגדול, במקום להתבסס על הנורמה של w (המכפלה הפנימית של w עם עצמו), נתבסס על המכפלה הפנימית של כל זוג ווקטורים. זה עובד גם עם פונקציות אחרות יותר מתקדמות שמבוססות על מכפלה פנימית, שאולי יכולות להפריד יותר נקודות:

Polynomial kernel – קרנל פולינומיאלי.
.$(x_i\cdot x_j+1)^d$



Radial Basis Function (RBF) – $e^{\frac{\|x_i-x_j\|}{-2\sigma^2}}$. ממפה את הנקודות למרחב גבוה יותר כדי שנוכל להפריד ע"י מישור. אז הפיתרון הלינארי במימד הגבוה, מקביל לפתרון לא ליניארי במימד המקורי.



השלמה משיעור 1 – הוכחת אי"ש צ'רנוף.