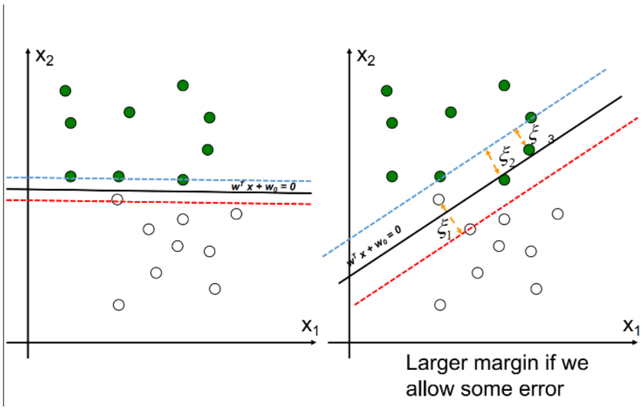
# למידה חישובית – שיעור 9

בשיעור הקודם הצגנו את אלגוריתם ה-SVM וציינו שהוא מבוסס 3 רעיונות:

1. kernel trick. מיפוי ה-data למימד גבוה, כדי למצוא מפריד לינארי בצורה קלה יותר.
2. Max margin. מציאת השוליים הרחבים ביותר שנוכל עבור בעיות מופרדות לינארית.
3. soft margin and regularization. הרחבה של ההגדרה ב-(2) עבור בעיות שאינן מופרדות לינארית, כך שיתמודד עם misclassification.

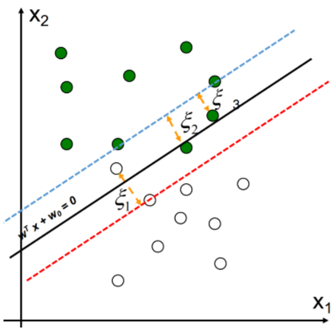
הרחבנו על ה-2 הראשונים, וכעת נרחיב על הרעיון השלישי.




בתמונה השמאלית המפריד מושלם אך הmargin קטנים, ובתמונה הימנית המפריד אינו מושלם, יש בו טעויות אך הmargin בו רחבים.

## Slack or Hinge variable

<u>נציג כעת משתנים חדשים:</u>




$\xi$ – מייצג לי את הslack של כל נקודה. עונה על השאלה כמה אני מוכנה לטעות בנקודה d. הטעות היא המרחק של הנקודה מהשוליים (הצד הנכון של הכביש בו הוא היה צריך להיות) – איור מימין מדגים מהו ה$\xi$. כלומר מותר לי להחליט באיזה נקודות אני טועה.

לכל נקודה נבחר $\xi$ משלה, ה- $\xi$ הוא אי שלילי וברוב הפעמים הוא יהיה 0 (הערך יהיה 0 עבור נקודות שלא הייתה טעות בסיווג שלהם).

נרצה להביא למינימום את הערך $||w||$ תוך כדי שאנו מתחשבים ב-3 תנאים:

1. $\forall \boldsymbol{d}, \boldsymbol{t_d}\left(\boldsymbol{w} \cdot \boldsymbol{x}^{(d)} + \boldsymbol{w_0}\right) \geq \mathbf{1} - \boldsymbol{\xi_d}$
ביצענו שינוי לאילוץ שהצגנו בשיעור שעבר. בנוסחה הקודמת רצינו שהביטוי בצד הימני של אי השוויון יהיה גדול שווה ל-1, כדי שהנקודה שתתקבל תסווג בצד הנכון של ה-margin. כעת אנו מאפשרים slack לכל אחת מהנקודות, כך שהצד הימני של אי השוויון יהיה גדול מ: $1 - \xi_d$ כלומר מאפשרים טעות כלשהי לכל נקודה.

2. $\boldsymbol{\xi_d} \geq \mathbf{0}$

3. $\sum \boldsymbol{\xi_d} \leq \boldsymbol{Const}$
הסכום של כל ה-$\xi$ים חסום על ידי קבוע, אנו רוצים להגביל את סך כל השגיאות קטן מקבוע כלשהו.
ה-Const הוא היפר- פרמטר של האלגוריתם. אנחנו צריכים להחליט עליו.

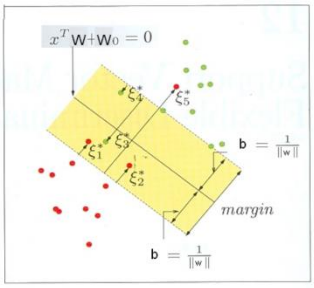
נרצה להביא למינימום את הביטוי:

$$Minimize \frac{1}{2}||w||^2 + C\sum_{d\in D} \xi\_d$$

כאשר:

- C – מחיר הטעויות. נקבע אותו לפי כמה אנו רוצים לשלם על טעויות. ככל שC קטן יותר אנו מאפשרים לאלגוריתם לבצע הרבה טעויות. כאשר C שואף לאינסוף אנו מתקרבים לSVM המקורי.
- $\sum_{d\in D} \xi_d$ – סכום הטעויות. מה שאני משלמת על כל הטעויות שלי.

הmargin מוגדר להיות $\frac{1}{||w||}$ ולכן אם נרצה margin גדול יותר נרצה שהנורמה של w תהיה קטנה יותר.




**SVM – סיכום**

קלט: data המכיל instanceים וlabelים כאשר instanceים מיוצגים באמצעות וקטורים. על סמך הקלט אני צריכה לדעת כיצד לסווג SVM.

אני צריכה להחליט על:

1. מה מחיר הטעות
2. kernel. נחליט על kernel על ידי ניסוי וטעיה. נתחיל בלי kernel, נעלה ל2 וכו'. בחיים האמיתיים אנו לא באמת נחליט על kernel אלא נשתמש באלגוריתמים שיחליטו עבורנו.

הפלט:

1. תת קבוצה של הsupport vector.
2. קבוצת המשקולות לsupport vector.

נרחיב כעת על kernelים:

## מטריצת גרם – Gram Matrix

תהי פונקציה סימטרית : $K: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$.

עבור כל קבוצה S סופית של נקודות $x_1, x_2, \dots, x_k$ ב-$\mathbb{R}^n$ מטריצת הגרם G עבור S ו-K מוגדרת:

$$G(i,j) = K\left(x_i, x_j\right), \quad 1 \le i,j \le k$$

הערה: המטריצה סימטרית ולכן מטריצת הגרם שלה תהיה סימטרית גם כן. מימדי המטריצה: $k \times k$.

## משפט מרסר – Mercer's Theorem

פונקציה סימטרית K היא kernel עבור $\varphi$ אם ורק אם לכל S סופי, מטריצת הגרם של K ו-S היא positive definite (חיובית בהחלט).

נאמר שמטריצה **חיובית בהחלט** אם המטריצה $A$ היא סימטרית, ולכל וקטור $x$ מתקיים $(x, Ax) > 0$, כלומר המכפלה הפנימית של $x$ ו-$Ax$. כלומר אם אני מכפילה את המטריצה בכל וקטור מאותו מימד, אני תמיד אהיה ממופה לאותו צד של המישור.

דוגמה:

נגדיר:

$$v = (a\ b)\ ,\ \forall v \neq 0$$

$$A = \begin{pmatrix} (x,x) & (x,y) \\ (x,y) & (y,y) \end{pmatrix}$$

$$\underbrace{(a\ b)}_{v} \cdot \underbrace{\begin{pmatrix} (x,x) & (x,y) \\ (x,y) & (y,y) \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} a \\ b \end{pmatrix}}_{v} = \underbrace{\left(a(x,x) + b(x,y), a(x,y) + b(y,y)\right)}_{vA} \cdot \underbrace{\begin{pmatrix} a \\ b \end{pmatrix}}_{v}$$

$$a^2(x,x) + ab(x,y) + ab(x,y) + b^2(y,y)$$
$$= a^2(x,x) + 2ab(x,y) + b^2(y,y)$$
$$= (ax + by,\ \ ax + by) > 0$$

כי מכפלה פנימית של וקטור שונה מ-0, גדולה ממש מ-0.

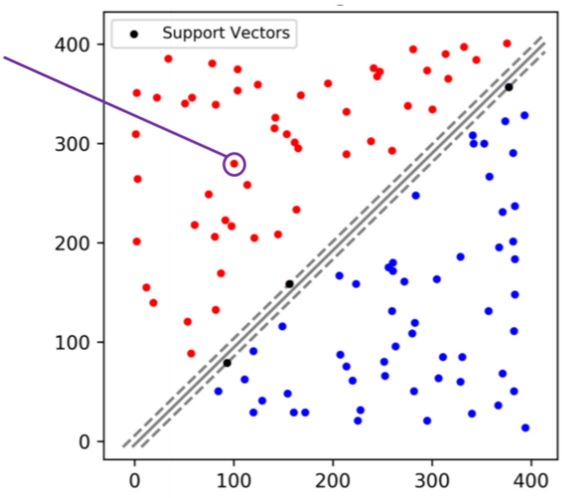
הוכחנו עבור המקרה הכללי, לכן קיבלנו שמטריצה A היא **חיובית בהחלט**.

השימוש היותר נפוץ של משפט מרסר, הינו להראות שמטריצה לא חיובית בהחלט. מספיקה דוגמה נגדית אחת בלבד כדי להראות זאת.

ניזכר בשיטת LOOCV שהצגנו בתרגול 6 שמגדירה לנו דרך להשתמש ב-dataset אחד ללמידה ולבדיקה.
השיטה:

1. נסיר מה-training data נקודת instance אחת בלבד, $x_d$.
2. נריץ את האלגוריתם הלומד על ה-training data החדש שלי
3. הרץ את המסווג שהתקבל על ה-instance הבודד שהוצאנו. נסמן את השגיאה עם $e_i$
4. נסכום את כל הטעויות ונחלק במספר הפעמים שהרצנו את האלגוריתם- ונקבל את ממוצע הטעויות.

נביט כעת על ה-dataset הבא:




אם נבחר בנקודה $x_d$ שאינה support vector היא לא תשפיע.

הנקודה הזו בהכרח לא תהיה misclassified באלגוריתם הSVM. כל הנקודות שאינן support vectors יסווגו נכון על ידי הSVM בעת שימוש בLOOCV.

לכן כאשר נשתמש בLOOCV נשאף תמיד לבחור בsupport vectors בתור ה-instance שאנו מוציאים.
קיבלנו שאם אנו עושים הערכה של הטעות שלנו באמצעות LOOCV עבור אלגוריתם SVM, ההערכה של הטעות שתתקבל היא:

$$error \leq \frac{|Support\ Vectors|}{|D|} = \frac{\text{מספר } Support\ Vectors}{\text{גודל ה}data}$$

אנחנו יכולים להסיק מכך שאם SVM החזיר לי מעט support vectors, אז ככל הנראה ה-data שלי מופרד.

*ניזכר שהגדרנו שה-support vectors הן כל הנקודות שמקיימות $\alpha_d \neq 0$, כלומר כל הנקודות שלא סווגו טוב.

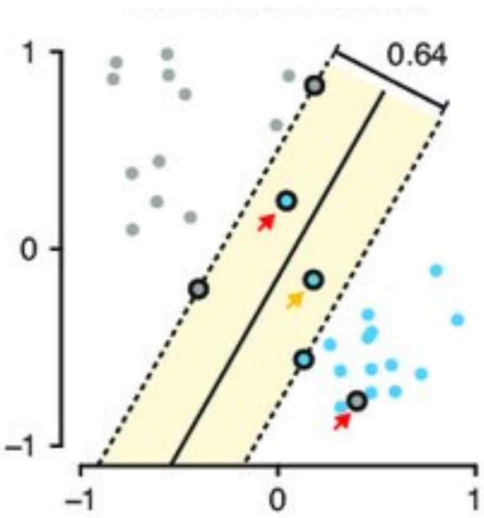
נוכל להכליל את הטעות שקיבלנו ולטעון:

$$E[error_D(h)] \leq \frac{E[\#support\ vectors\ (D)]}{|D|}$$

כלומר שהתוחלת של הטעות קטנה או שווה לתוחלת של מספר הsupport vectors ב-data שלי לחלק לגודל ה-data.

אנחנו יכולים לחשוב על ה-support vectors כנקודות ב-training data שיכולות לשנות את הצורה של המסווג שלנו, אם נסיר אותן.

הטענה נכונה עבור SVM עם שוליים קשיחים ושוליים "רכים" כאחד.

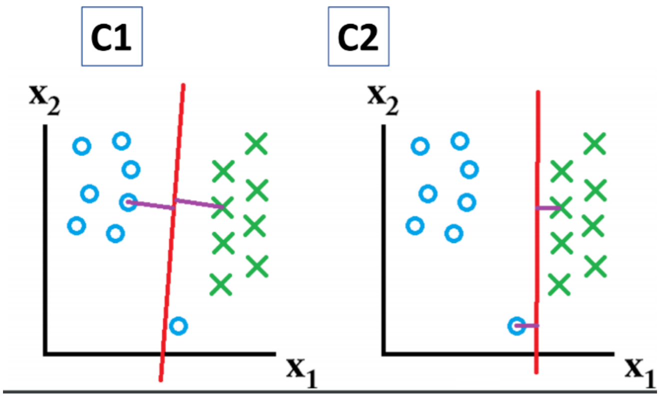



נגדיר את ה-support vectors להיות כל הנקודות שנמצאות על ה-margin או כאלה שעברו את ה-margin (בצד הלא נכון).

לדוגמה, הנקודה האפורה שנמצאת מימין למסווג, היא support vector. שילמנו עליה הרבה כי היא רחוקה מאוד מהמקום בו היא אמורה להיות.

ההשפעה של C:

בדוגמה שמולנו אנו רואים 2 אפשרויות לבחור את $w$, כאשר בשמאליץ אנו מאפשרים טעות אחת על נקודה שסווגה לא נכון ובימנית אנו לא מאפשרים טעות בכלל.




נבחר את הגישה הימנית, במידה ו-C שלי מאוד גבוה. במקרה זה כל טעות עולה לי המון ואני לא מעוניינת לאפשר טעויות כמעט בכלל (כמה שניתן).

נבחר בגישה השמאלית במידה ו-C שלי יחסית נמוך, ואז אני אוכל להביא את w לערך מינימלי.

נזכור שאני מגדירה את C לפני ריצת האלגוריתם, בהתאם לפלט האלגוריתם שאני מעוניינת לקבל.

## Performance Evaluation of Classifier

נתמקד ב-classifier עם 2 אפשרויות: positive, negative.

ניזכר ב-Confusion Matrix:

| True class ↓ | Predicted class | |
|---|---|---|
| | positive | negative |
| positive (#P) | #TP | #P - #TP |
| negative (#N) | #FP | #N - #FP |

קיבלנו 2 נוסחאות חדשות:

$$True\ Positive\ Rate = \boldsymbol{TPR} = \frac{\#TP}{\#P}$$

הסיכוי שננבא True בהינתן שה-instance הוא True.

$$False\ Positive\ Rate = \boldsymbol{FPR} = \frac{\#FP}{\#N}$$

הסיכוי שננבא False בהינתן שה-instance הוא True.

$$Accuracy = \frac{Correctly\ Classified}{All\ Instances} = \frac{TP + TN}{TP + FP + TN + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

אנחנו רוצים להיות בכמה שיותר TPR ובכמה שפחות FPR, ולכן נגדיר את הפונקציה הבאה שתחשב את הביצועים של האלגוריתם

$$\pi = \alpha \cdot TPR - FPR$$

כך ש - $\alpha \geq 0$ והיא בעצם פרמטר לנוסחה.

## ROC Curve

הצגה גרפית לביצועים של מסווג בינארי. למסווג בינארי יש שני פרמטרים שמעניינים אותנו: TPR ו-FPR. קורדינטת x היא ה- FPR, וקורדינטת y היא ה - TPR. המסווג הטוב ביותר יהיה המסווג הקרוב ביותר לפינה השמאלית עליונה של הגרף, כי במיקום זה יש הרבה TPR ואין בכלל FPR.

<u>איך בונים את ה - ROC Curve?</u>

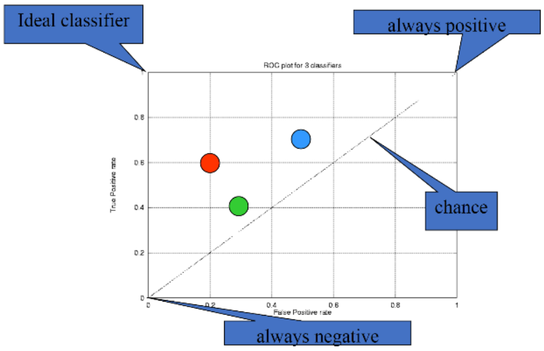
מנסים classifiers שונים (ע"י הזזה של הגבול) ומכניסים אותם לעקומה. כל classifier הוא נקודה על הגרף. בסוף נעביר קו בין הנקודות השונות.

<u>מה המשמעות של $\alpha$?</u>

נרצה לתת משקל כלשהו לסוגי הטעות. $\alpha$ מגדיר מה יותר חשוב: TPR או FPR.

$$\pi = \alpha \cdot TPR - FPR$$

<u>דוגמה:</u>




| | Predicted | |
|---|---|---|
| True | pos | neg |
| pos | 40 | 60 |
| neg | 30 | 70 |

| | Predicted | |
|---|---|---|
| True | pos | neg |
| pos | 70 | 30 |
| neg | 50 | 50 |

| | Predicted | |
|---|---|---|
| True | pos | neg |
| pos | 60 | 40 |
| neg | 20 | 80 |

Classifier 1
TPR = 0.4
FPR = 0.3

Classifier 2
TPR = 0.7
FPR = 0.5

Classifier 3
TPR = 0.6
FPR = 0.2

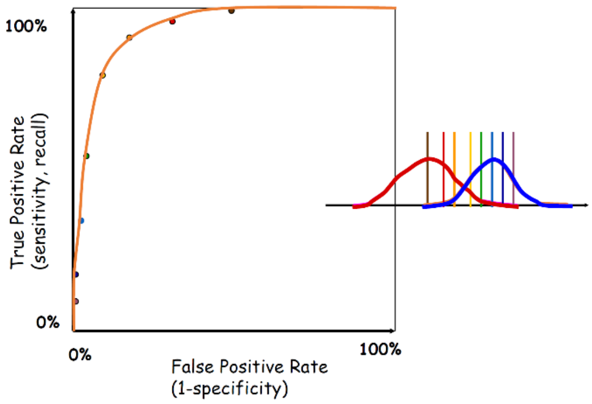
<u>איך נדע איזה classifier הוא הטוב ביותר?</u>

נעביר קו בשיפוע של $\alpha$. אם $\alpha = 1$, השיפור יהיה $45°$, נתחיל לעלות למעלה לכיוון הפינה השמאלית העליונה. הנקודה האחרונה שניתקל בה היא גם הנקודה של ה-classifier הטוב ביותר.

נשים לב שהדבר תלוי כמובן בקביעת $\alpha$ ובהחלטה שלנו מי חשוב יותר: TPR או FPR. ההחלטה הזו תשנה את השיפוע של הקו שנעביר, ולכן תשנה גם את הנקודה האחרונה בה ניתקל.

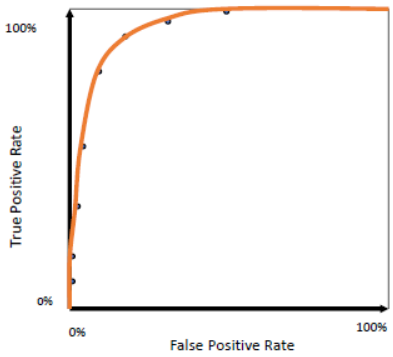
נביט בדוגמה הבאה:

בדוגמה זו יש 8 classifiers שונים, כך שלכל אחד מהם FPR ו-TPR שונים. הצבנו כל classifier כזה על הגרף (לפי ערכי הFPR ו-TPR שלהם) ומתחנו קו בין כל הנקודות שהתקבלו.

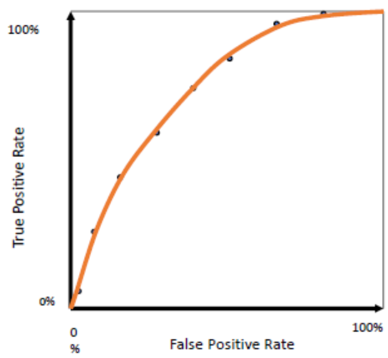



לעקום שקיבלנו קוראים: ROC Curve והוא **מייצג את יכולת הסיווג של ה-data** וממנו נוכל להסיק את המסווג האופטימלי ל-data לכל $\alpha$ נתון על ידי העברת **המשיק לעקום עם קו ששיפועו $\alpha$**.

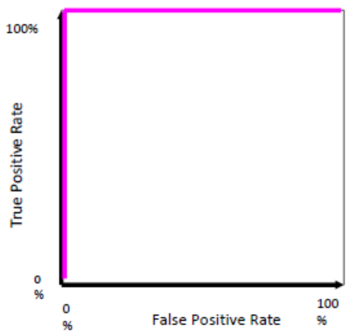
A good test:



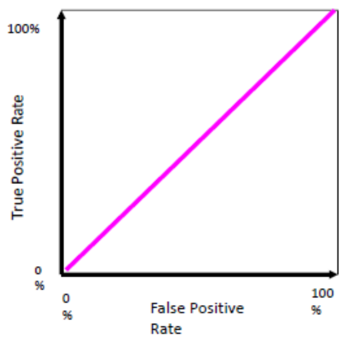

A poor test:




Best Test:




Worst test:




כדי למדוד את טיב ה-classifier אנחנו מסתגלים על השטח שמתחת לקו שנוצר. ככל שהשטח גדול יותר, כך ה-Roc Curve טוב יוצר, ולהפך.

<u>Best Test</u>

אין חיתוך בין ה-True ל-False.

<u>Worst Test</u>

חיתוך מלא בין ה-True ל-False.